Fundamental Storage Mechanisms for Location-based Services in Mobile Ad-hoc Networks

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik der Universität Stuttgart zur Erlangung der Würde eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von

Dominique Dudkowski

aus Ruit auf den Fildern

Hauptberichter:	Prof. Dr. rer. nat. Dr. h. c. Kurt Rothermel
Mitberichter:	Prof. Dr. rer. nat. habil. Pedro José Marrón
Tag der mündlichen Prüfung:	15. September 2009

Institut für Parallele und Verteilte Systeme (IPVS)
der Universität Stuttgart

2009

To my parents and the memory of my beloved Oma,
with love and gratitude

Abstract

The proliferation of mobile wireless communication technology has reached a considerable magnitude. As of 2009, a large fraction of the people in most industrial and emerging nations is equipped with mobile phones and other types of portable devices. Supported by trends in miniaturization and price decline of electronic components, devices become enhanced with localization technology, which delivers, via the Global Positioning System, the geographic position to the user. The combination of both trends enables location-based services, bringing information and services to users based on their whereabouts in the physical world, for instance, in the form of navigation systems, city information systems, and friend locators.

A growing number of wireless communication technologies, such as Wireless Local Area Networks, Bluetooth, and ZigBee, enable mobile devices to communicate in a purely peer-to-peer fashion, thereby forming mobile ad-hoc networks. Together with localization technology, these communication technologies make it feasible, in principle, to implement distributed location-based services without relying on any support by infrastructure components. However, the specific characteristics of mobile ad-hoc networks, especially the significant mobility of user devices and the highly dynamic topology of the network, make the implementation of location-based services extremely challenging. Current research does not provide an adequate answer to how such services can be supported. Efficient, robust, and scalable fundamental mechanisms that allow for generic and accurate services are lacking.

This dissertation presents a solution to the fundamental support of location-based services in mobile ad-hoc networks. A conceptual framework is outlined that implements mechanisms on the levels of routing, data storage, location updating, and query processing to support and demonstrate the feasibility of location-based services in mobile ad-hoc networks.

The first contribution is the concept of location-centric storage and the implementation of robust routing and data storage mechanisms in accordance with this concept. This part of the framework provides a solution to the problems of data storage that stem from device mobility and dynamic network topology. The second contribution is a comprehensive set of algorithms for location updating and the processing of spatial queries, such as nearest neighbor queries. To address more realistic location-based application scenarios, we consider the inaccuracy of position information of objects in the physical world in these algorithms.

Extensive analytical and numerical analyses show that the proposed framework of algorithms possesses the necessary performance characteristics to allow the deployment of location-based services in purely infrastructureless networks. A corollary from these results is that currently feasible location-based services in infrastructure-based networks may be extended to the infrastructureless case, opening up new business opportunities for service providers.

Zusammenfassung

Die Verbreitung mobiler drahtloser Kommunikationstechnologie hat ein beträchtliches Ausmaß erreicht: Im Jahre 2009 ist bereits ein großer Teil der Menschen in den Industrie- und Schwellenländern mit Mobiltelefonen sowie einer Vielzahl weiterer Arten von tragbaren Geräten ausgestattet. Unterstützt durch die technologischen Entwicklungen in der Miniaturisierung sowie dem Preisverfall elektronischer Komponenten werden Geräte zunehmend mit Lokalisierungstechnologien ausgerüstet, mit deren Hilfe die geographische Position von Benutzern ermittelt werden kann. Zu diesen Technologien zählen beispielsweise leistungsfähige integrierte Schaltungen, die mit Hilfe des satellitengestützten Global Positioning System (GPS) die geographische Position eines Benutzers mit hoher Genauigkeit ermitteln können. Durch die Verknüpfung dieser beiden Entwicklungen werden lokationsbasierte Dienste ermöglicht, die Benutzern Informationen und Dienste in Abhängigkeit von ihrem Ort in der physischen Welt zur Verfügung stellen können. Beispiele solcher Anwendungen sind Navigations- und Stadtinformationsysteme sowie Systeme zur gegenseitigen Lokalisierung von Personen.

Eine wachsende Zahl drahtloser Kommunikationstechnologien, darunter drahtlose lokale Netze (WLANs), Bluetooth und ZigBee, ermöglichen den mobilen Geräten eine Kommunikation nach dem Peer-to-Peer-Schema, wonach Geräte so genannte mobile Ad-hoc-Netze bilden. Gemeinsam mit den Lokalisierungstechnologien wird dadurch grundsätzlich die Umsetzung verteilter lokationsbasierter Dienste möglich, ohne dabei auf infrastrukturbasierte Netze zurückzugreifen. Die spezifischen Merkmale mobiler Ad-hoc-Netze, vor allem die beträchtliche Mobilität von Geräten und die hochdynamische Topologie dieser Netze, machen jedoch die Implementierung lokationsbasierter Dienste zu einer Herausforderung. Die aktuelle Forschung gibt keine hinreichende Antwort auf die Frage, wie solche Dienste in mobilen Ad-hoc-Netzen unterstützt werden können. Effiziente, robuste und zugleich skalierbare Grundverfahren, welche die Implementierung leistungsfähiger lokationsbasierter Dienste ermöglichen, fehlen gänzlich.

Die vorliegende Dissertation befasst sich mit dem Entwurf, der Implementierung und der Bewertung eines Rahmenwerkes, das grundlegende Verfahren für die Speicherung und Verwaltung von Daten in mobilen Ad-hoc-Netzen bereitstellt. Das Rahmenwerk trägt dabei insbesondere den Merkmalen mobiler Ad-hoc-Netze Rechnung, indem es Strategien und Mechanismen definiert, die diesen Merkmalen wirksam begegnen. Die Dissertation zeigt ferner, wie unter Ausnutzung der vorgestellten Grundverfahren solche Funktionalität bereitgestellt werden kann, die für lokationsbasierte Dienste von Bedeutung ist. Dies wird beispielhaft anhand räumlicher Anfragen gezeigt, wobei insbesondere die Ungenauigkeit der geographischen Positionen von Benutzern in der physischen Welt berücksichtigt wird. Ausführliche Messergebnisse zeigen, dass die betrachteten Verfahren als Grundlage für die Umsetzung einer Vielzahl von lokationsbasierten Diensten in mobilen Ad-hoc-Netzen geeignet sind.

Eigenschaften mobiler Ad-hoc-Netze

Die vorliegende Arbeit grenzt sich vom Stand der Forschung ab durch die Betrachtung von Grundverfahren der Datenspeicherung in reinen mobilen Ad-hoc-Netzen. Im Gegensatz zu den klassischen mobilen Netzen besitzen diese Netze eine Reihe spezifischer Merkmale, die den Entwurf von Verfahren für die Datenspeicherung besonders schwierig gestalten. Diese Eigenschaften umfassen die beschränkten Rechen-, Speicher- und Energiekapazitäten von Geräten, einer im Vergleich zu Festnetzen geringeren Bandbreite und der sich stets ändernden Qualität des Kommunikationskanals, sowie der Gerätemobilität und -dichte.

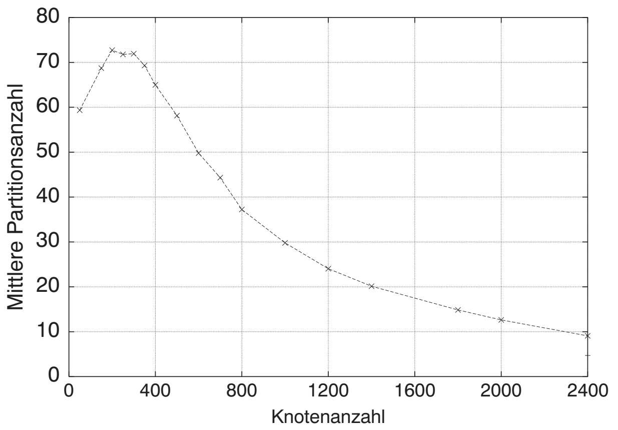
In Abhängigkeit von der zu lösenden Aufgabe beeinflussen diese Merkmale die Gestaltung von Verfahren in mobilen Ad-hoc-Netzen unterschiedlich stark. Aus diesen Grund ist zunächst zu untersuchen, welche der Eigenschaften für den Entwurf von Speicherverfahren besonders zu berücksichtigen sind. Hierbei zeigt sich, dass Gerätemobilität und -dichte den größten Einfluss ausüben, da die Dynamik dieser Größen unmittelbar zu Änderungen in der Netztopologie führen kann, wodurch die Datenübertragung über mehrere benachbarte Geräte erschwert wird.

Insbesondere kann die Mobilität bei einer geringen Gerätedichte zur Entstehung von Netzpartitionen führen, da die Kommunikationsreichweite von Geräten (im Folgenden Netzknoten oder Knoten genannt) nicht mehr ausreicht, um einen zusammenhängenden Netzgraphen zu bilden. Hingegen führt der nichtdeterministische Charakter mobiler Ad-hoc-Netze, die z.B. durch die Geräte von Personen in Stadtgebieten gebildet werden, auch zu der vorteilhaften Situation, bei der sich zuvor getrennte Netzpartitionen wieder vereinigen.

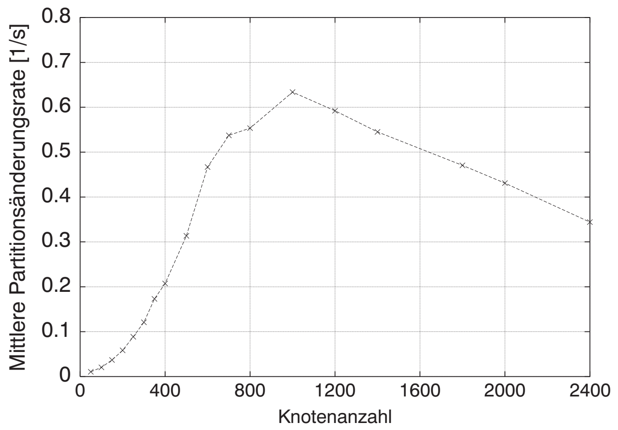
Grundsätzlich können Netzpartitionen die Datenspeicherung stark beeinträchtigen, da zwischen Partitionen naturgemäß weder Daten ausgetauscht noch Anfragen verarbeitet werden können. Aus diesem Grund ist ein Verständnis des Partitionierungsverhaltens mobiler Ad-hoc-Netze für den Entwurf möglichst robuster Speicherverfahren von grundlegender Bedeutung. Zu diesem Zweck stellt die Dissertation eine Menge von Partitionsmetriken vor, die in der Lage sind, verschiedene Eigenschaften der Netzpartitionierung geeignet zu beschreiben.

Im Folgenden wird dies beispielhaft anhand zweier Partitionsmetriken gezeigt. Um die Relevanz der Netzpartitionierung hervorzuheben wird zunächst die Partitionsanzahl in einem bestimmten Netzszenario betrachtet. Bei dieser Metrik handelt es sich um eine so genannte netzzentrische Metrik, da sie das Partitionsverhalten eines mobilen Ad-hoc-Netzes als Ganzes beschreibt. Abbildung 1.a zeigt die mittlere Partitionsanzahl in einem typischen Innenstadtszenario am Beispiel von Manhattan. Zunächst steigt die mittlere zu erwartende Anzahl der Partitionen stark an, da mehr und mehr einzelne Knoten isoliert voneinander existieren. Nach Durchschreiten eines Maximums vermindert sich mit steigender Dichte die Partitionsanzahl, was durch die allmähliche Verschmelzung kleinerer zu immer größeren Partitionen zu erklären ist. Insbesondere zeigt Abbildung 1.a, dass auch bei der höchsten gemessenen Dichte noch immer eine signifikante Anzahl von Netzpartitionen auftritt.

Zur Verdeutlichung der möglichen Dynamik des Partitionierungsverhaltens mobiler Ad-hoc-Netze wird anhand der Partitionsänderungsrate gezeigt, wie häufig ein einzelner Netzknoten seine eigene Partition wechselt. Da hierbei die Sicht eines einzelnen Knotens im Vordergrund steht, handelt es sich bei dieser Metrik um eine so genannte knotenzentrische Partitionsmetrik. Abbildung 1.b zeigt den Verlauf der mittleren knotenzentrischen Partitionsänderungsrate in



a. Mittlere Partitionsanzahl



b. Mittlere Partitionsänderungsrate

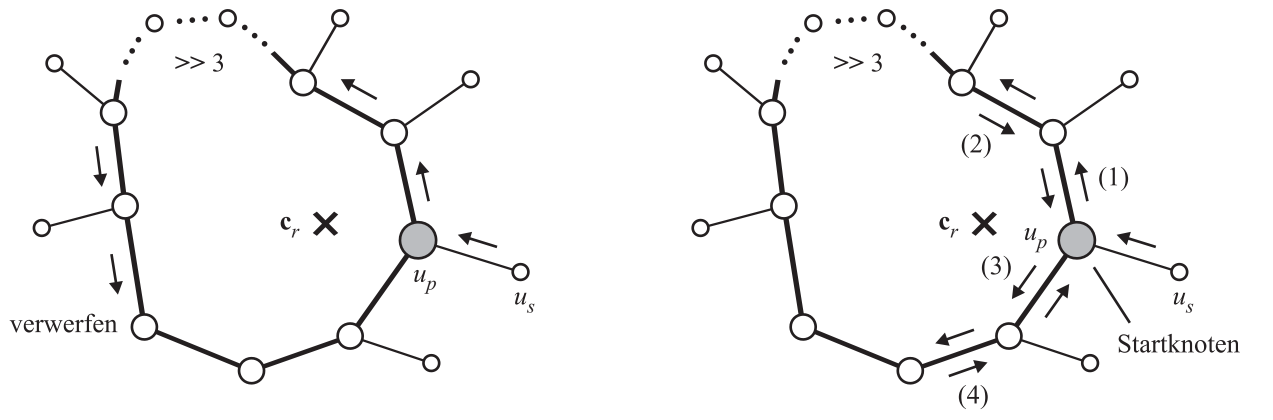
Abbildung 1: Mittlere Partitionsanzahl und knotenzentrische Partitionsänderungsrate in Abhängigkeit von der Knotenanzahl im Stadtgebiet von Manhattan.

Abhängigkeit von der Knotendichte. Während bei sehr geringen Dichten aufgrund der weit voneinander entfernten Knoten Änderungen der Partitionszugehörigkeit nur selten stattfinden, treten diese Änderungen bei mittleren Dichten häufig auf. Dieses Verhalten zeigt insbesondere, dass ein einzelner Knoten durchaus in der Lage ist, auch Knoten in anderen Partitionen bereits nach einer relativ kurzen Zeit erneut zu kontaktieren.

Die Ergebnisse im Hauptteil der Dissertation führen zu zwei Schlussfolgerungen. Einerseits ist das Auftreten von Netzpartitionen von den Verfahren der Datenspeicherung als häufig auftretender Normalfall anzusehen. Andererseits zeigt sich bei einer zumindest für eine Kommunikation sinnvollen Mindestknotendichte eine Dynamik, die zu einer endlichen Dauer von Partitionierungssituationen führt. Das bedeutet, dass Speicherverfahren eine Partitionierung zwar berücksichtigen müssen, den durch eine Partitionierung entstehenden möglichen Inkonsistenzen in der Datenverwaltung jedoch mit Hilfe geeigneter Maßnahmen entgegenwirken können.

Grundverfahren der Datenspeicherung

Um den Eigenschaften mobiler Ad-hoc-Netze wirksam entgegenzutreten, definiert die vorliegende Arbeit das Paradigma der lokationszentrischen Speicherung, das Ausgangspunkt für die Grundverfahren der Datenspeicherung ist. Bei diesem Ansatz werden räumliche Daten, also Daten mit einer zugehörigen räumlichen Position, auf denjenigen Knoten eines mobilen Ad-hoc-Netzes gespeichert, die sich in der Nähe einer geographischen Referenzposition befinden. Diese Position befindet sich ihrerseits in der Nähe derjenigen geographischen Position, die durch das Datum selbst definiert ist. Beide Nähebegriffe lassen sich nun so flexibel gestalten, dass geringe Knotendichten, hohe Mobilität und Netzpartitionierungen zeitweise toleriert werden können. Grundlegend ist hierbei auch der Begriff der räumlichen Kohärenz, welche definiert ist als die mittlere geographische Entfernung zwischen einem Datum und der zugehörigen Referenzposition. Diese Metrik ist für die Bewertung von Speicherverfahren grundlegend, und ein Verfahren ist umso effizienter, desto geringer diese Entfernung ist.



>> 3
$\mathbf{c}_r$
u_p
u_s
verwerfen
>> 3
(2)
(1)
$\mathbf{c}_r$
u_p
(3)
u_s
Startknoten
(4)
a. Verwerfung bei unidirektionaler Paketweiterleitung
b. Konvergenz bei bidirektionaler Paketweiterleitung

Abbildung 2: Unidirektionale und bidirektionale Paketweiterleitung entlang von langen Umkreisen (z.B. aufgrund von physischen Hindernissen).

Die in der Dissertation vorgestellten Speicherverfahren setzen die lokationszentrische Speicherung auf der Grundlage des Umkreises um. Bei diesem handelt es sich um eine Menge von Knoten, die um eine gegebene geographische Referenzposition einen geschlossenen Pfad bilden (Abbildung 2.a). Die in der Abbildung dargestellte Punktlinie deutet an, dass ein Umkreis beispielsweise beim Auftreten einer Netzpartitionierung unterbrochen sein kann.

Um den durch nicht geschlossene oder auch sehr lange Umkreise auftretenden Problemen entgegenzuwirken, vermeiden die Grundverfahren der Datenspeicherung Annahmen, die auf der Geschlossenheit und Endlichkeit von Umkreisen beruhen, vollständig. Im Gegensatz zu unidirektionalen Verfahren der Paketweiterleitung, wie sie z.B. in [KK00] betrachtet werden, verwenden die Grundverfahren die so genannte bidirektionale Paketweiterleitung, welche beispielhaft in Abbildung 2.b dargestellt ist. Ausgehend von einem Startknoten durchläuft dabei ein Datenpaket zunächst eine begrenzte Anzahl von Knoten in eine Richtung des Umkreises. Danach werden Knoten in der umgekehrten Richtung besucht, bis die gleiche Entfernung zum Startknoten erneut erreicht wird. Unabhängig von der Form eines Umkreises terminiert dieser Algorithmus stets und ein Paket wird schließlich zum Startknoten des Umkreises zurückgeleitet.

Im Gegensatz zu den auf unidirektionaler Paketweiterleitung beruhenden Verfahren, die z.B. in [RKY$^+$02, ACNP07] eingesetzt werden, ist dieser Knoten jedoch nicht unmittelbar als Speicherknoten geeignet, da sich aufgrund der Knotenmobilität im Gegensatz zu Sensornetzen dieser Knoten ständig ändert. Das Grundverfahren der Datenspeicherung geht deshalb davon aus, dass ein Speicherknoten grundsätzlich ein anderer als der Startknoten auf einem Umkreis sein kann, so wie dies beispielhaft in Abbildung 3.a dargestellt ist. Ein Speicherknoten übermittelt zunächst in die Richtung einer zugeordneten Referenzposition so genannte Bekanntmachungen, welche die Zuständigkeit des Knotens für diese Referenzposition signalisieren. Dies geschieht mit Hilfe der bidirektionalen Paketweiterleitung, während deren jeweils ein Bekanntmachungseintrag des Speicherknotens auf den besuchten Knoten gespeichert wird. Insbesondere wird so die Verfügbarkeit des Speicherknotens auf den Knoten eines Umkreises rund um die geographische Referenzposition veröffentlicht.

Anfragen an ein bestimmtes Datenobjekt, das auf einem Speicherknoten abgelegt ist, werden in zwei Schritten weitergeleitet. Im Gegensatz zu knotenzentrischen Ansätzen wird bei der loka-

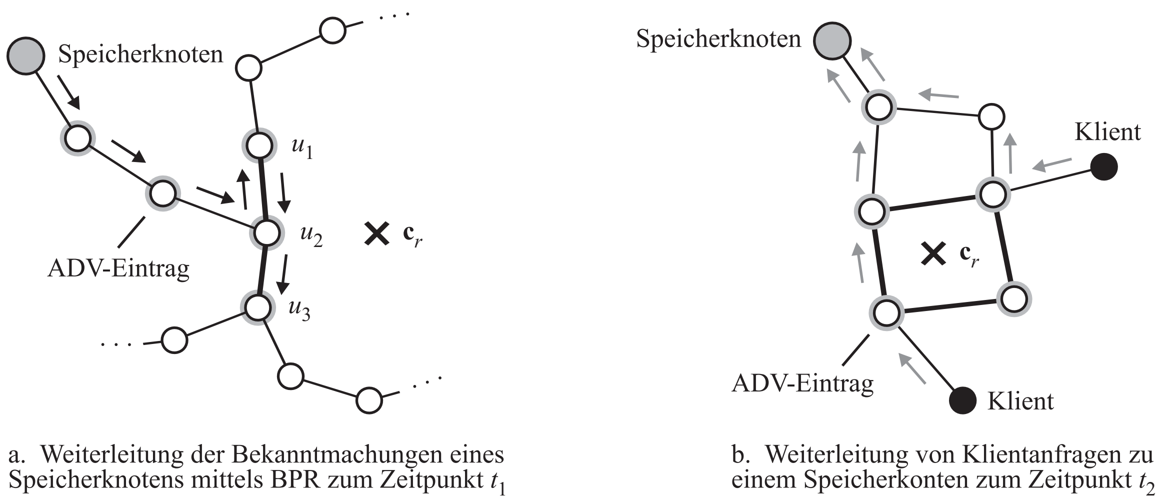

Speicherknoten
u_1
u_2
u_3
$\mathbf{c}_r$
ADV-Eintrag
a. Weiterleitung der Bekanntmachungen eines Speicherknotens mittels BPR zum Zeitpunkt t_1
Speicherknoten
Klient
$\mathbf{c}_r$
ADV-Eintrag
Klient
b. Weiterleitung von Klientanfragen zu einem Speicherkonten zum Zeitpunkt t_2

Abbildung 3: Weiterleitung der Bekanntmachungen von Speicherknoten.

tionszentrischen Speicherung jedoch das Datenobjekt nicht auf einen Knoten direkt, sondern zunächst auf die Referenzposition abgebildet, ähnlich wie dies in datenzentrischen Ansätzen (z.B. [RKY$^+$02, GGC03]) geschieht. Eine Anfrage wird in einem ersten Schritt in die Richtung der Referenzposition geleitet (Abbildung 3.b). Sobald ein mit der Referenzposition assoziierter Bekanntmachungseintrag eines Speicherknotens lokalisiert werden kann, wird die Anfrage in einem zweiten Schritt zu dem Speicherknoten weitergeleitet. Abbildung 3.b zeigt, wie Anfragen von Klienten im Falle eines sehr kleinen Umkreises schnell auf Bekanntmachungseinträge stoßen, woraufhin sofort eine Weiterleitung zum Speicherknoten stattfindet.

Weil ein Speicherknoten nur lose mit einem Umkreis und der umschlossenen Referenzposition assoziiert ist, muss aufgrund von Knotenmobilität zur Erhaltung der räumlichen Kohärenz regelmäßig ein neuer Speicherknoten bestimmt werden. Hierzu stellt die Dissertation geeignete Verfahren zur Verfügung, die eine Migration der von einem Speicherknoten verwalteten Daten bei Abschwächung der räumlichen Kohärenz vornehmen. Die Migrationsverfahren lassen sich dabei unterteilen in Strategien einerseits und Mechanismen andererseits. Erstere dienen dazu, Entscheidungen zu treffen hinsichtlich der Notwendigkeit, des Nutzens und des zu erwartenden Erfolgs einer potentiellen Datenmigration. Dies ist wichtig, da eine Migration von Daten zu einem Zielknoten beispielsweise nur dann sinnvoll ist, wenn die Netztopologie zwischen Ausgangs- und Zielknoten ausreichend stabil ist, sodass eine erfolgreiche Migration aller Daten mit hoher Wahrscheinlichkeit erwartet werden kann.

Die Migrationsstrategien werden dabei weiter unterteilt in solche, die den lokalen Zustand des migrierenden Knotens analysieren, und erweiterte Strategien, die auch entfernte Knoten berücksichtigen, um letztlich einen geeigneten Zielknoten der Migration zu bestimmen. Die lokalen Migrationsstrategien untersuchen, ob eine Migration ausschließlich aus der Sicht des zu migrierenden Knotens notwendig und nützlich ist, das heißt, ob eine Migration mit hoher Wahrscheinlichkeit zu einer Verbesserung der räumlichen Kohärenz führt. Die vorläufige Betrachtung von ausschließlich dem aktuellen Speicherknoten ist dabei von Bedeutung, da dieser Zustand ohne Kommunikation erfasst werden kann. In diese Bewertung fließen die aktuelle Distanz d_0 des Speicherknotens zur Referenzposition ein, sowie die Zeitdauer t_0, mit welcher der aktuelle Speicherknoten bereits für die Speicherung von Daten zuständig war. Anhand dieser Größen wird das Prädikat P_{MRP} bestimmt, das die notwendige Bedingung einer Migration definiert und genau dann wahr ist, wenn d_0 oder t_0 einen durch d_{thresh} und t_{thresh} gegebenen räumlichen bzw. zeitlichen Grenzwert überschreitet:

$$P_{\mathrm{MRP}} := d_0 > d_{\mathrm{thresh}} \vee t_0 > t_{\mathrm{thresh}} \tag{1}$$

Nur bei einem wahren Prädikat wird in einem zweiten Schritt eine Menge entfernter Knoten betrachtet, um einen Zielknoten zu bestimmen, der für den Empfang der zu migrierenden Daten geeignet ist (im anderen Fall wird nach Ablauf eines Zeitintervalls eine erneute Prüfung des Prädikats in (1) vorgenommen). Grundsätzlich sind in diesem Schritt solche Knoten zu bevorzugen, die sich näher an der Referenzposition aufhalten. Ebenfalls zu berücksichtigen ist die Geschwindigkeit von Knoten, da sich Knoten mit hoher Geschwindigkeit schneller von der Referenzposition entfernen als langsame Knoten, und somit bereits nach kurzer Zeit erneut eine Migration erforderlich ist. Ferner spielt die Zeitdauer eine Rolle, mit der ein Knoten bereits in der Vergangenheit als Speicherknoten fungierte. Dies ist vor allem für die Lastverteilung zwischen möglichst allen Knoten eines mobilen Ad-hoc-Netzes von Bedeutung. Zusammenfassend lässt sich aus diesen Betrachtungen die Gesamteignung eines Knotens bestimmen. Diese setzt sich multiplikativ aus den individuellen Eignungen $\epsilon_i^d, \epsilon_i^{\delta t}$ und $\epsilon_i^{\Delta t}$ zusammen, welche die zuvor beschriebenen Eigenschaften modellieren:

$$\epsilon_i := \epsilon_i^d \cdot \epsilon_i^{\delta t} \cdot \epsilon_i^{\Delta t} \tag{2}$$

Neben der Bestimmung der Eignungswerte ϵ_i einer Menge von Knoten ist die Stabilität der möglichen Netzpfade zu diesen Knoten eine wichtige Größe, da die Eignung der Knoten allein nicht für die Bewertung einer erfolgreichen Migration ausreicht. Die Dissertation stellt hierfür einen Algorithmus zur Verfügung, der effizient möglichst stabile Pfade zwischen dem aktuellen Speicherknoten und der Menge möglicher Zielknoten einer Migration bestimmt. Gemeinsam mit den Eignungswerten ϵ_i wird am Ende des zweiten Bewertungsschrittes ein Knoten als Migrationsziel endgültig festgelegt. Wird kein geeigneter, über einen ausreichend stabilen Netzpfad erreichbarer Knoten gefunden, so beginnt die Auswertung der Strategien nach Ablauf eines zweiten Zeitintervalls von Neuem.

Für die anschließende Migration vom Ausgangs- zum Zielknoten über den gewählten Netzpfad stellt die Dissertation einen Mechanismus vor, der Migrationen möglichst effizient ausführt und das Auftreten von Migrationsfehlern vermeidet. Bei diesem Mechanismus wird insbesondere die Distanz zwischen Ausgangs- und Zielknoten berücksichtigt, um den Datentransfer aufgrund von Eigenschaften des gemeinsamen drahtlosen Kommunikationsmediums zu optimieren. Nach erfolgter Migration aller Daten wird der ursprüngliche Speicherknoten deaktiviert und der Zielknoten übernimmt dessen Aufgabe. Während dieser Übergabe wird auch die Versendung von Bekanntmachungen konsistent vom ursprünglichen auf den zukünftigen Speicherknoten übertragen, sodass dieser seine Zuständigkeit im Netz so mitteilt, dass Anfragen von Klienten nun an diesen neuen Knoten geleitet werden.

Aufgrund der Problematik der Netzpartitionierung wird der Migrationsmechanismus von einem Konsolidierungsmechanismus ergänzt, der in der Lage ist, auftretende Redundanzen von Speicherknoten aufzulösen. Aufgrund kommunikationstheoretischer Eigenschaften und unter bestimmten Fehlermodellen kann ein Migrationsprozess in einer solchen Art fehlschlagen, dass

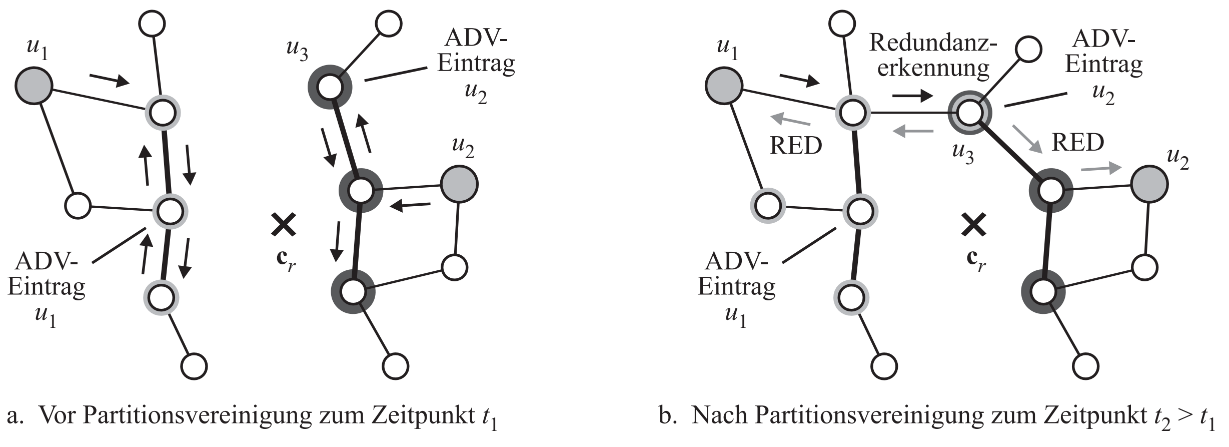


Abbildung 4: Erkennung von redundanten Speicherknoten.

eine Entscheidung, welcher Speicherknoten die Datenspeicherung fortführt, nicht möglich ist. Dieser Sachverhalt ist für den Fall einer Netzpartitionierung in Abbildung 4.a dargestellt. In jeder Partition existiert nach einer fehlgeschlagenen Migration jeweils ein Speicherknoten. Beide Speicherknoten sind in dieser Situation dazu bestimmt, ihre eigenen Bekanntmachungen in die Richtung der geographischen Referenzposition zu senden.

Der Konsolidierungsmechanismus ist in solchen Situationen in der Lage, Redundanzen ohne zusätzliche Kommunikation zu erkennen, sobald eine Wiedervereinigung von Partitionen stattfindet. Abbildung 4.b zeigt die Vereinigung der zuvor dargestellten Partitionen. Da nun ein einziger Umkreis die Referenzposition umschließt, führt das Bekanntmachungsverfahren dazu, dass Bekanntmachungen beider Speicherknoten schließlich in einem Knoten (Knoten u_3 in Abbildung 4.b) zusammentreffen. Dieser Knoten kann somit unmittelbar eine Redundanz feststellen, die er sodann an die beteiligten Speicherknoten signalisiert. Die Speicherknoten ihrerseits stoßen daraufhin einen Konsolidierungsprozess an, der nach einem ähnlichen Verfahren wie dem der Migration verfährt. Nachdem einer der Speicherknoten seinen Datenbestand zum anderen übermittelt hat, werden beide Datenbestände verschmolzen und anschließend der übermittelnde Speicherknoten, ähnlich wie im Falle der Migration, abgelöst. Im Ergebnis existiert somit nach einer Konsolidierung lediglich ein Speicherknoten.

Hinsichtlich der Grundverfahren stellt die Dissertation ausführliche analytische und simulative Ergebnisse zur Verfügung. Ein Auszug aus der simulativen Leistungsbewertung zeigt, dass die Grundverfahren der Datenspeicherung effizient, robust und skalierbar sind. Für die Bewertung wird eine Gesamtfläche von $600 \cdot 600$ m^2 angenommen, in deren Zentrum eine Zelle der Größe $200 \cdot 200$ m^2 platziert ist. Insgesamt bewegen sich 150 Knoten nach dem Random-Waypoint-Mobilitätsmodell mit einer Geschwindigkeit von 1,5 m/s und Verweilzeit von 30 s innerhalb des Simulationsgebiets. Die Kommunikationsreichweite der Knoten ist 100 m.

Zunächst wird die Leistungsfähigkeit der Grundverfahren anhand der Erfolgsrate von Klientanfragen dargelegt. Die Erfolgsrate von Anfragen ist definiert als derjenige Bruchteil von Anfragen, die erfolgreich zu einem Speicherknoten übermittelt werden können. Als Vergleichsverfahren wurde die datenzentrische Speicherung (DCS/GPSR) nach [RKY$^+$02] gewählt, um die Vorteile des lokationszentrischen Ansatzes dieser Dissertation zu zeigen. Für das datenzentrische Verfahren ist die Erfolgsrate derjenige Bruchteil von Anfragen, die erfolgreich am Startknoten des Umkreises (Abbildung 2), der den Speicherknoten definiert, eintreffen.

Zur Darlegung der Leistungsfähigkeit der Grundverfahren vor allem bei geringen Knotendichten zeigt Abbildung 5 die Erfolgsrate von Anfragen zweier LCS-Varianten sowie des DCS/GPSR-basierten Verfahrens in Abhängigkeit von der Knotenanzahl. Die Unterscheidung der beiden LCS/BPR-Varianten erfolgt anhand unterschiedlicher Methoden, mit denen Bekanntmachungen von Speicherknoten verteilt werden. Die Verteilung geschieht entweder mit Hilfe der bidirektionalen Paketweiterleitung (LCS/BPR) oder durch einfaches Fluten (LCS/Geocast). Abbildung 5 zeigt, dass beide LCS-Varianten bei sehr geringen Knotendichten eine Erfolgsrate von über 94% erzielen. Im Gegensatz dazu fällt die Leistung von DCS/GPSR bei 70 und weniger Knoten stark ab. Dies ist mit der ansteigenden Länge von Umkreisen bei abnehmender Knotendichte zu erklären, wodurch das Durchlaufen eines vollständigen Umkreises für DCS/GPSR immer aufwändiger wird. Bemerkenswert ist die Beobachtung, dass für eine Knotenanzahl von über 120 Knoten die Leistungsfähigkeit von DCS/GPSR ebenfalls abnimmt, während sie für beide LCS-Varianten konstant hoch bleibt. Dies hängt mit den Eigenschaften der für die geometrische Weiterleitung notwendigen Planarisierungsverfahren für Netzgraphen zusammen, die einen weitaus geringeren Einfluss auf die Erfolgsrate von LCS als von DCS haben.

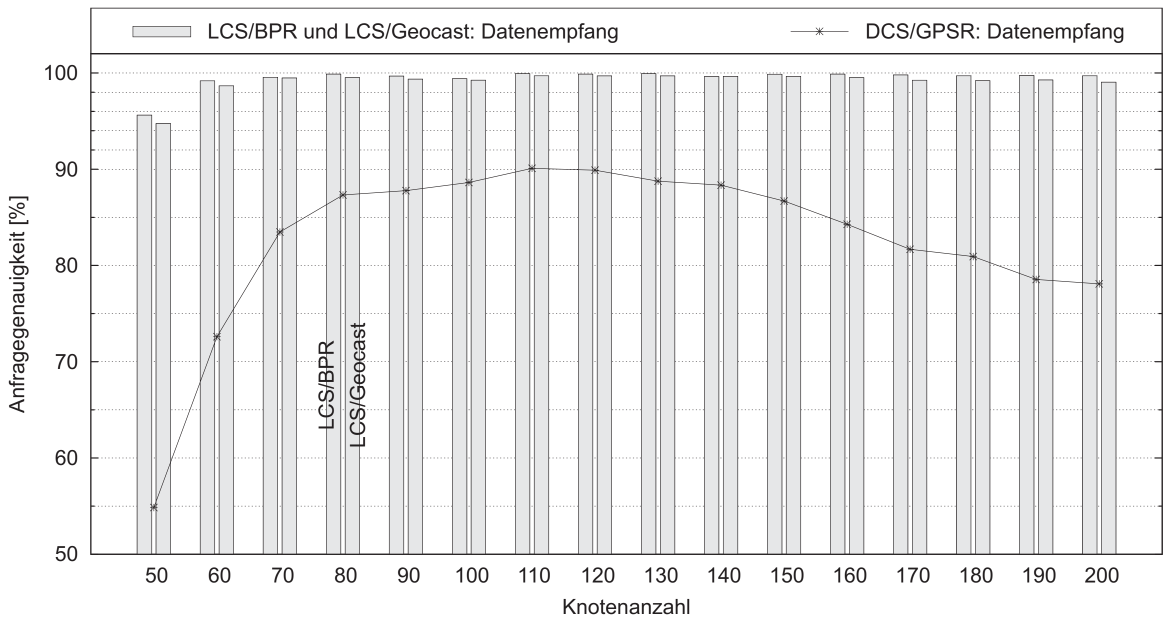


Abbildung 5: Anfragegenauigkeit des untersuchten Grundverfahrens der Datenspeicherung in Abhängigkeit von der Knotenanzahl.

Zur Beurteilung der Datenmigration wird im Folgenden die Robustheit von Migrationen im Vergleich zu einigen naheliegenden Verfahren untersucht. Hierbei muss unterschieden werden zwischen Migrationsfehlern, die ohne zusätzliche Maßnahmen aufgelöst werden können, und solchen, die zu den nicht unmittelbar auflösbaren Redundanzen der Speicherknoten führen. Im ersten Fall ist es möglich, dass die jeweils an einer Migration beteiligten Speicherknoten einen Migrationsfehler selbständig feststellen und sich in ihren jeweiligen Anfangszustand zurückführen. Die zweite Art von Fehlern führt jedoch zu den eingangs beschriebenen Redundanzen, die nur durch den zusätzlichen Konsolidierungsmechanismus aufgelöst werden können. Dieser Mechanismus wird jedoch nicht von den Vergleichsverfahren eingesetzt, sodass sich die nicht auflösbaren Redundanzen bei diesen Verfahren über die Zeit vervielfältigen.

Abbildung 6 zeigt die Häufigkeit kritischer Migrationsfehler in einem typischen Szenario. Hierbei wurde angenommen, dass während einer Migration eine Datenmenge von insgesamt 320 Kilobytes übertragen wird, die bei einer Größe von 32 Bytes pro Datenobjekt und einer Paketkapazität von 1500 Bytes einer Gesamtzahl von 212 Datenpaketen entspricht. Ferner wurde eine Simulationszeit von 3600 Sekunden angenommen. Abbildung 6 zeigt zwei Varianten des in der Dissertation vorgestellten Migrationsverfahrens, die zwei weiteren naheliegenden Verfahren gegenübergestellt werden. Während auf die spezifischen Unterschiede der einzelnen Verfahren an dieser Stelle verzichtet werden kann, ist von Bedeutung, dass die in der Dissertation entwickelten Verfahren in Abhängigkeit von der Knotenanzahl (d.h. der Knotendichte) im Gegensatz zu den naheliegenden Verfahren eine wesentlich höhere Robustheit besitzen.

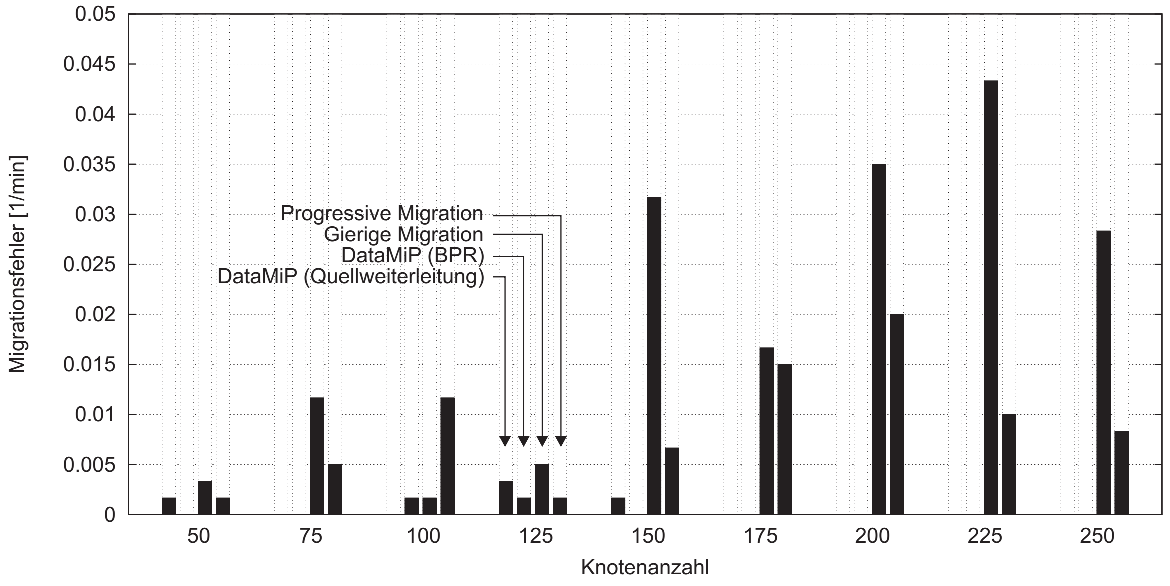


Abbildung 6: Mittlere Anzahl kritischer Migrationsfehler der vier untersuchten Migrationsmechanismen in Abhängigkeit von der Knotenanzahl.

Während die Anzahl der kritischen Migrationsfehler im untersuchten Szenario grundsätzlich sehr niedrig ist, muss berücksichtigt werden, dass ein realistisches Szenario aus einer Vielzahl von Referenzpositionen und somit Speicherknoten besteht und sich über einen langen Zeitraum erstreckt. Wird das Szenario der Simulation linear auf eine Gebietsgröße von $2 \cdot 2$ km^2 skaliert, so führt dies bei einer Fehlerrate von 0.005 Fehlern pro Minute im Falle der naheliegenden Verfahren zu etwa 30 kritischen Fehlern pro Stunde. Da kritische Fehler von diesen Verfahren nicht behandelt werden, entwickeln sich so über einen längeren Zeitraum erhebliche Redundanzen von Speicherknoten. Im Gegensatz dazu tritt bei den vorgestellten Varianten der Datenmigration eine sehr viel geringere Anzahl kritischer Fehler auf, die außerdem durch den Konsolidierungsmechanismus aufgelöst werden. Dadurch ergeben sich auch bei sehr großen Flächen und langen Zeiträumen nur zeitlich und geographisch punktuelle Redundanzen.

Lokationsaktualisierung und Anfrageverarbeitung

Die im letzten Abschnitt beschriebenen Grundverfahren sind in der Lage, den speziellen Eigenschaften mobiler Ad-hoc-Netze so entgegenzuwirken, dass eine robuste, effiziente und skalierbare Datenspeicherung ermöglicht wird. Für lokations- und kontextbasierte Dienste und Anwendungen sind diese Verfahren jedoch nicht von unmittelbarer Bedeutung, da nur einzelne Datenobjekte ohne Berücksichtigung ihrer Semantik aktualisiert und angefragt werden können. Das in der Dissertation vorgestellte Rahmenwerk umfasst deshalb zusätzlich Dienste, die erweiterte Funktionalität bereitstellen. Im Folgenden wird gezeigt, wie Verfahren zur Lokationsaktualisierung einerseits und zur Verarbeitung räumlicher Anfragen andererseits durch Ausnutzung der grundlegenden Speicherverfahren umgesetzt werden können.

Wichtig für die Entwicklung von Verfahren zur Lokationsaktualisierung ist die Erkenntnis, dass jedes Modell der physischen Welt mit Hilfe von Sensorik nur innerhalb bestimmter Genauigkeitsgrenzen erfasst werden kann. Insbesondere gilt dies für die Position physischer Objekte, denen stets eine Ungenauigkeit anhaftet. Positionsungenauigkeiten können mit Hilfe einer Wahrscheinlichkeitsdichte und einem zugehörigen kreisförmigen Gebiet beschrieben werden, innerhalb dessen sich das Objekt mit einer gewissen Wahrscheinlichkeit aufhält. Die Lokation **L** eines Objektes lässt sich formal wie folgt darstellen:

$$\mathbf{L} := (\varrho(X), \mathrm{CEP}_Y) \tag{3}$$

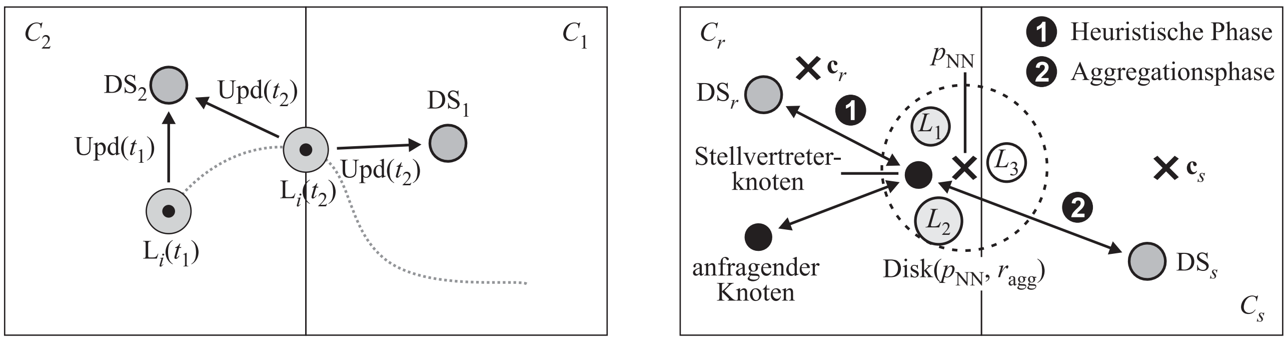
In (3) bezeichnet $\varrho(X)$ die Wahrscheinlichkeitsdichte und CEP_Y das so genannte circular error probable (CEP) für die Wahrscheinlichkeit Y, mit der ein Objekt innerhalb des durch CEP_Y definierten kreisförmigen Gebietes lokalisiert werden kann.

Auf der Grundlage der durch (3) gegebenen Semantik von Lokationen eines Objekts definiert die Dissertation einen Algorithmus zur Lokationsaktualisierung, dessen Funktionsweise im Folgenden anhand von Abbildung 7 erläutert wird. Das geographische Gebiet, innerhalb dessen sich das mobile Ad-hoc-Netz befindet, ist dabei in Zellen unterteilt, für die jeweils ein Speicherknoten (DS) zuständig ist. Jeder Knoten ist in der Lage, die Lokation eines Objekts mit Hilfe seiner eigenen Sensorik zur Positionsbestimmung zu erfassen und eine Lokationsinformation nach (3) zu erzeugen. Sodann wird diejenige Menge von Zellen bestimmt, die mit dem durch CEP_Y definierten kreisförmigen Lokationsgebiet überlappen. In Abbildung 7.a überlappt beispielsweise Lokationsgebiet $L_i(t_1)$ mit Zelle C_2, $L_i(t_2)$ mit den Zellen C_2 und C_1.

Im nächsten Schritt wird jeweils eine Kopie der Lokationsinformation an die für die zuvor bestimmten Zellen zuständigen Speicherknoten gesendet. An dieser Stelle greift das Verfahren der Lokationsaktualisierung auf die indirekte Paketweiterleitung des Grundverfahrens zurück, das die Lokationsinformationen in zwei Schritten an die zuvor bestimmten Zellen und danach an die Speicherknoten DS_2 und DS_1 weiterleitet. Die empfangenen Lokationsinformationen des aktualisierten Objekts werden sodann in der lokalen Datenbank des jeweiligen Speicherknotens aktualisiert, in der sie für die weitere Verarbeitung von Anfragen zur Verfügung stehen.

Das Verfahren zur Lokationsaktualisierung stellt ferner Methoden bereit, die Kopien eines Datenobjekts auf mehreren Speicherknoten löscht, wenn diese nach einer erneuten Lokationsaktualisierung nicht mehr auf die entsprechenden Zellen abbilden. Damit wird stets eine möglichst hohe Konsistenz von Datenobjekten erzielt, wodurch insbesondere die Genauigkeit von den im Folgenden betrachteten räumlichen Anfragen erhöht wird.

Am Beispiel von Nachbarschaftsanfragen soll nun gezeigt werden, wie effiziente Algorithmen für die Verarbeitung von räumlichen Anfragen in mobilen Ad-hoc-Netzen effizient gestaltet werden können. Dabei ist zu beachten, dass Positionsungenauigkeiten auch bei der Definition einer geeigneten Semantik für Nachbarschaftsanfragen zu berücksichtigen sind. Hierzu



a. Lokationsaktualisierungen zu den Zeitpunkten t_1, t_2 b. Beispiel einer Nachbarschaftsanfrage mit $k = 2$

Abbildung 7: Funktionsweise von Lokationsaktualisierung und Nachbarschaftsanfragen.

wird in der Dissertation unter Verwendung von Begriffen aus der Wahrscheinlichkeitstheorie ein Nähebegriff definiert, der für jeweils zwei Objekte beschreibt, welches dieser Objekte sich mit welcher Wahrscheinlichkeit näher zu einer gegebenen Anfrageposition aufhält. Dieser Nähebegriff lässt sich formal wie folgt fassen:

$$o_j < o_k|_{p_{\text{NN}}} \Leftrightarrow P_{jk} > 0.5 \cdot P_{\max} \tag{4}$$

Für zwei Objekte o_j, o_k befindet sich nach (4) das Objekt o_j genau dann näher an der Anfrageposition p_{NN} als o_k, wenn die Wahrscheinlichkeit, mit der o_j das näher gelegene Objekt ist, größer ist als die Hälfte der maximal möglichen Wahrscheinlichkeit $P_{\max}$. Der Wert für $P_{\max}$ ergibt sich aus der paarweisen Integration der Wahrscheinlichkeitsdichten ϱ_j, ϱ_k der beiden Objektlokationen. Unter Verwendung von (4) lässt sich das Ergebnis R_{PNQ} einer wahrscheinlichkeitsbasierten Nachbarschaftsanfrage definieren, das genau k Objekte enthält:

$$R_{\text{PNQ}} \ := \ \{o_1, \ldots, o_k\} \ : \ \forall o_r \in R_{\text{PNQ}}, o_s \notin R_{\text{PNQ}} : o_r < o_s\,|_{p_{\text{NN}}} \tag{5}$$

Für den Entwurf eines geeigneten Verfahrens zur Verarbeitung wahrscheinlichkeitsbasierter Nachbarschaftsanfragen ist von Bedeutung, dass zwar die Anzahl k der zu bestimmenden Objekte bekannt ist, nicht aber, bis zu welcher geographischen Entfernung zur Anfrageposition sich diese Objekte in der physischen Welt aufhalten. Der in der Dissertation vorgestellte Algorithmus verwendet deshalb ein zweiphasiges Verfahren, das im Folgenden anhand von Abbildung 7.b beispielhaft für eine Nachbarschaftsanfrage mit $k = 2$ erläutert wird.

Zunächst wird von dem anfragenden Knoten die Anfrage an einen Stellvertreterknoten gesendet, der die eigentliche Anfrageverarbeitung ausführt. Dieses Vorgehen erhöht die Effizienz der Anfrageverarbeitung, da die Aggregation des Endergebnisses in räumlicher Nähe zu den zu lokalisierenden Objekte ausgeführt wird. Sodann wird in der ersten Phase, der so genannten heuristischen Phase, eine begrenzte Anzahl von Speicherknoten abgefragt, um eine genäherte Menge von Objekten zu erhalten. In Abbildung 7.b wird während der heuristischen Phase der Speicherknoten der Zelle C_r mit Hilfe einer partiellen Nachbarschaftsanfrage angefragt. Unter Anwendung von (5) liefert der Speicherknoten DS_r als Teilergebnis die beiden Objekte zurück, die sich am nächsten zur Anfrageposition p_{NN} aufhalten. Im Beispiel sind dies die mit den Lokationen L_1 und L_2 gekennzeichneten Objekte.

Nach der Bestimmung der Näherungsmenge von Datenobjekten werden in der anschließenden Aggregationsphase diejenigen Objekte festgestellt, die sich räumlich näher an der Referenzposition befinden als die zuvor bestimmten. Dazu wird ein Kreis mit Mittelpunkt p_{NN} konstruiert, der die Lokationsgebiete der zuvor bestimmten Objekte enthält (Abbildung 7.b). Anhand dieses Kreises lässt sich durch Überlapp mit weiteren Zellen bestimmen, welche Speicherknoten Objekte verwalten, die noch näher an der Anfrageposition liegen als die bereits bestimmten Objekte. Im Beispiel existiert ein solches Objekt auf DS_s, dargestellt durch Lokationsgebiet L_3. Während der Aggregationsphase werden schließlich weitere partielle Nachbarschaftsanfragen versendet, um die verbleibenden Speicherknoten zu kontaktieren. Im Beispiel ist dies DS_s, der daraufhin das Objekt mit der Lokation L_3 an den Stellvertreterknoten zurückliefert. Der Stellvertreterknoten führt die durch partielle Anfragen erhaltenen Objekte schließlich mittels (4) so zusammen, dass nur die tatsächlich nächsten k Objekte im Endergebnis der Anfrage enthalten sind. In Abbildung 7.b sind dies L_3 und L_1, während L_2 nach Aggregation nicht mehr Bestandteil des Ergebnisses ist. Am Ende wird das Endergebnis der Nachbarschaftsanfrage an den anfragenden Knoten zurückgesendet.

Im Folgenden wird die Leistungsfähigkeit der Verarbeitung von Nachbarschaftsanfragen anhand einer Beispielmessung aus dem Hauptteil der Dissertation aufgezeigt. Das durch das mobile Ad-hoc-Netz abgedeckte geographische Gebiet hat eine Fläche von $600 \cdot 600$ m^2, welches in vier Zellen der Größe $300 \cdot 300$ m^2 unterteilt ist. Die Bewegung der Knoten wird wie im Falle der Speicherverfahren mit Hilfe des Random-Waypoint-Mobilitätsmodells beschrieben, das mit einer Knotengeschwindigkeit von 1.5 m/s und einer Verweilzeit von 30 s parametrisiert ist. Ferner wird eine Positionsungenauigkeit von 5 m angenommen, was den Eigenschaften moderner GPS-Empfänger entspricht. Der Anfrageparameter ist $k = 3$.

Zur Qualitätsbewertung von Anfrageergebnissen werden zwei Metriken verwendet. Einerseits wird durch die Anfragegenauigkeit definiert, welcher Bruchteil der zurückgelieferten Objekte mit einer idealen Ergebnismenge übereinstimmt. Die ideale Ergebnismenge ist dabei definiert als das Resultat der Ausführung des Anfragealgorithmus unter idealen Bedingungen, bei der eine zentrale Datenbank sowie eine verzögerungsfreie Lokationsaktualisierung und Anfrageverarbeitung angenommen werden. Andererseits beschreibt der Anfrageversatz, um welche Anzahl von Objekten ein Anfrageergebnis vom idealen Anfrageresultat entfernt ist. Beispielsweise ist der Anfrageversatz genau eins, falls die zurückgelieferten Objekte einer 3-Nachbarschaftsanfrage vollständig in der Ergebnismenge einer idealen 4-Nachbarschaftsanfrage liegen.

Abbildung 8 zeigt die Ergebnisse der Anfragegenauigkeit und des Anfrageversatzes als Funktion der Anzahl der Objekte, die mit der Aktualisierungshäufigkeit korreliert, für drei unterschiedliche Zellgrößen. Die Ergebnisse zeigen, dass in einem Bereich von bis zu 300 Datenobjekten, d.h. bis zu einer Aktualisierungshäufigkeit von 80/s, die Anfragegenauigkeit und der Anfrageversatz knapp unterhalb der maximal möglichen Werte liegen. Ab einer Anzahl von 300 Datenobjekten zeigt sich ein starker Abfall der Anfragegenauigkeit. Dies hängt mit dem Einfluss der Positionsungenauigkeiten bei hohen Objektdichten zusammen. Wird die Objektdichte erhöht, so erhöht sich ebenso die Wahrscheinlichkeit von sich überlappenden Lokationsgebieten unterschiedlicher Objekte. Aufgrund von diskreten Aktualisierungsereignissen ergeben sich ständig Änderungen in der Art und Weise, wie sich Objektlokationen überlappen. Aufgrund der Tatsache, dass Anfragen zu ihrer Bearbeitung eine gewisse Zeit in Anspruch nehmen, führen die während dieser Bearbeitungszeit auftretenden Aktualisierungsereignisse zu einer größeren

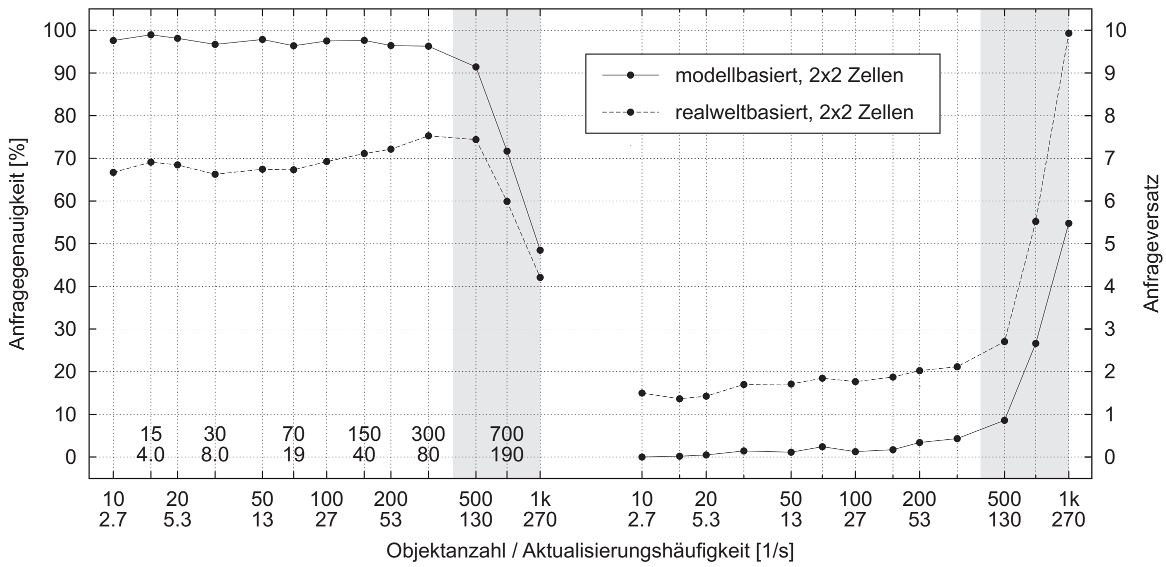


Abbildung 8: Anfragegenauigkeit und Anfrageversatz von k-Nachbarschaftsanfragen ($k = 3$) in Abhängigkeit von der Objektanzahl bzw. Aktualisierungshäufigkeit.

Anzahl an Inkonsistenzen, je höher die Objektdichte ist. Dennoch lässt sich erkennen, dass sich der Anfrageversatz relativ zur Anzahl der Objekte stark in Grenzen hält. Dies zeigt, dass auch bei hohen Dichten räumlich sehr nahe Objekte zurückgeliefert werden. Dadurch ergibt sich trotz der Positionsungenauigkeiten ein großer Nutzen für lokationsbasierte Anwendungen, da diese häufig nicht auf ideale Ergebnisse angewiesen sind.

Zusammenfassung und Ausblick

Die in dieser Dissertation vorgestellten Verfahren bilden ein umfassendes Rahmenwerk für die robuste, effiziente und skalierbare Speicherung und Verarbeitung dynamischer Daten in mobilen Ad-hoc-Netzen. Auf der einen Seite tragen die vorgestellten Speicherverfahren den speziellen Merkmalen mobiler Ad-hoc-Netze Rechnung, vor allem, deren Mobilität und Dichte und der daraus resultierenden Gefahr der Netzpartitionierung. Auf der anderen Seite sind die Verfahren zur Lokationsaktualisierung und Anfrageverarbeitung in der Lage, unter Ausnutzung der Speicherverfahren die für lokations- und kontextbasierte Dienste wichtigen Grundfunktionalitäten zur Verfügung zu stellen. Durch die Trends im Bereich des Internets, in dem infrastrukturbasierte und mobile Netze in einem hybriden Netzverbund verschmelzen werden, ist damit ein wichtiger Grundstein gelegt, um im Bereich der Mobilkommunikation existierende Anwendungen auch in die Domäne der mobilen Ad-hoc-Netze zu übertragen.

Acknowledgements

Publishing my work would not have been possible without a great deal of help at every stage along the journey from conception to its realization in this final form.

This dissertation has been developing in my mind for several years, and I would like to thank my advisor, Professor Kurt Rothermel, for guiding me through this intriguing and challenging topic. I am also thankful to my co-advisor, Professor Pedro José Marrón, for the support and feedback while being my colleague and my friend at the Distributed Systems Group.

Many colleagues helped me shape my ideas over the years, and I must thank them for their fellowship and priceless comments on my work. I am fortunate to have worked with wonderful people, such as Martin Bauer, Christian Becker, Susanne Bürklen, Frank Dürr, Tobias Farrell, Lars Geiger, Jörg Hähner, Boris Koldehofe, Ralf Lange, Steffen Maier, Annemarie Rösler, Harald Weinschrott, and many others, to whom I offer my heartfelt thanks.

During my time at the Distributed Systems Group, I had the opportunity to work with some bright and talented students who greatly inspired my work and with whom I shared some good times. Many thanks to you guys.

Last but not least, I would like to thank the German Research Foundation for their financial support through the Nexus project, which enabled my research in the first place.

In writing this dissertation, as in all else, I am specially indebted to my parents Stan and Anita, my grandmother Maminka, my aunt Brigitte, my best friend Holger, and my partner Elena, for their patience and sustained moral support, and for granting me in the past years the privilege of worrying about nothing else but my work.

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

1.1 Motivation

Paris, rush hour, Saturday, September 12, 2015 - Crowds of people populate the city's avenues, from La Villette to Montparnasse, from Bercy to La Défense, in their hundreds of thousands they surge the streets. Equipped with invisible, unobtrusive technology, each individual seamlessly becomes part of a vast pervasive and ambient interconnection network, which extends across hundreds of intermediate participants in every direction. Running at previously unimaginable end-to-end rates beyond the 1 Gbit/s, terabits of information are transferred in each second through the complement of devices in the network. While everyone is on the move, the dynamic network organizes itself to sustain continuous and uninterrupted operation, making each and everyone to become a part of "FutureNet 21" (Figure 1.1).

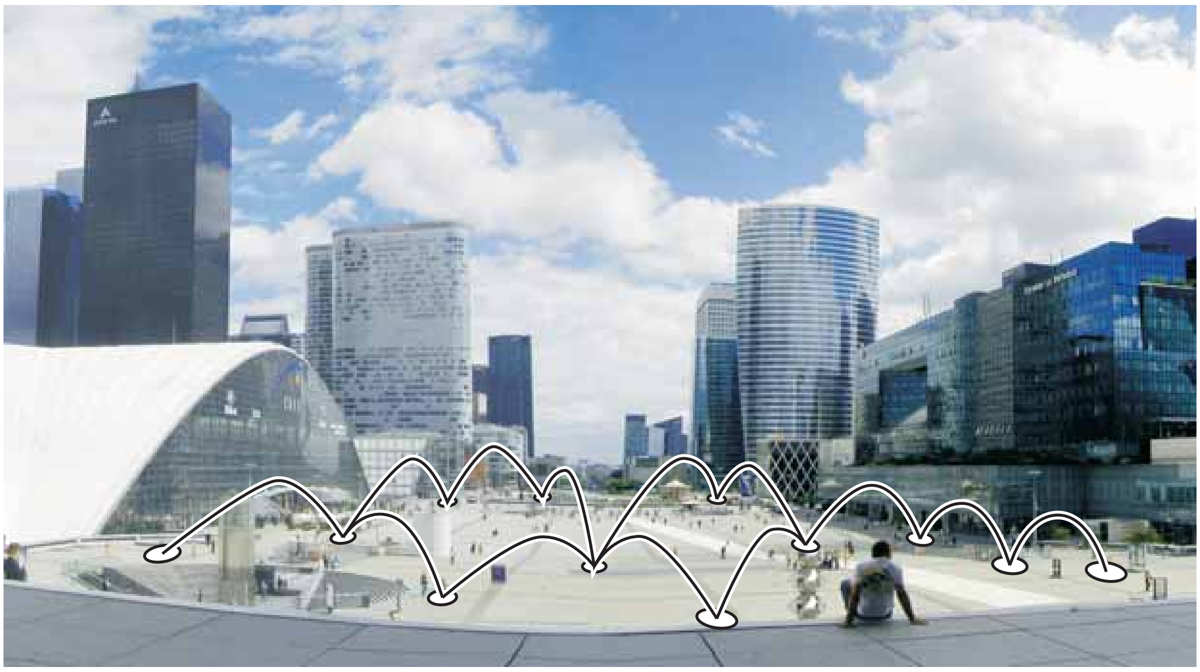


Figure 1.1: FutureNet 21: mobile ad-hoc network scenario in the year 2015.

Alice, who is new to the city, carries some of the most advanced mobile equipment on her, branched into FutureNet 21 in an autonomous way. Through her devices, she gains access to an abundance of services dispersed throughout the network, which continuously collect and process information across the city, adjusted to her personal profile and her whereabouts. Supported by sophisticated visual displays, she perceives her environment in an augmented way, delivering to her familiarity with the city from the very first moment.

Scenarios of this kind will become possible by the amalgamation of trends in computing and communication technology and the paradigms of mobile, ubiquitous, and context-aware computing. In contrast to current location-based services, which operate through the support of service and information infrastructures implemented in the fixed part of communication networks, the previous scenario foregoes the use of any infrastructure-based support and relies on the collaborative utilization of user devices only.

Such a radically different system structure would allow to exploit the vast storage and communication capacity available in wireless ad-hoc networks, in addition to existing infrastructure-based networks, for the purpose of realizing location-based services. Not only would core and radio access networks be relieved of large loads that inevitably come with the delivery of location-based services to a large number of users, but service and information coverage could, in general, be significantly extended by the ad-hoc communication paradigm.

However, a missing link in enabling large-scale location-based application scenarios are sophisticated mechanisms that operate at the basis of the information and service infrastructure, which are able to defy the harsh characteristics of infrastructureless networks. Only by providing a level of efficiency, robustness, and scalability that qualitatively matches the characteristics of fixed networks to the best possible extent, will it be possible to build accurate location-based services in a generic way also in infrastructureless networks.

This dissertation targets at the development of such a set of fundamental mechanisms that will enable scenarios as the one described in the first place. In this chapter, we provide the background for and develop the specific objectives addressed by this dissertation.

We first require to highlight in more detail the essential technological and parallel paradigmatic trends in Section 1.2, which enable, in principle, scenarios of the described kind. In Section 1.3, we embed our work into the Nexus Center of Excellence, in whose context this dissertation was pursued. We will ultimately frame the specific focus and contributions of this dissertation in Section 1.4, in relation to the objectives addressed by Nexus. In Section 1.5 we outline the structure that we will pursue in the chapters to follow.

1.2 Technological and Paradigmatic Trends

During the past three decades, some significant technological revolutions have occurred that act as enablers for the realization of scenarios based on the paradigms of mobile, ubiquitous, and context-aware computing. Some of the most influential leaps in technological advancements are depicted in Figure 1.2, annotated with key paradigms presented in parallel. The depiction allows us to identify three main branches, which become manifest in the realms of computing, communication, and sensing technology. The grey-shaded funnel indicates the degree of technological diversity and convergence, in which a growing number of individual computing and communication technologies are combined into single and highly capable mobile devices.

1.2.1 Computing

In the 1980s, computing technology has achieved an essential breakthrough and triggered the era of portable computing. Thanks to the general advances in computing and miniaturization, the first commercially available portable computer, Osborne 1, entered the market in 1981. This event preceded the revolution of mobile computing, pertaining to all aspects of computing performed by users on a mobile computing device. Putting the individual user with his ability to change location in the center, nomadic computing [IB93, Kle95] reflects this newly gained freedom of users (Figure 1.2).

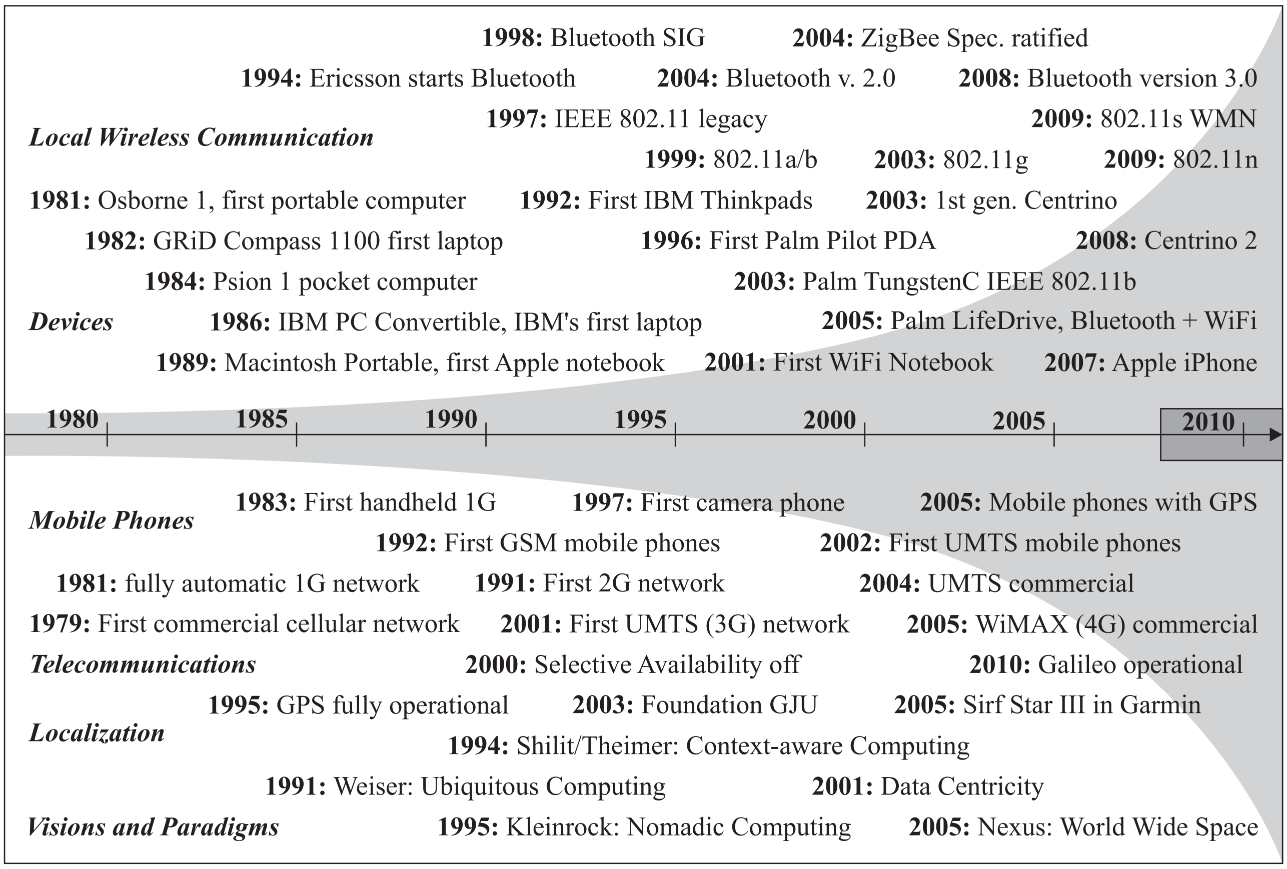

1998: Bluetooth SIG
2004: ZigBee Spec. ratified
1994: Ericsson starts Bluetooth
2004: Bluetooth v. 2.0
2008: Bluetooth version 3.0
1997: IEEE 802.11 legacy
2009: 802.11s WMN
Local Wireless Communication
1999: 802.11a/b
2003: 802.11g
2009: 802.11n
1981: Osborne 1, first portable computer
1992: First IBM Thinkpads
2003: 1st gen. Centrino
1982: GRiD Compass 1100 first laptop
1996: First Palm Pilot PDA
2008: Centrino 2
1984: Psion 1 pocket computer
2003: Palm TungstenC IEEE 802.11b
Devices
1986: IBM PC Convertible, IBM's first laptop
2005: Palm LifeDrive, Bluetooth + WiFi
1989: Macintosh Portable, first Apple notebook
2001: First WiFi Notebook
2007: Apple iPhone
1980
1985
1990
1995
2000
2005
2010
Mobile Phones
1983: First handheld 1G
1997: First camera phone
2005: Mobile phones with GPS
1992: First GSM mobile phones
2002: First UMTS mobile phones
1981: fully automatic 1G network
1991: First 2G network
2004: UMTS commercial
1979: First commercial cellular network
2001: First UMTS (3G) network
2005: WiMAX (4G) commercial
Telecommunications
2000: Selective Availability off
2010: Galileo operational
1995: GPS fully operational
2003: Foundation GJU
2005: Sirf Star III in Garmin
Localization
1994: Shilit/Theimer: Context-aware Computing
1991: Weiser: Ubiquitous Computing
2001: Data Centricity
Visions and Paradigms
1995: Kleinrock: Nomadic Computing
2005: Nexus: World Wide Space

Figure 1.2: Enabling technologies for mobile, ubiquitous, and context-aware computing.

The benefit of this new autonomy experienced by business and later private users, ignited the development of a sheer variety of mobile devices. Parallel to the ever increasing diversity of mobile devices besides the notebook, such as the first personal digital assistant (PDA) in 1984, massive enhancements occurred. More features were constantly added to single devices, such as high resolution displays and mobile hard disks.

Starting with the first kind of wireless technology integrated into notebooks in 2001 and the ongoing downsizing of devices, an important step from portability to true mobility was achieved.

Since then, by substantial advances in energy efficiency of mobile computation and communication technology, with edge-breaking technologies such as Intel's Centrino generations since 2003, an important ingredient to sustainable nomadic computing has become available.

1.2.2 Communication

With the progress in computing and device technology, the realm of communication technology has experienced in parallel a number of significant developments, which can be roughly classified into cellular networks and local wireless networks.

Developments in cellular network technology started in 1981, where the first fully automated first-generation (1G) network was set up, two years after the analogue cellular age had been introduced with the first commercial cellular network. Since then, rapid advances via second (2G) and third generation (3G) cellular technology have occurred. With the first commercial WiMAX infrastructure, operational in Germany since 2005, the 100 Mbit/s barrier was breached. Together with cellular networks and fuelled by miniaturization, appropriate devices were developed. While these devices were first used only for the purpose of telephoning, they have continuously integrated additional features, for instance, the first camera phone in 1997, which emphasizes one more time the trend towards technological convergence.

In the realm of local wireless networks, the first initiative started with Bluetooth in 1994, which was achieving at that time a maximum data transfer rate of 732 kbit/s. Bandwidth of wireless local area networks (WLAN) and related technologies have increased ever since. While IEEE 802.11 legacy, introduced in 1997, was able to achieve data rates of up to 2 Mbit/s, IEEE 802.11a/g, omnipresent in today's mobile devices, is able to communicate at up to 54 Mbit/s. IEEE 802.11n, projected for 2009, is designed for up to 600 Mbit/s, which shows impressively the trend towards the 1 Gbit/s limit that we have assumed in the described scenario.

A key feature supported by these technologies from the beginning is a special type of ad-hoc communication mode, which allows devices to communicate with one another independently of any infrastructure components. This mode, together with the general trends in technological miniaturization and computing, were essentially responsible for the advent of ubiquitous computing. This paradigm, foreseen by Mark Weiser [Wei91] in 1991, describes the scenario where information and communication devices are embedded into objects of everyday life [HHSW05], forming ad-hoc networks themselves. Pervasive computing, a frequently used synonym, emphasizes the hidden character of this computing paradigm.

1.2.3 Sensing

The third essential ingredient vital to the support of new computing paradigms are the still ongoing achievements in sensor technology. While the employment of sensors has a long tradition, progress in technological miniaturization was essential for the feasibility of embedding sensors into devices. This opened up a vast new field of possible applications, most of all, in wireless sensor networks and localization systems.

On one side, the advent of wireless sensor networks (WSNs) literally opened the window to the physical world for computing platforms. For the first time, highly distributed computation and communication infrastructures were able to gather and process information about the physical world for specific applications. On the other side, localization systems, such as the Global Positioning System (GPS) [EM99], operational since 1995, were able to deliver accurate position information to the user's end device. With the high-performance SiRFstar III chipset, first incorporated in Garmin handheld devices in 2005, highly accurate position acquisitions in the range of only a few meters became possible.

These developments allowed for a whole new class of applications in mobile communication networks, based on the paradigm of context-aware computing, a term presumably first coined in [ST94]. Also referred to as sentient computing, this paradigm incorporates information that is relevant for a user's current situation in order to adapt the services and information accordingly. The special case of location-based computing puts the position as a specific type of context in the center, enabling the delivery of services and information to the user in dependence of his current position in the physical world.

1.3 Background

1.3.1 Explicit Context Models

In a broader sense, context can be defined according to [RBB03, RDD$^+$03] and based on [DA99] as "the information which can be used to characterize the situation of an entity". In this definition, entities may refer to "persons, locations, or objects which are considered to be relevant for the behavior of an application."

When dealing with a large number of users with similar but also different needs regarding their personal context, it is of great value to provide structured context information in the form of explicit context models. Appropriately implemented, such models can provide each user with his particular subset of relevant information. Research related to all aspects of explicit context models can be positioned, based on the analysis provided in [HHSW05], in the intersections of mobile, ubiquitous, and context-aware computing according to Figure 1.3, with the backing enabling technologies discussed in the previous section.

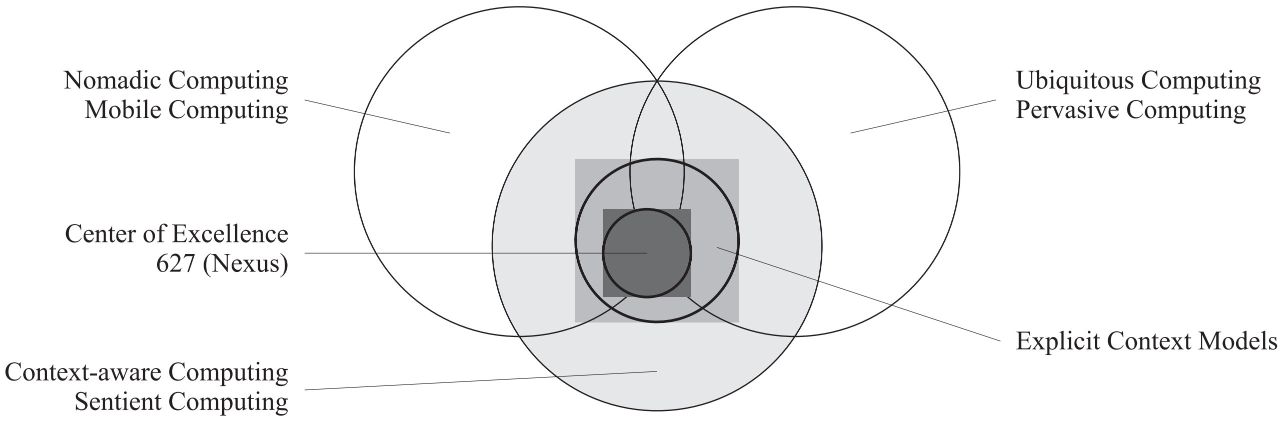


Figure 1.3: Computing paradigms, explicit context models, and the SFB 627 (Nexus).

Regarding the collection of large quantities of context information, an essential distinction between primary and secondary context according to [RBB03, RDD$^+$03] is useful. Primary context, shared by virtually all possible types of entities, encompasses identity, location, and time, which designate the most essential properties of an entity. Secondary context, comprising any other conceivable context information, e.g., a car's current speed and direction of movement, may model other relevant aspects of individual entities.

Besides the mere provisioning of context information in a generic form, the specific way of how context is used is a key indicator for selecting methods that provide context information efficiently. For example, if the location information of a user is not required distant from him, it makes no sense to set up a distributed index structure that materializes the information at distant storage locations, which would require constant maintenance with no benefit.

For our convenience, the authors of [RBB03, RDD$^+$03] have identified three basic ways in which applications make use of context information: context-based selection, presentation, and action. These functionalities provide essential hints to more basic required services. For instance, in order to select information relevant to a user's current location, that location needs to be continuously available. Another example is the presentation of information in the vicinity of a user's current location, which implies the usage of spatial queries that retrieve information such as a defined number of nearest objects of a specific type relative to the user's current location. Yet another example is to perform certain functions when defined spatial constellation between the user and other entities in the world occur (spatial events), such as the popping up of a message based on the user reaching a certain geographic region (e.g., a building).

1.3.2 SFB 627 (Nexus)

The provisioning of explicit context models for context-based applications has been addressed by a large number of efforts in the research community. In a nutshell, the majority of these attempts provide context models for a specific purpose only [BCQ$^+$07].

Dealing with a broad range of issues related to the provisioning of context models on a large scale has been addressed by the Center of Excellence 627 (Nexus)[1] [HKL$^+$99, GBH$^+$05, REF$^+$06]. The project deals with many objectives related to the modelling, acquisition, processing, and provisioning of context to context-aware applications. Figure 1.3 shows the location of the Center's research scope relative to mobile, ubiquitous, and context-aware computing.

Central to the Nexus project is the architecture of the Nexus platform, depicted in Figure 1.4 and based on an adaptation of [REF$^+$06]. It provides not only an extensive context model of the real world, but also the basic services to accomplish the context-based interactions selection, presentation, and action), which we have described in the previous section.

The Nexus platform architecture can be divided into three main tiers: the client, federation, and service tier [FV03]. Each server in the service tier stores the context information of a specific geometric region in the physical world. This is due to the fact that context information is frequently queried by referring to a geometric location, which emphasizes the importance of context models to location-based applications in particular.

[1]German: Sonderforschungsbereich (SFB) 627

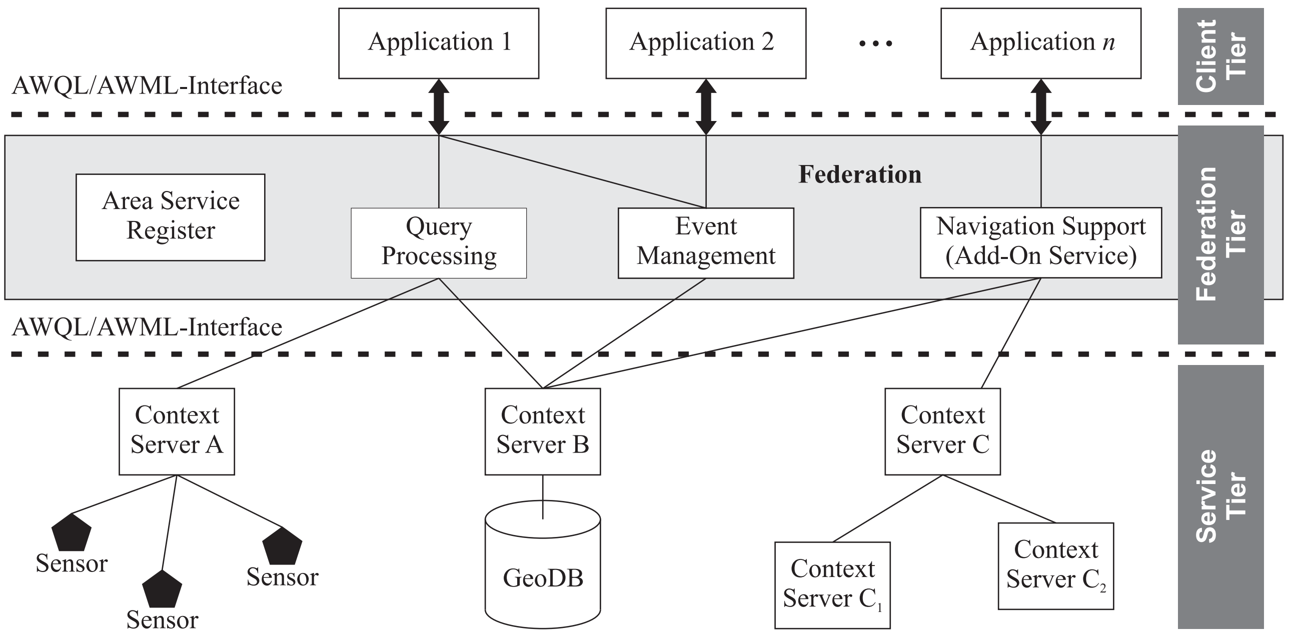


Figure 1.4: High-level architecture of the Nexus platform.

Each context server may store any type of context information provided by some context provider, such as dynamic sensor information (Context Server A) or geographical data (Context Server B). A context server may also abstract over an extensive geometric area that contains multiple context servers. Each of these servers may in turn be responsible for a specific subregion (Context Server C, combining context information from server C_1 and C_2).

Altogether, the Nexus vision describes the scenario where the collective of a large number of context servers stores a comprehensive global model of the physical world. This equates to an extensive digital representation of the real world, which is also referred to by the notion of World Wide Space [RJM$^+$06], in the style of the World Wide Web.

In order to provide a unified and consistent view on the combined context model, the federation tier provides all necessary mechanisms for merging individual model subsets stored at each context server. For each reference to a context model subset, the Area Service Register is consulted to locate the relevant context servers based on the geometric region that each server's context model captures. Consequently, the federation tier also provides a unified view for context-based applications with respect to essential base services for the previously discussed context-based interactions. Most importantly, these services encompass the processing of queries and the observation of real-world events based on the digital representation of the physical world.

1.4 Focus and Contributions

1.4.1 Focus

After the foregoing elaborations, we are now in the position to specify the focus and contributions of this dissertation. Regarding the demarcation of its scope, we can identify three major dimensions, namely, system structure, location model, and data characteristics, which we have depicted in Figure 1.5. Two additional properties, which we refer to as context quality and service functionality, will be discussed thereafter.

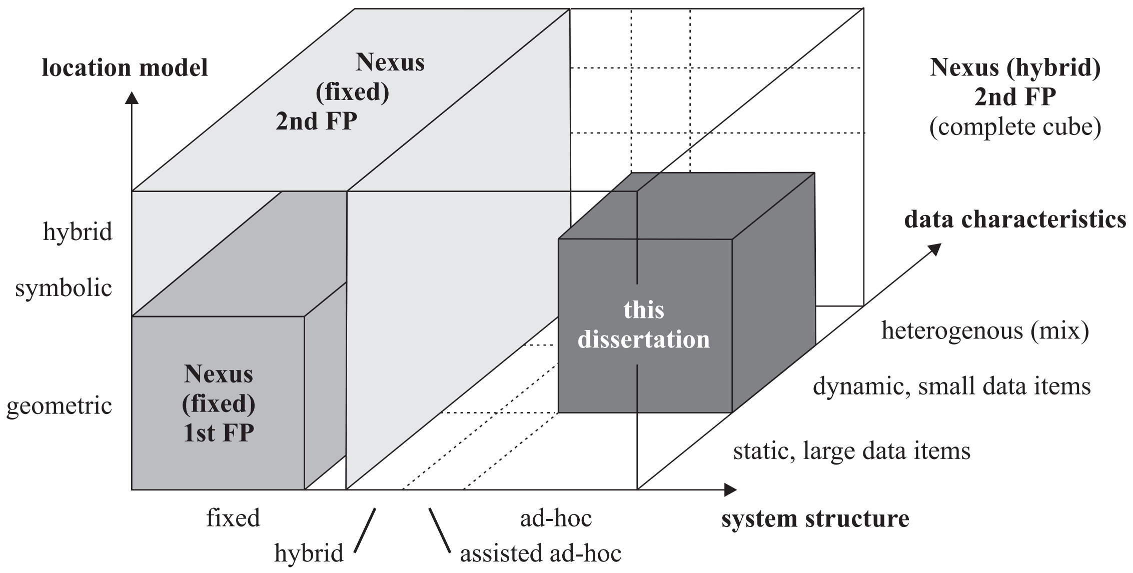


Figure 1.5: Design space: system structure, location model, and data characteristics.

The dimension of the *system structure* describes the structure of the underlying physical network assumed for the provisioning of context models and context-based services. Figure 1.5 indicates the relative location of this dissertation's scope with respect to the system structure addressed by Nexus in the first (1st FP, 2003-2006) and second funding period (2nd FP, 2007-2010).

Based on the discussions in Section 1.2.2, the trends in technology suggest a strong potential of wireless ad-hoc networks not only as a stand-alone, but also as a complementary system structure in conjunction with infrastructure-based networks. The likely increasing desire for users to "become more nomadic" according to mobile and nomadic computing paradigms (Section 1.2.1) suggests to migrate functionality related to context management closer to the user, that is, to the wireless ad-hoc network itself. In the system structure, this dissertation focuses for these reasons on context management in pure wireless ad-hoc networks. This will be the prerequisite for the future consideration of hybrid context management approaches in a hybrid system structure combining infrastructure-based and infrastructureless networks.

The dimension of the *location model* is of importance due to the primary context of location, which possesses a special role and leads to the important class of location-based services and the vision of a World Wide Space (Section 1.3.2). Apart from geometric location models, which are based, for instance, on geometric coordinates obtained from GPS (Section 1.2.3), we can identify symbolic location models and a combination of both into hybrid location models.

In general, the complexity of location models increases from geometric, via symbolic to hybrid location models, including more complex definitions and relations between individual locations in the physical world. To restrict the complexity in this dimension, this dissertation focuses on geometric location models only. This choice is sufficient for showing the principal techniques necessary for efficient context management in mobile ad-hoc networks.

The dimension of the *data characteristics* describes general properties of the data items to be managed. Apart from many different characteristics that can be consulted for classification purposes, the dynamics and the size of data items can be regarded as the most influencing ones. This is due to the fact that the product of dynamics and size directly correlates to an increase in the cost of keeping data items in the system up to date.

A key observation is that the dynamics and size of data items generally exhibit a strong correlation. For example, city maps, constituting large amounts of data, are mostly static, whereas the rather small items of location information are likely to be highly dynamic in a scenario of mobile users. Therefore it is valid and absolutely sufficient for our purposes to arrange both characteristics in a combined form in Figure 1.5.

From the discussion about the importance of the primary context of location for location-based applications in particular (Section 1.3.1), this dissertation focuses on dynamic and small data items. In addition, straightforward solutions are conceivable for rather static data items in mobile ad-hoc networks, e.g., their replication on a predefined subset of network nodes.

An additional fact that complicates the provisioning of model data and the implementation of functionality on top of it is that a model of the physical world can be captured in the system only within certain limits of accuracy. This is due to technological limitations in the acquisition of data pertaining to physical objects, as discussed in Section 1.2.3 for the case of GPS.

In general, the imperfections in any observation of the physical world can be subsumed by the term *data quality*, or context quality when referring to the management of context information. In this dissertation, we will address the issue of data quality by taking into consideration the inaccuracy of geometric locations of objects of the real world and its impact on the mechanisms for the management of dynamic location information.

Besides the provisioning of context models, it is important to demonstrate the applicability and practicability of the concepts developed in this dissertation. To this end, we make use of selected base service functionality that builds on the previously discussed items and which is essential in the support of the diverse types of context-based applications.

During our discussion on forms of interaction in Section 1.3.1 and the base service functionality provided by the Nexus platform (Figure 1.4), we can identify queries and events as two of the most fundamental services required for context-based and location-based services. In this dissertation, we will take up the processing of spatial queries, which are centered around the primary context of location and constitute a fundamental type of query especially for location-based services. We will briefly discuss events in the outlook Section 6.2.

1.4.2 Contributions

The objectives of this dissertation are the design, implementation, and evaluation of a framework for fundamental strategies and mechanisms for context management in mobile ad-hoc networks. The dissertation lays the foundation for enabling the development and deployment of location-based and context-based services in these networks.

The individual contributions within this framework follow from the derivation of the focus in the previous section and can be concisely stated as follows:

(1) We present a quantitative analysis of the topological characteristics of mobile ad-hoc networks by looking at the partitioning behavior of these networks;

(2) We derive a compact framework comprising the essential components for dynamic data storage in mobile ad-hoc networks and discuss in detail the related literature;

(3) We introduce a fundamental set of algorithms for the management of dynamic data, which is robust against the specific characteristics of mobile ad-hoc networks;

(4) We devise an analytical model for the assessment of storage costs in mobile ad-hoc networks and present a taxonomy and comparison of fundamental storage approaches;

(5) We define a formal accuracy model for the semantics of location information and spatial queries, the latter including geometric range and nearest neighbor queries;

(6) We introduce algorithms for the updating of dynamic and inaccurate location information and probabilistic spatial queries based on the defined formal accuracy model;

(7) We provide an extensive simulative performance analysis to validate the algorithms for fundamental storage and query processing in mobile ad-hoc networks.

1.5 Structure of the Dissertation

The main part of the dissertation is structured as follows. In Chapter 2 we introduce fundamental notions regarding mobile ad-hoc networks and data storage relevant for the scope of this dissertation. This chapter also presents the analysis of the partitioning behavior of mobile ad-hoc networks, and derives the principal structure of the storage framework. The chapter concludes with an extensive discussion of the related literature.

Chapter 3 introduces a resilient core approach to the storage of dynamic data items in mobile ad-hoc networks based on the paradigm of location-centric storage. This chapter presents the analytical comparison of our own storage approach with other fundamental approaches, and a simulative performance analysis of a set of key performance metrics.

In Chapter 4, we complement the core storage approach with comprehensive strategies and mechanisms of data migration to address the impact of node mobility on data storage. Additional simulative performance results compare our own migration approach with a number of other conceivable approaches and demonstrate its superiority over these approaches.

Chapter 5 defines the formal accuracy models for location information and spatial queries. This chapter also introduces the algorithms for location updating and query processing in mobile ad-hoc networks and provides a simulative analysis to point out the impact of imperfect location information on the performance of these algorithms.

The dissertation is concluded in Chapter 6 by summarizing key results and stating some important implications for future considerations. Finally, a repertory of promising research directions deduced from the work in this dissertation is highlighted to stimulate future efforts in the field of data and context management in mobile ad-hoc networks.

Chapter 2

Fundamentals

In this chapter we specify the fundamental constraints, assumptions, and requirements that underly the algorithms developed in the remainder of this dissertation. The key differentiation to other work is the model of mobile ad-hoc networks and its unique characteristics, which make the design of fundamental algorithms a highly challenging task and which we describe in detail in Section 2.1. To emphasize the importance of MANET characteristics, we present a detailed analysis of the partitioning behavior in these networks in Section 2.2.

In Section 2.3 we develop the notion of location-centric storage, the fundamental paradigm that follows from arguments based on the important role the primary context of location plays for context-based applications, and which underlies our algorithms. Based on the constraints and assumptions of Section 2.1 through 2.3, we derive the key requirements that the storage algorithms must take into account and derive our reference model for context management in MANETs based on these requirements in Section 2.4. Finally, we discuss related work that is relevant in the scope of our reference model in Section 2.5.

2.1 Mobile Ad-hoc Networks (MANETs)

2.1.1 Network Model

Mobile ad-hoc networks (MANETs, e.g. [LC04]) are a class of communication networks formed by portable devices that communicate with one another over wireless communication (radio) links. Devices may comprise any of the equipment described in Section 1.2 and Figure 1.2, such as mobile phones, personal digital assistants (PDAs), and notebooks.

Due to the limited radio range of wireless communication technologies, such as IEEE 802.11 and Bluetooth (Section 1.2.2), any device is able to communicate directly only with a subset of other devices at any time. Due to the ability of devices to move about freely while being carried by users, however, devices continuously discover new devices by moving into one another's communication range. Thus, communication links between mobile devices are established and severed in a spontaneous manner. Viewed across the whole network, mobility leads to a constantly changing topology of the physical network.

Another type of wireless multihop networks are wireless sensor networks (WSNs). In contrast to MANETs, the devices in WSNs possess significantly less capabilities in terms of battery lifetime, processing, storage, and communication. Typical WSNs furthermore exhibit much lower degrees of device mobility than MANETs. A further important differentiator is that WSNs are designed for specific applications based on the processing of sensor information gathered from their environment, and, in contrast to MANETs, they do generally not allow for more general services due to their limited capabilities.

A second type of wireless multihop networks are wireless mesh networks (WMNs). Different from MANETs, nodes of WMNs consist of stationary devices, forming a relatively stable "mesh" of nodes through wireless radio links. Furthermore, the nodes in WMNs are powered externally and have significantly more capabilities with respect to processing and storage. WSNs are sometimes also considered to be more general and to comprise not only wireless mesh routers, but also MANETs as a complementary network structure [AWW05]. In this view, MANETs can be regarded as a special case of WMNs.

In MANETs, in order to coordinate access to the shared medium when devices attempt to transmit at the same time, protocols for medium access control (MAC) that are specific to wireless media are employed. Such protocols are designed to minimize the number of collisions on the medium to avoid frequent retransmissions and congestions. Due to the limited communication range of devices, configurations such as the hidden and exposed terminal situation may occur where collision detection schemes cannot be applied. Therefore, protocols employ alternative methods, for instance, the CSMA/CA with RTS/CTS mechanism in IEEE 802.11, which builds on collision avoidance instead by negotiating which device is to transmit next. In this dissertation, we will assume the aforementioned protocol, without limiting the generality of our approaches to be developed in the remainder of this dissertation.

To function as a network, an essential property is that communication between two nodes generally occurs over multiple hops of intermediary nodes, since communication range strongly limits direct communication between nodes. While routing is also essential in infrastructure-based networks, for instance the Internet, the constantly changing network topology make routing one of the major challenges in MANETs. For this reason, a large number of routing protocols have been devised that attempt to establish efficient routes on-demand, for instance AODV, rather than pre-computing routes beforehand. While these protocols are able to significantly minimize the overhead for route establishment, route failures are still critical because they require costly route discoveries. For this reason, geometric routing protocols, such as GPSR, are preferred over topological routing protocols, like AODV, when the geometric position is known to network nodes. In this protocol class, routing is virtually stateless in the sense that explicit route maintenance and discovery are not required. Instead, routing is accomplished based on decisions that are taken from the viewpoint of a single node.

In this dissertation, we will assume that the nodes in MANETs are able to acquire their position information, for example, based on an embedded GPS receiver (Section 1.2.3). Therefore, geometric routing protocols may be employed for packet routing. We will investigate in Chapter 3 in more detail additional routing mechanisms required for our purposes.

Besides routing, the transport of messages that exceed the size of single packets is an important mechanism also in MANETs. While TCP's faulty assumptions of its congestion control mechanism in wireless networks is a well-known problem, additional properties further complicate message transport in MANETs. The dynamic network topology challenges the establishment of communication paths between nonadjacent nodes, because fixed routes cannot generally be maintained for a prolonged period of time. Furthermore, the length of a path over which a connection is established, and thus, the incurred latency, may vary significantly, which is not a critical issue in the Internet. Plus, on the shared medium adjacent nodes frequently request the medium at the same time, which is not the case in switched networks, and which has a significant impact on performance. We will revisit this problem in Chapter 4 in the treatment of data migration protocols, which move potentially large amounts of data between nodes.

In summary, MANETs adhere to the paradigm of peer-to-peer networking, according to which all devices function in an equal way with respect to essential tasks to be performed for the network to operate as a whole, including medium access control, packet routing, and message transport. In further chapters, we will assume that this peer-to-peer characteristic also holds for data storage and service provisioning.

2.1.2 Discussion of Network Characteristics

In this section we elaborate in more detail on the specific characteristics of mobile ad-hoc networks that make the design of efficient algorithms for data storage and services challenging. The purpose is to understand the significance of each of the discussed characteristics in relation to the goal to be achieved, and in order to prioritize the most critical ones. Table 2.1 shows a summary of each of the discussed characteristics, in order of increasing importance.

Computing capabilities (CPU)	less critical
Memory capacity (RAM)	less critical
Energy resources (battery)	critical
Link bandwidth and quality	critical
Node mobility	very critical
Network density and topology	very critical

Table 2.1: Characteristics of mobile ad-hoc networks.

Computing capabilities are required by each node to perform operations related to data storage and services. For instance, data items are stored in an index structure that needs to be maintained, and the calculation of query results requires to perform computations based on inaccurate position information, which involve mathematical integration methods. In general, the computational resources required for the algorithms developed in this dissertation are outweighed by much more complex processes that occur in the context-based applications themselves, such as the dynamic displaying of geographic maps, which is already well supported by today's devices. Furthermore, the tasks any node must execute for the purpose of data storage and query processing will change with a node moving through the network. Thus, in the long term, processing is shared by the combined computational resources of all nodes in the network in a uniform way. We therefore rate computing capabilities as less critical, according to Table 2.1, while we take general algorithmic complexity considerations into account.

While devices in MANETs are generally less powerful than those in infrastructure-based networks and the previously discussed wireless mesh networks, even the smallest state-of-the-art mobile device according to Figure 1.2 today is equipped with several GB of memory. An additional constraint is our focus on small data items in the dimension of data characteristics (Figure 1.5), which allows to store a potentially large quantity of data items on a single node. We will further discover in Chapter 4 that other performance issues related to the managed amount of data will become relevant before do memory constraints. Therefore, the memory capacity can be classified as less critical, but will nevertheless be addressed in the form of load-balancing concepts if this is required in some cases.

In MANETs, more critical than the previous characteristics are the limited energy resources due to the utilization of only batteries in mobile devices. Two main processes significantly accelerate battery consumption. First, transmission over the wireless interface is extremely power-consuming. The authors of [BA06] state that the "energy required for transmission of a single bit has been measured to be over 1000 times greater than a single 32-bit computation". Secondly, each position fix performed by an embedded receiver for positioning technology also consumes significant amounts of energy [DLR07]. On the opposite site, two properties reduce the significance of energy constraints. First, the time a device participates in typical MANET scenarios is strongly limited due to the involvement of users carrying the devices, and batteries may be recharged in the meantime. Second, technological trends as stated in Section 1.2.1 indicate that energy constraints will become a less significant issue in the long run.

Having identified energy resources as an important influencing parameter, its impact on the network is more relevant. The effects of a battery depletion while a device is active in a network is either a controlled device shutdown or a device failure. In the case of a controlled shutdown, a device is able to take the necessary measures to yield its role in data management to other nodes. Therefore, such shutdowns are only critical when a large fraction of devices are detached from the network, which then adversely impacts overall operation of the MANET in the first place. Device failures, however, are more critical because each device in the network may be in a state that is relevant for data management. In this case, only proactive measures, like data replication, are a remedy. In this dissertation, we assume that device failures are rare events, which is consistent with the observation of today's mobile devices that are in service. Nevertheless, we will point out in Chapter 6 the importance and role of data replication when node failures are also to be taken into account.

Based on these discussions, we rank energy resources as critical in Table 2.1. In all algorithms, we will therefore put strong emphasis on general communication efficiency. Research with specific emphasis on energy-efficient position acquisition in the tracking of mobile objects is e.g. [DLR07], which is complementary to this dissertation.

At the current technological state-of-the-art in communication technology, link bandwidth and quality are still perceived as critical parameters. While trends in communication technology indicate a significant improvement in the case of link bandwidth (Section 1.2.2) in the near future, it is still necessary, due to the shared medium in MANETs, to avoid frequent access to the medium when possible. This may otherwise lead to undesirable congestions and as a consequence, to congestion-related packet loss. According to the data characteristics considered in Section 1.4.1, we focus on dynamic data items that are also small. The magnitude of wireless transmissions required to keep an accurate context model with these characteristics in the

MANET can therefore be considered, roughly speaking, somewhere between small and static data items on one end, and large and dynamic data items on the other end. We therefore consider this issue as critical and emphasize, similar to the case of energy resources, on a general consideration of communication efficiency in the design of our algorithms.

Regarding link quality, a major observation in wireless networks is its degradation with the distance between communicating peers. This may lead to additional mobility-related packet loss, where devices attempt to communicate with each other while no longer within transmission range. Packet loss, undesired in general, may then lead to an increased number of retransmissions that may in turn lead to congestions again. Therefore, we also consider link quality to be a critical characteristic, and we will address mobility-related packet loss that is due to distance-related link degradation explicitly in Chapter 4 in the context of data migration.

The most relevant property to be considered in MANETs is mobility, the primary differentiator from infrastructure-based networks, WMNs and WSNs. The impact of mobility on algorithm performance is twofold: immediate impact of mobility itself, and the impact of mobility on secondary characteristics, namely, node density and network topology. The immediate impact of mobility is on the performance of data access. This is due to the fact that data, which is stored at nodes, is transported by these nodes while they are carried along by users. This effect in turn leads to a significant dispersion of data items throughout the network and makes the access to subsets of context information inefficient due to extensive searches required. With increasing mobility, this effect is amplified, which is the reason for our indicating the impact of mobility as highly critical in Table 2.1. Chapter 4 is dedicated exclusively to the design of adequate countermeasures against the translocation of data.

Node density, in its static form, becomes critical when it assumes extreme values. This is in fact true for both large as well as low node densities. For large node densities, broadcast storms may occur [NTCS99], and even routing may be adversely impacted as we will show in Section 4.6. Much more critical, however, are small node densities, where the topology of the network is strongly impacted. The additional impact of node mobility leads to dynamics in node density and network topology. While on the small timescale, density and topology are only affected in the vicinity of individual nodes, large timescales may lead to globally significant changes in these parameters. In particular, heterogeneous node distributions may lead to both densely and sparsely populated regions in different parts of a single network.

Altogether, the impact of mobility and the complex correlations between density and topology must be considered as highly critical (Table 2.1). The impact of these parameters will be addressed by the large part of the contributions in Chapter 3 and 4.

2.2 Partitioning in Mobile Ad-hoc Networks

In the previous section we have classified node density as a highly critical MANET property, because low values lead to weak network connectivity. With node mobility, communication links between pairs of devices may be established and lost, yet overall connectivity of the network will still be maintained for sufficiently dense node population. With further decrease of node density, however, parts of the network separate into disjoint networks that are no longer connected, each part forming a *network partition* of its own.

Network partitioning in MANETs is especially critical because algorithms running in the network are not able to access data in more than one partition at a time. For instance, in the context of this dissertation, data that is acquired by a network node in one partition cannot be updated to a potential storage node in another network partition. On the other side, mobility acts in a beneficial way because it induces constant changes in network connectivity, and in turn, in the partitioning situation. Effectively, this means that disconnected network partitions will eventually join again after a finite amount of time.

An understanding of the characteristics of network partitioning is therefore crucial to deduce constraints that are relevant for the algorithm design. For that purpose, we present the following quantitative analysis of network partitioning, which addresses Contribution 1 in Section 1.4.2 and which is published in [HDM+05, HDM+07]. For the proposed set of partition metrics, each quantifying a specific aspect of the partitioning situation in MANETs, we underpin their relevance with example algorithms that are related to mobile data management in MANETs. Furthermore, we will consider conclusions we draw from the results in this section in the fundamental storage algorithms in Chapter 3 and 4.

In Section 2.2.1 we review existing work related to modelling network partitioning in MANETs. Section 2.2.2 defines the simulation model, and Section 2.2.3 basic notions used in subsequent sections. Section 2.2.4 provides a taxonomy of network partition metrics and introduces formal definitions of the set of partition metrics that we use to characterize different types of MANET scenarios. A comprehensive set of simulation experiments is discussed in Section 2.2.5, before we conclude the quantitative analysis of network partitioning in Section 2.2.6.

2.2.1 Related Work

Related work with respect to metrics that target at the quantification of the impact of density and mobility on connectivity in MANETs can be classified into the analysis of basic link properties and the analysis of higher-level properties beyond single links.

The author of [Bet02] provides an analytical model for minimum node degree and graph connectivity. The analysis of k-connectivity shows a relationship between the transmission range of nodes and the probability of the graph being k-connected for different k. The analytical model has been compared with simulation results. However, the results do not allow an interpretation concerning the number and size of partitions. In [Bet03] the author analyzes the connectivity of an ad-hoc network regarding the transmission range and number of nodes for the random waypoint mobility model. However, the results do not allow an interpretation of partitioning characteristics in mobile ad hoc networks. The authors of [GdWFM02] only consider the duration of links between adjacent nodes. All of these analytical studies provide a detailed insight into the nature of mobility, node distribution, and the properties of individual links in mobile networks. However, the cause for partitions in general is due to temporal aggregations of link failures between different pairs of nodes and requires a more abstract set of parameters.

Some previous work has been conducted that examines network properties that go beyond the state of individual links. The authors of [NTCS99] consider MANETs where nodes are distributed with high density. They concentrate on the fact that message flooding leads to an increasing number of collisions in such dense networks (broadcast storms) and propose optimizations to effectively reduce flooding overhead. In contrast, we focus on the impact of low node density on network connectivity. The authors of [HS01] introduce an algorithm to overcome partitioning in MANETs by increasing the transmission power of the wireless interfaces in the presence of network partitions. Their evaluation only states whether partitions were present at all in the given scenarios. In our work we went further by examining, among others, the number of partitions over time and the frequency of changes in partitioning.

2.2.2 Simulation Model

We assume a mobile ad-hoc network comprising a set of n mobile nodes. Each node occupies a position (x, y) inside of a fixed geographic area, denoted by A. The transmission properties of all nodes are based on the unit disc model, in accordance with the free space radio propagation model. Thus, two nodes n_i, n_j are within transmission range r_{tx}, if the Euclidean distance $d(n_i, n_j)$ between n_i and n_j is less than or equal to r_{tx}. The *topology graph* $G(N, E)$ consists of a set of vertices, denoted by N, representing the nodes of the network, and the set E of undirected edges, corresponding to communication links between nodes. An undirected edge $\{n_i, n_j\} \in E$ exists, if and only if $d(n_i, n_j) < r_{\text{tx}}$. A *network partition* is a subset $P \subseteq N$ where i) a path exists between all pairs of vertices $n_i, n_j \in P$, and ii) no path exists between any pair of vertices $n_i \in P, n_k \in N \setminus P$. Finally, by $\text{PART}(G)$ we denote the set of partitions in G.

2.2.3 Preliminary Notations

Let $T = (t_{\min}, t_{\max})$ denote a *physical* time interval. The topology graphs at $t_{\min}$ and $t_{\max}$ are defined to be $G_{\min}$ and $G_{\max}$, respectively. A *partition event* e occurring at a discrete point in time $t \in T$, is defined by a tuple $e = (\mathit{type}, t, P_1, P_2, G, G')$. By $e.\mathit{type}$, $e.t$, $e.P_1$, $e.P_2$, $e.G$, and $e.G'$ we refer to each of tuple e's element. The *type* attribute is either *split* or *join*, indicating that partitions P_1 and P_2 are split or joined, respectively. G and G' are the topology graphs before and after the occurrence of the event e. For a join event, both partitions P_1 and P_2 are contained in set $\text{PART}(G)$ and $P_1 \cup P_2$ is in G'. For split events the opposite holds.

The effect of events with equal timestamps on the topology graph is commutative, i.e., it leads to the same topology graph, independent from the order in which these events are applied. Informally, removing an edge $l_1 \in E$ and adding another edge $l_2 \in E$ for a given topology graph G at the same time will result in the same topology graph G' independent of the order of these modifications. Thus, we can arrange all events linearly according to a total order $<_o$ based on timestamps and a specific (arbitrary) order for events that occur at the same time. By $E_{\text{part}}(T) = \{e \mid e.t \in T\}$ we denote the set of partition events in T. The indexing function $\epsilon : \{1, \ldots, |E_{\text{part}}(T)|\} \to E_{\text{part}}(T)$ is the bijection that preserves the total order on $E_{\text{part}}(T)$, i.e. $\forall i, j \in \{1, \ldots, |E_{\text{part}}(T)|\} : i < j \leftrightarrow \epsilon(i) <_o \epsilon(j)$.

The two types of events indicated by the *type* attribute are defined as follows: A *partition join event* $e = (\text{join}, t, P_1, P_2, G, G')$, or join event for short, is a partition event in E_{part} transforming a topology graph G into G' such that $P_1, P_2 \in \text{PART}(G)$ and $\exists P \in \text{PART}(G')$ for which $P = P_1 \cup P_2$. A *partition split event* $e = (\text{split}, t, P_1, P_2, G, G')$, or split event for short, is a partition event transforming a topology graph G into G' such that $P_1, P_2 \in \text{PART}(G')$ and $\exists P \in \text{PART}(G)$ for which $P = P_1 \cup P_2$. A summary of the notations is given in Table 2.2.

Notation	**Definition**	**Section**
n	number of nodes in the network	2.2.2
r_{tx}	transmission range of nodes	
$G(N, E)$	topology graph with node set N and edge set E	
$P \subseteq N$	network partition	
$\text{PART}(G)$	set of partitions in G	
$T = (t_{\min}, t_{\max})$	physical time interval	2.2.3
$G_{\min}, G_{\max}$	topology graph at $t_{\min}, t_{\max}$	
$e = (\mathit{type}, t, P_1, P_2, G, G')$	partition event of type $\mathit{type} \in \{\text{split}, \text{join}\}$	
$E_{\text{part}}(T)$	set of partition events in physical time interval T	
$\Pi(e)$	set of partitions for event e	2.2.4.2
$\|\Pi(e)\|$	number of partitions for event e	
$\|\Pi\|_{\text{avg}}(T)$	average number of partitions in T	
$S_{\text{avg}}(e)$	average size of partitions for e	
$S_{\text{avg}}(T)$	average size of partitions over T	
$H(x, T)$	number of partitions with size x in T	
$R_{\text{part}}(T)$	average partition change rate in T	
$\text{PSR}(e)$	partition size ratio for e	
$\text{PSR}_{\text{avg}}(T)$	average partition size ratio in T	
$R_{\text{n}}(n_k, T)$	node partition change rate for n_k in T	2.2.4.3
$R_{\text{navg}}(T)$	average node partition change rate in T	
$\tau_{\text{sep}}(n_1, n_2, T)$	separation time between n_1 and n_2 in T	
$\tau_{\text{con}}(n_1, n_2, T)$	connection time between n_1 and n_2 in T	
$K_{\text{sep}}(n_1, n_2, T)$	number of separations between n_1 and n_2 in T	
$K_{\text{con}}(n_1, n_2, T)$	number of connections between n_1 and n_2 in T	
$N_{\text{cont}}(n_k, T)$	size of continuous node visibility set for n_k in T	
$N_{\text{acc}}(n_k, T)$	size of accumulative node visibility set for n_k in T	

Table 2.2: Network partitioning: notations.

2.2.4 Definition of Partition Metrics

2.2.4.1 Taxonomy

We consider two classes of metrics: *network-wide* partition metrics and *node-centric* partition metrics, shown in Figure 2.1. Network-wide partition metrics characterize the partitioning situation in a MANET viewed as a single entity. In this class, we define the following metrics: number of partitions, size of partitions, partition change rate, and partition size ratio.

Node-centric partition metrics characterize the partitioning behavior in the network from the perspective of single nodes. This class of metrics comprises node partition change rate, the separation and connection time and number, and two visibility sets, which describe the continuous and accumulative sets of nodes visible to a single node during the formation of partitions.

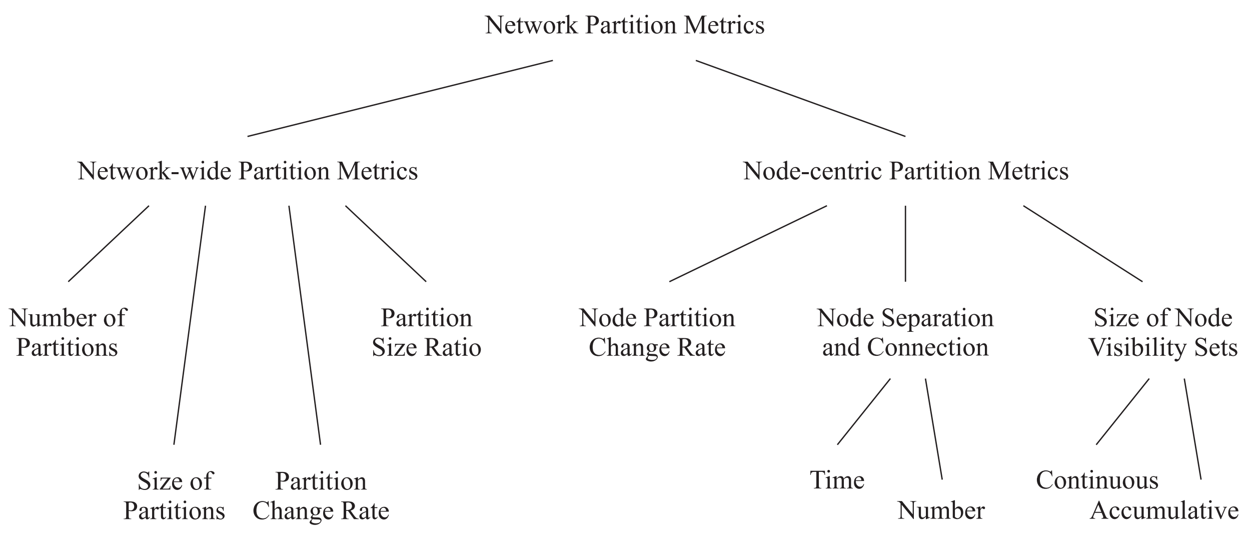

Network Partition Metrics
Network-wide Partition Metrics
Node-centric Partition Metrics
Number of Partitions
Size of Partitions
Partition Change Rate
Partition Size Ratio
Node Partition Change Rate
Node Separation and Connection
Size of Node Visibility Sets
Time
Number
Continuous
Accumulative

Figure 2.1: Taxonomy of network partition metrics in mobile ad-hoc networks.

2.2.4.2 Network-wide Partition Metrics

Number of Partitions

Let $\Pi(e) = \mathrm{PART}(e.G')$ be the set of partitions that exist after a given event e has occurred. If the *number of partitions* that exist in the network after the occurrence of event e is denoted as $|\Pi(e)|$, then the average number of partitions over the time interval T is defined as follows:

$$|\Pi|_{\mathrm{avg}}(T) = \tfrac{1}{t_{\max}-t_{\min}} \sum_{i=1}^{|E_{\mathrm{part}}(T)|-1} (\epsilon(i{+}1).t - \epsilon(i).t) \cdot |\Pi(\epsilon(i))| \qquad (2.1)$$

In other words, the average number of partitions is the time-weighted average of all partitions that exist in the system after each (split or join) event. For ease of exposition, we have not explicitly stated the number of partitions before the first event and after the last event in the above equation, although we have considered these corner cases in the simulation study. For the number of partitions, the optimum case is defined as $|\Pi|_{\mathrm{avg}}(T) = 1$, that is, all nodes are contained in a single partition and are connected at all times. The worst case occurs if $|\Pi|_{\mathrm{avg}}(T) = n$, that is, every node $n_k \in N$ is isolated.

Given an ad hoc networking scenario, the average number of partitions $|\Pi|_{\mathrm{avg}}(T)$ is the primary metric that can be used to identify how many partitions can be expected. Many distributed algorithms, e.g. data replication and distributed data aggregation, rely on communication primitives such as broadcasting or multicasting, which operate best in the presence of high connectivity within the network. For example, in the case of data replication, the average number of partitions is an important indicator for the number of replicas that should be created in a particular network scenario. Obviously, if the number of replicas is much smaller than the average number of partitions, there is a high probability that some nodes will not be able to access any replica in the system. On the other hand, if the number of replicas is greater than the average number of partitions, the probability of reaching at least one replica in a partition is much higher. If the number of replicas is much larger than the number of partitions, too many redundant replicas might be available that may not be required.

As an extension to (2.1), the distribution of the number of partitions gives additional information which can be of use, for example, for algorithms that employ network-wide data dissemination. If the distribution of $\Pi(e)$ indicates that the network is never fully connected, that is, $\forall e \in E_{\text{part}}(T) : |\Pi|(e) > 1$, a complete dissemination will never be possible without buffering messages temporarily at intermediate nodes.

Size of Partitions

Let $|P|$ denote the *size of a partition* P, that is, the number of nodes in that partition. The average size of partitions, written $S_{\text{avg}}(e)$, that exist in the system after a given event e can be derived from the average number of partitions by computing $n/|\Pi(e)|$. Thus, the average size of partitions over T is

$$S_{\text{avg}}(T) = \frac{n}{|\Pi|_{\text{avg}}(T)} \tag{2.2}$$

For given size x, the number of partitions with size x over the interval T is calculated as:

$$\begin{aligned} H(x,T) = {} & |\{P \in \text{PART}(G_{\min}) \;\mid\; |P| = x\}| \\ & + |\{e \mid e.\mathit{type} = \text{join} \wedge |e.P_1 \cup e.P_2| = x\}| \\ & + |\{e \mid e.\mathit{type} = \text{split} \wedge |e.P_1| = x\}| \\ & + |\{e \mid e.\mathit{type} = \text{split} \wedge |e.P_2| = x\}| \end{aligned} \tag{2.3}$$

where $e \in E_{\text{part}}(T)$. The domain of x is $\{1, \ldots, n\}$, because the minimum and maximum size of a partition is 1 and n, respectively. The graph of $H(x,T)$ is the frequency distribution of partition sizes within time interval T. With $H(x,T)$ it is possible to identify characteristic distributions of partition sizes. For example, peaks in the graph defined by H indicate that many partitions are nearly equally sized, e.g. many small and few large partitions.

Concerning the impact on data management algorithms, a distribution of partition sizes that reveals many large partitions implies high connectivity of the network. This enables a single node to reach many nodes with a high probability, e.g., during a message broadcast. If, however, many small partitions exist, many nodes are isolated and cannot be reached. For the particular case of broadcast algorithms, this metric can be used to determine whether simple flooding or gossiping mechanisms [NTCS99] are sufficient, or if algorithms, which are optimized for settings where many partitions exist, are necessary, for instance, Adaptive Flooding [OT98].

Partition Change Rate

The *average partition change rate* $R_{\text{part}}(T)$ is defined as the number of partition events that have occurred over time interval T, divided by $t_{\max} - t_{\min}$:

$$R_{\text{part}}(T) = \frac{|E_{\text{part}}(T)|}{t_{\max} - t_{\min}} \tag{2.4}$$

The partition change rate is an indicator for the frequency of partition changes in general. On one hand, high partition change rates are beneficial for data dissemination algorithms in order to deliver data to nodes in different partitions. The higher the rate, the more contacts between different partitions occur per unit of time. On the other hand, low partition change rates are beneficial for algorithms that require a relatively stable topological structure, for instance, aggregation trees. Here, any partition split event may damage the tree if the structure extends over multiple partitions after a split. These two examples show that the interpretation of the partition change rate strongly depends on the concrete application scenario.

Partition Size Ratio

Whenever two partitions P_1 and P_2 are joined or a partition is split in two, in the case of a join or split event e, respectively, we consider the ratio between the size of these two sets of nodes. This ratio is called the *partition size ratio* and is defined as:

$$\mathrm{PSR}(e) = \frac{\max(|e.P_1|, |e.P_2|)}{|e.P_1 \cup e.P_2|} \tag{2.5}$$

for a partition event $e \in E_{\text{part}}(T)$. The average partition size ratio over all events in T is

$$\mathrm{PSR}_{\text{avg}}(T) = \frac{1}{|E_{\text{part}}(T)|} \sum_{e \in E_{\text{part}}(T)} \mathrm{PSR}(e) \tag{2.6}$$

$\mathrm{PSR}(e)$ maps to values in the interval $[0.5, 1[$. If $\mathrm{PSR}(e)$ is 0.5, two equally sized partitions are joined into one partition for a join event, or one partition is split into two equally sized partitions for a split event. Values close to 0.5 indicate that the involved partitions are approximately equally sized. For values close to 1, the sizes of the partitions involved in the respective partition event are more and more unbalanced.

An example for a potential application of this metric is the situation where a node wants to detect if it has experienced a partition change. For that, we assume that $\mathrm{PSR}_{\text{avg}}(T)$ is close to 1 for many partition events and that a node is able to observe its k-hop neighborhood. Depending on the selection of k and the frequency of occurrence of strongly unbalanced partition size ratios, a node can determine with a certain probability whether it has moved from a very small to a very large partition, or vice versa.

2.2.4.3 Node-centric Partition Metrics

Node Partition Change Rate

For this metric, we first determine the number of partition events that a node n_k experiences. For a partition event e to be counted, it must hold that $n_k \in e.P_1 \cup e.P_2$, where $e.P_1$ and $e.P_2$ are the partitions involved in the event e. We define the set of partition events that a node n_k is involved in over time interval T by $E_{\text{part}}(n_k, T) = \{e \in E_{\text{part}}(T) \mid n_k \in e.P_1 \cup e.P_2\}$. We define the *node partition change rate* for a node n_k as follows:

$$R_{\text{n}}(n_k, T) = \frac{|E_{\text{part}}(n_k, T)|}{t_{\max} - t_{\min}} \tag{2.7}$$

For every node n_k, it holds that $R_{\mathrm{n}}(n_k, T) \leq R_{\mathrm{part}}(T)$, because $E_{\mathrm{part}}(n_k, T) \subseteq E_{\mathrm{part}}(T)$. The average node partition change rate for interval T is

$$R_{\mathrm{navg}}(T) = \frac{1}{n} \sum_{k=1}^{n} R_{\mathrm{n}}(n_k, T) \tag{2.8}$$

and will be used in comparison to the network-wide partition change rate $R_{\mathrm{part}}(T)$.

Patterns in the distribution of node partition change rates provide helpful insights into the characteristics of algorithms operating in a given geometric area. For example, if the network-wide partition change rate $R_{\mathrm{part}}(T)$ is high, the node partition change rate is able to reveal, for example, if a few nodes being involved in frequent partition changes dominate the global rate, while other nodes may experience only a few partition changes. Especially the latter case would indicate stable, but isolated partitions.

Node Separation and Connection

In the presence of network partitions, nodes being located in different partitions become pairwise separated. Due to the evolution of the partitioning situation over time, nodes also become connected again when being located in the same partition. In order to describe the impact of partitioning on node separation and connection characteristics, we define metrics related to both the time and number of separations and connections.

The *node separation time*, denoted by $\tau_{\mathrm{sep}}(n_1, n_2, T)$ and defined between pairs of nodes n_1, n_2 during time interval T, describes the time for which nodes are located in different partitions and thus, cannot communicate with each other. To derive this node-centric metric, two events e_1, e_2 have to be found for which the following is true: $e_1.\mathit{type} =$ split and $e_2.\mathit{type} =$ join and $n_1 \in e_1.P_1$ and $n_2 \in e_1.P_2$ (w.l.g.) and the next join event $e_2 \in E_{\mathrm{part}}(T)$ for which $n_1, n_2 \in (e_2.P_1 \cup e_2.P_2)$. The set of all such pairs of events is denoted as $E_{\mathrm{sep}}(T)$. The sum of all node separation times during interval T is defined as follows:

$$\tau_{\mathrm{sep}}(n_1, n_2, T) = \sum_{(e_1, e_2) \in E_{\mathrm{sep}}(T)} e_2.t - e_1.t \tag{2.9}$$

Note that no further join event might exist after the last split of the node pair and before t_{max}. In this case, we assume that the node separation time extends up to t_{max}.

The node separation time is a fundamental metric that allows algorithms to make assumptions about the expected time other nodes may be out of the scope of communication of a particular node. Notice that for this metric, we assume that nodes that are able to communicate via multiple hops are not considered separated. If the node separation time is small, an algorithm might simply tolerate node separations by applying simple timeout strategies. However, in the case of large node separation times, it may be required for algorithms to take explicit steps before a separation occurs, e.g., the explicit exchange of larger amounts of data.

The *node connection time* is the inverse of the separation time and describes to which extents pairs of nodes are located in the same network partition. The sum of all connection times is thus defined similarly to the sum of node separation times as the time difference between two events e_1, e_2 for which the following holds: $e_1.\mathit{type} =$ join and $e_2.\mathit{type} =$ split and $n_1, n_2 \in (e_1.P_1 \cup e_1.P_2)$

and the next split event $e_2 \in E_{\text{part}}$ for which $n_1 \in e_2.P_1$ and $n_2 \in e_2.P_2$ (w.l.g.). The sum of all connection times of n_1 and n_2 over T is denoted as $\tau_{\text{con}}(n_1, n_2, T)$.

The node connection time is crucial because it determines how much data can potentially be exchanged between nodes. For example, if the network characteristics are such that sufficiently large connection times never exist, an algorithm may explicitly perform remote operations in multiple steps in the first place.

While the node separation and connection time give a picture of the aggregated time that can be expected for nodes to be separated or connected, it does not specify the frequency of separations and connections that occur. In other words, each of the separation and connection time do not state whether the expressed sum is due to a small number of large intervals, or, vice versa, to a large number of small intervals. Therefore, we also examine the *number of node separations* $K_{\text{sep}}(n_1, n_2, T)$ and *connections* $K_{\text{con}}(n_1, n_2, T)$ per node pairs, which define the number of separations and connections that occur during time interval T, respectively.

The relevance of this further specification of node separation and connection behavior is shown by the following example. Consider the transmission of a large message between two nodes currently located in the same partition, and for which it is known that both nodes are connected in the same partition during 90 % of the time. However, if the number of connections indicate high frequency in the separation and reconnection of the two nodes, it is not likely that a transmission between both nodes can be maintained for a prolonged period of time. This is due a node separation that may follow shortly after a reconnection, and which is due to a recurring partition split. Such situations may require additional precautions to optimize transmission protocols, depending on the magnitude of node separation and connection frequencies.

Node Visibility Sets

We define two metrics related to the set of nodes visible to a specific node n_k during the splitting and joining of partitions: the continuous and accumulative node visibility set.

The *continuous node visibility set* $E_{\text{cont}}(n_k, T) \subseteq E_{\text{part}}(T)$ is the subset of n_k's split events within the time interval T defined as $E_{\text{cont}}(n_k, T) = \{e \in E_{\text{part}}(T) \mid n_k \in e.P_1 \wedge e.\mathit{type} = \text{split} \wedge e.t \in T\}$.[1] The size of this set is defined as

$$N_{\text{cont}}(n_k, T) = \Big| \bigcap_{e \in E_{\text{cont}}(n_k, T)} e.P_1 \Big| \tag{2.10}$$

$N_{\text{cont}}(n_k, T)$ gives information about how many nodes are located in the same partition as n_k during the time interval T. The gradient of $N_{\text{cont}}(n_k, T)$ based on a variation of T is an indicator of how fast the individual nodes in n_k's partition are separated into other partitions.

Data management algorithms that fragment information across multiple nodes are only suitable for systems with a reasonably small gradient, because only a slowly decreasing continuous node visibility set ensures a sufficiently long availability of information stored on remote nodes. The value of the gradient tolerable for algorithms depends on how long remote data has to be available. In the area of distributed query processing, for example, this includes the time between sending a query and receiving all answers from remote nodes.

[1]Without loss of generality, we assume that n_k is always contained in the first partition P_1 of e.

The *accumulative node visibility set* $E_{\text{acc}}(n_k, T)$ is the subset of n_k's join events within T defined as $E_{\text{acc}}(n_k, T) = \{e \in E_{\text{part}}(T) \mid n_k \in (e.P_1 \cup e.P_2) \wedge e.\mathit{type} = \text{join} \wedge e.t \in T\}$. Its size is

$$N_{\text{acc}}(n_k, T) = \Big| \bigcup_{e \in E_{\text{acc}}(n_k,T)} e.P_1 \cup e.P_2 \Big| \tag{2.11}$$

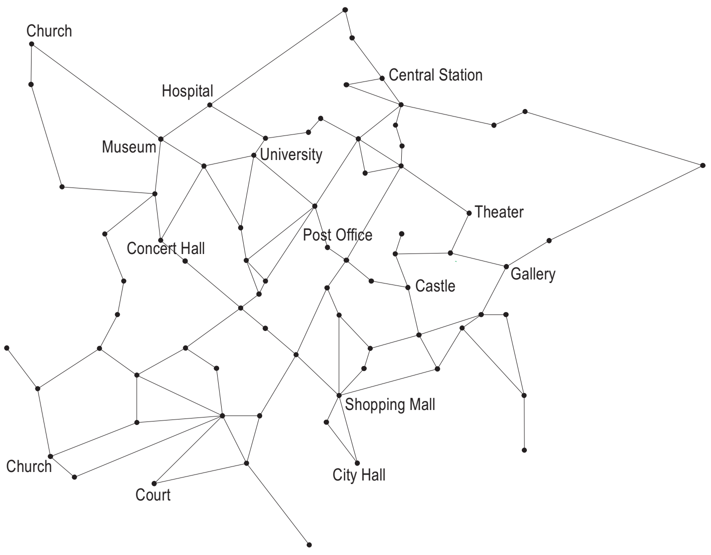
$N_{\text{acc}}(n_k, T)$ increases monotonically as partitions with new nodes are joined with n_k's partition and converges towards the set of all nodes; it will be N, if n_k has been in the same partition with all nodes over the time interval T at least once. In the context of data dissemination the gradient of $N_{\text{acc}}(n_k, T)$ is an upper bound for how fast an individual node can access all other nodes throughout the network.

2.2.5 Simulation Study

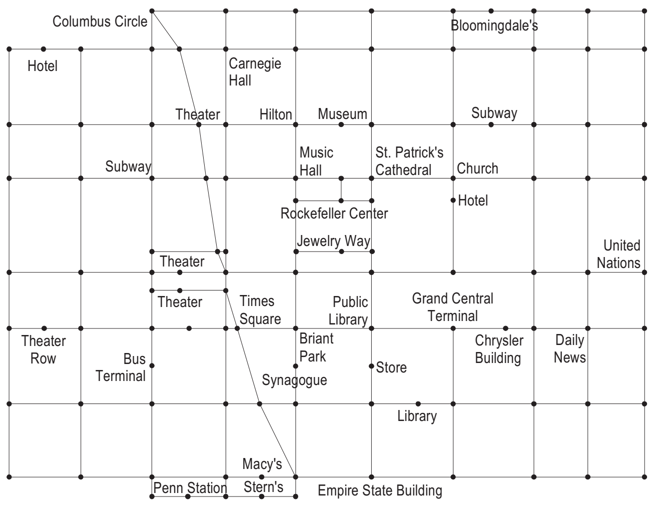
To conduct our experiments, we have used an event-based simulator which, given a mobility trace, constructs and alters the topology graph over time. The event-based approach is able to capture *every* occurring state of the topology graph over time. The mobility traces were obtained using the simulation environment presented in [SHB$^+$03]. The random waypoint model operates on an area of 2×2 km^2. For the graph-based random walk mobility model, we have modelled two graphs, representing a typical Central European city (Figure 2.2.a) and Midtown Manhattan (Figure 2.2.b). Each graph spans a total area of approximately 2.5×1.8 km^2. The similar areas of the mobility models allow us to compare the simulation results for different mobility patterns with one another. The speed of nodes was randomly chosen from the interval $[0.5, 2.0]$ m/s. The total simulation time was 3600 s.

The key parameters for performance evaluation of algorithms in MANETs are the spatial area in which the mobile nodes may move, the number of nodes in the network, the mobility model the nodes follow, and the transmission range of the wireless communication technology. The spatial node density is defined by the size of the spatial area and the number of nodes. In the following simulations, we provide detailed results for the various metrics and focus on node density and a discussion of the impact of each of the mobility models. A discussion of the impact of the transmission range is provided in Appendix B.

We assume that all network nodes are mobile, thus inducing changes in the topology graph over time. We use two mobility models in our experiments: the *random waypoint mobility model* [BMJ$^+$98] and the *graph-based mobility model* [THB$^+$02], which model typical scenarios in an open field and urban area, respectively. In the random waypoint model, nodes move on straight lines inside of the simulation area by repetitively selecting a random destination and random speed. Because this model has the property of a steadily decreasing average node speed over time [YLN03], we choose the random speed from an interval $[v_{\min}, v_{\max}]$, with $v_{\min} > 0$. In the graph-based model, the mobility of nodes is spatially constrained by a graph. Each vertex in the graph represents a particular location (x, y) within a geographic region, such as points of interest or road intersections. Graph edges connect these locations, and represent routes in the region, such as streets. Each node chooses a vertex of the graph as its destination and a speed from a given interval randomly and moves to that destination on the shortest path of the graph at the chosen speed. In contrast to the random waypoint mobility model, the graph-based random walk mobility model is more restrictive and prohibits completely arbitrary movement, reflecting more closely scenarios in urban areas.



a. Central European city graph.



b. Midtown Manhattan graph.

Figure 2.2: Input graphs to the graph mobility model.

2.2.5.1 Network-wide Partition Metrics

Number of Partitions

Figure 2.3 shows the results for the average number of partitions $|\Pi|_{\text{avg}}(T)$ as a function of the number of nodes in the network. For small networks, $|\Pi|_{\text{avg}}(T)$ is mostly determined by the number of nodes in the network, which define the upper bound of the number of partitions. Once the node density is higher than 75 nodes/km^2, which corresponds to 300 nodes in our simulations, $|\Pi|_{\text{avg}}(T)$ steadily decreases. However, even for larger networks with a density of 600 nodes/km^2, corresponding to 2400 nodes, the relatively high number of partitions (5 to 10 on average) still needs to be taken into account, especially for the implementation of algorithms that require a reachability of all nodes in the network, such as reliable broadcast techniques.

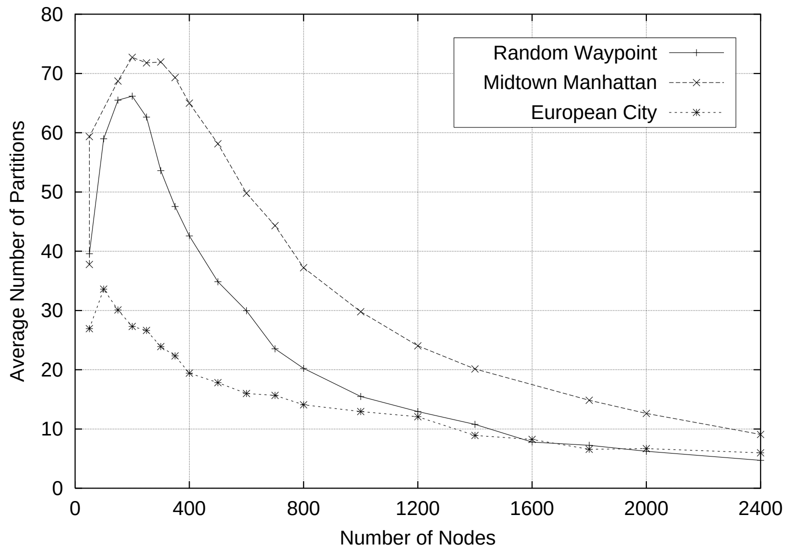


Figure 2.3: Average number of partitions $|\Pi|_{\text{avg}}(T)$ as a function of the number of nodes n for $r_{\text{tx}} = 100$ m and all three mobility models.

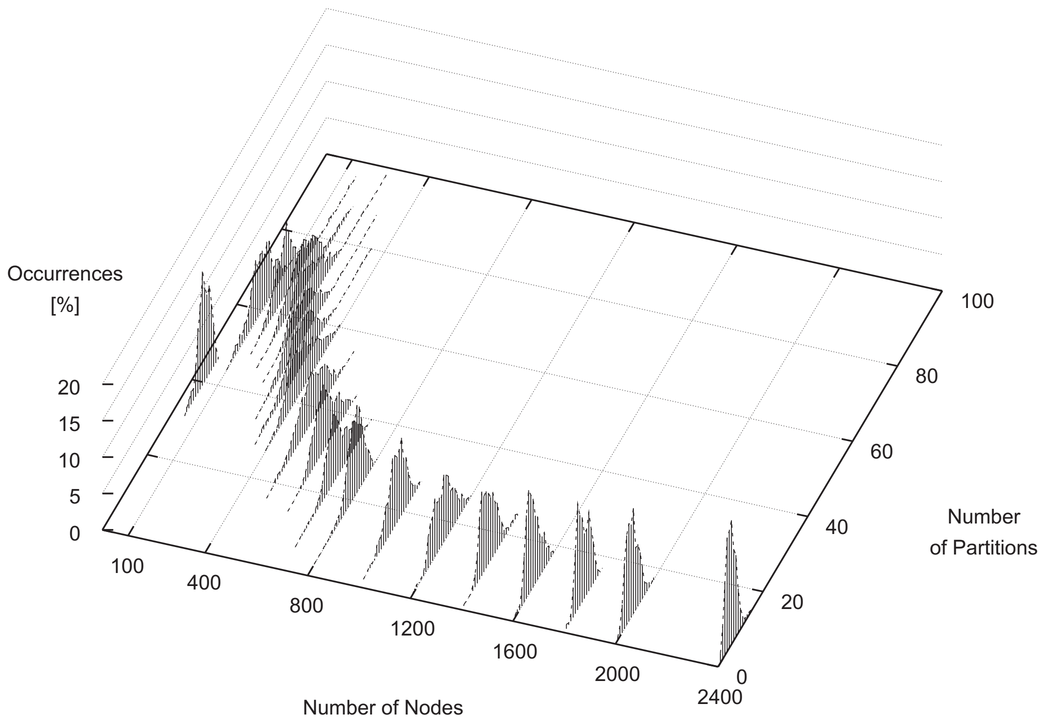


Figure 2.4: Frequency distributions of the number of partitions $|\Pi|(e)$ as a function of the number of nodes n for $r_{\text{tx}} = 100$ m and the random waypoint mobility model.

A more detailed view of Figure 2.3 is depicted in Figure 2.4 for the random waypoint mobility model scenario. Each average number of partitions in Figure 2.3 is the frequency distribution of the number of partitions for a particular number of nodes.

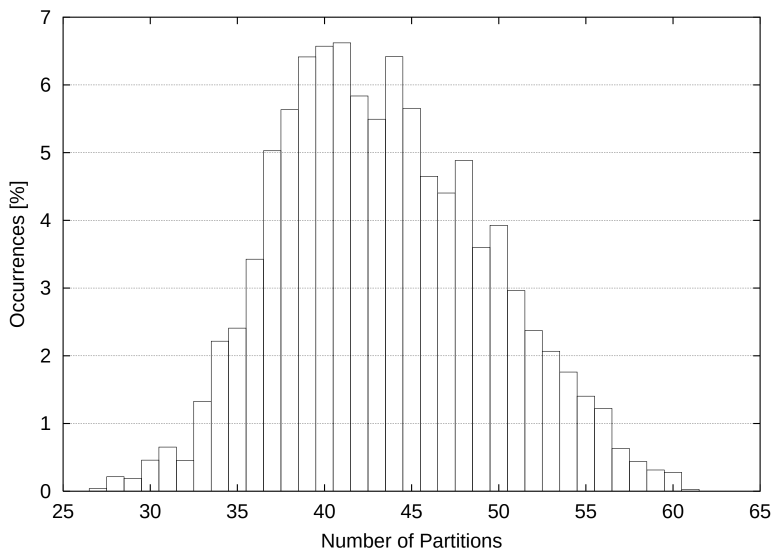


Figure 2.5: Frequency distribution of the number of partitions $|\Pi(e)|$ for $n = 400$ nodes, $r_{\text{tx}} = 100$ m and the random waypoint mobility model.

Figure 2.5 shows an extract of the 3D plot for the case where the number of nodes is 400. For the graph-based models, we have observed a very similar distribution with respect to the averages in the number of partitions. A key observation is the fact that no arbitrary number of partitions occur for any selected number of nodes. With increasing number of nodes, the distribution is even more distinct around the average. Assuming that nodes are able to detect the node density in their vicinity, the selection of a matching distribution allows reasoning about the number of partitions in regions of the network for the presented mobility models.

Size of Partitions

The results for the average size of partitions $S_{\text{avg}}(T)$ as a function of the number of nodes is presented in Figure 2.6. $S_{\text{avg}}(T)$ increases with the number of network nodes. Furthermore, the standard deviation of the average partition size also increases with the size of partitions.

Figure 2.7 shows the distribution of partition sizes for each measured average in Figure 2.6 and reveals the reason for the high standard deviation. In all scenarios, the occurrence of very small partitions, e.g., isolated nodes and partitions of up to 5 nodes, is very common. In the settings with low node density, these small sizes dominate the distribution. With increasing node density, the occurrence of very large partitions becomes more frequent and leads to a large standard deviation for the size of partitions.

Figure 2.8 displays the distribution of partition sizes for the 400 node random waypoint scenario, which is extracted from Figure 2.7. It shows the situation where the elevation at around 280 nodes is becoming more distinct, which is more and more the case towards the high-density end of the considered density spectrum.

The result that average sized partitions are rare leads to the conclusion that, if a node is not located inside a small partition, there is a high probability that it is located in a large partition.

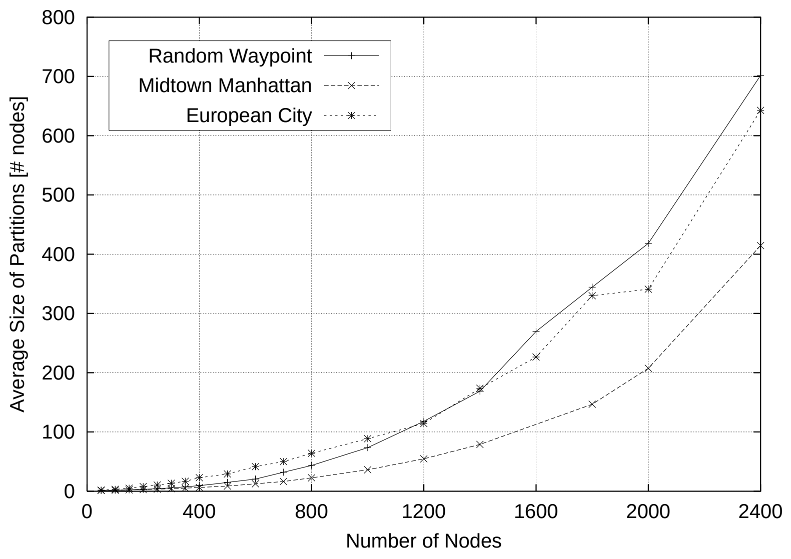


Figure 2.6: Average size of partitions $S_{\text{avg}}(T)$ as a function of the number of nodes n for $r_{\text{tx}} = 100$ m and all three mobility models.

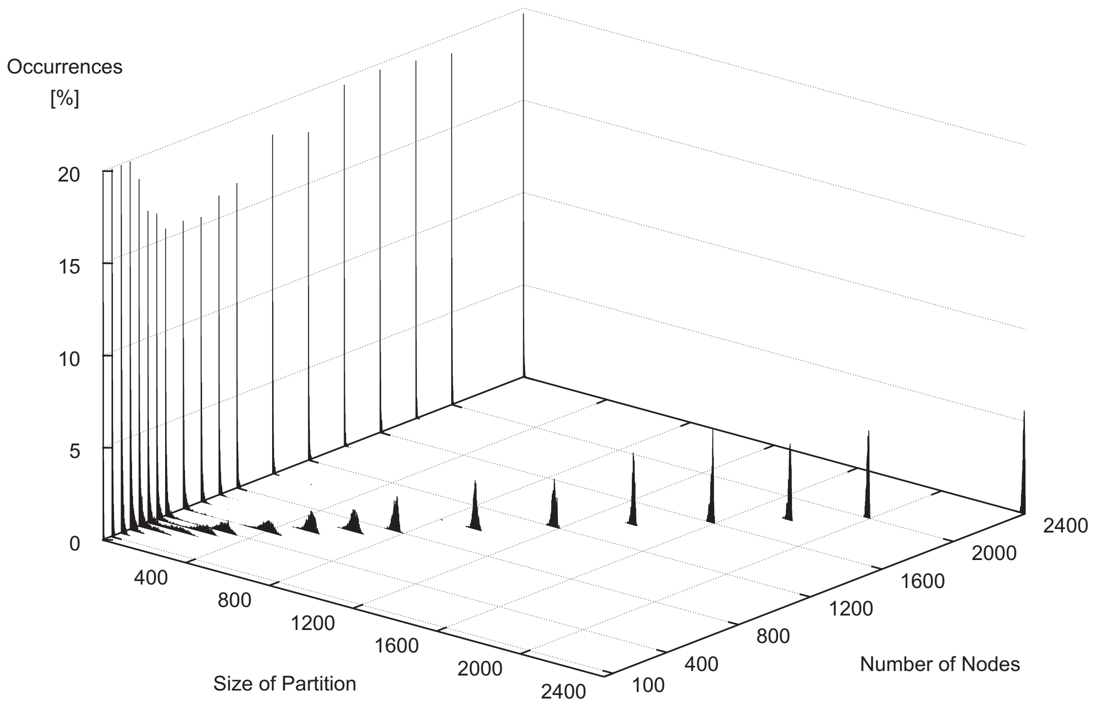


Figure 2.7: Frequency distribution of the partition size $H(x, T)$ as a function of the number of nodes n for $r_{\text{tx}} = 100$ m and the random waypoint mobility model.

A node may distinguish these cases with relatively low communication overhead by calculating the k-hop neighborhood, where, for example, $k = 4$ to decide whether or not the node is in a partition with at most 5 nodes. In our results, simply checking for partitions of a size smaller than 5 would allow us to determine with a high probability ($\geq 80\%$) whether or not a node is located in a large partition. This information could be valuable for many data management

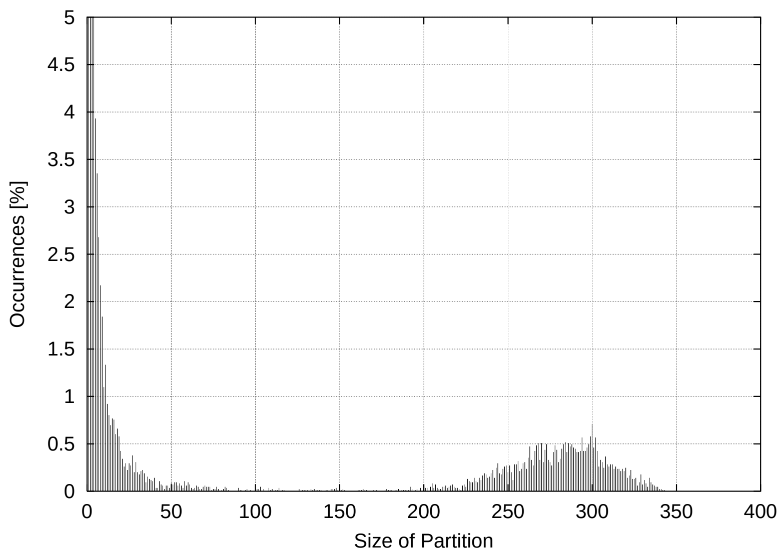


Figure 2.8: Frequency distribution of the partition size $H(x, T)$ for $n = 400$ nodes, $r_{\text{tx}} = 100$ m and the random waypoint mobility model.

algorithms involving, for example, multicasting or broadcasting, where it would be useful to send important information while being inside a large partition.

Partition Change Rate

Figure 2.9 shows the average partition change rate $R_{\text{part}}(T)$ over the number of nodes. For a very small number of nodes, the partition change rate is low and increases up to approximately 300 nodes. For this low density, the number of partitions containing only one or a few nodes is high. Such partitions are connected with a small number of links only. Therefore, a link change leads to a partition change very frequently. Partitions become connected with more links as the number of nodes increases further, leading to partitions which are more robust against link changes. As a consequence, the partition change rate decreases.

Partition Size Ratio

The results for the average partition size ratio $\text{PSR}_{\text{avg}}(T)$ as a function of the number of nodes is presented in Figure 2.10. While $\text{PSR}_{\text{avg}}(T)$ steadily increases with the size of the network, it asymptotically approaches its maximum value of one. This is due to the fact that join and split events mostly involve a large and a small partition. It is interesting to notice that independent from the specific mobility model used in the experiments, the partition size ratio metric indicates that there is a stable, well-connected core and a series of “satellite” groups that join and split throughout the simulation.

To underline this effect, we have visualized the sizes of both partitions involved in a partition event in Figure 2.11. The complete plot corresponds to the measuring point at 400 nodes for the random waypoint mobility model in Figure 2.10. The number of occurrences of any possible pair of partition sizes has been drawn as columns on the plane. The distribution shows

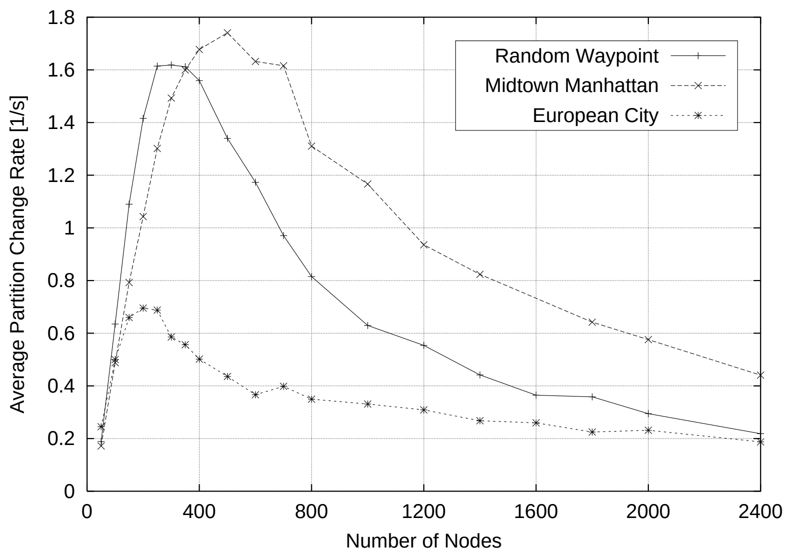


Figure 2.9: Average partition change rate $R_{\text{part}}(T)$ as a function of the number of nodes n for $r_{\text{tx}} = 100$ m and all three mobility models.

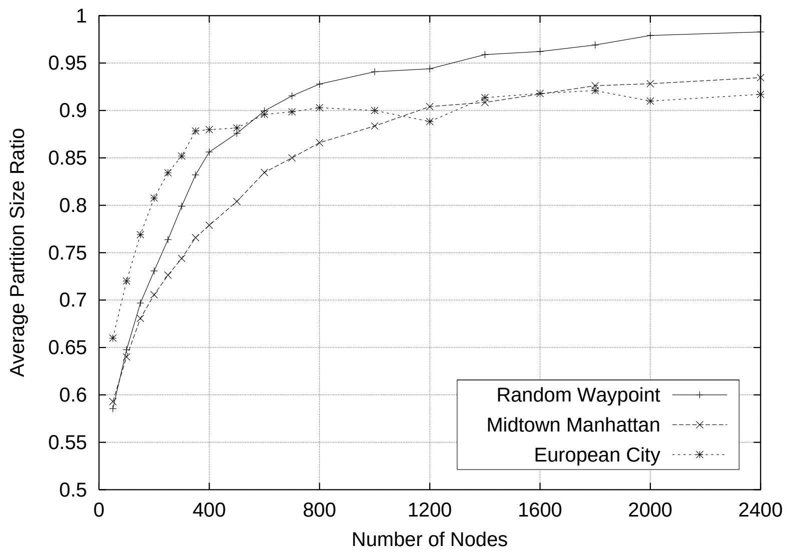


Figure 2.10: Average partition size ratio $\text{PSR}_{\text{avg}}(T)$ as a function of the number of nodes n for $r_{\text{tx}} = 100$ m and all three mobility models.

that bursts dominate in two regions. For large partition size ratios, which is visible from the elevation at approximately 250 nodes of partition 1, many pairs of one large and one small partition are involved in partition splits and joins. In addition, a high peak occurs at a very low number of nodes of partition size 1, where very small partitions are split and joined.

We observed that with increasing node density, this pattern evolves similarly as that of partition sizes presented in Figure 2.7, resulting in two narrow peaks at the low and high end of the partition 1 scale. Correlating both of these metrics leads to the conclusion that large partitions remain large over time and are only affected by small groups of joining and leaving nodes.

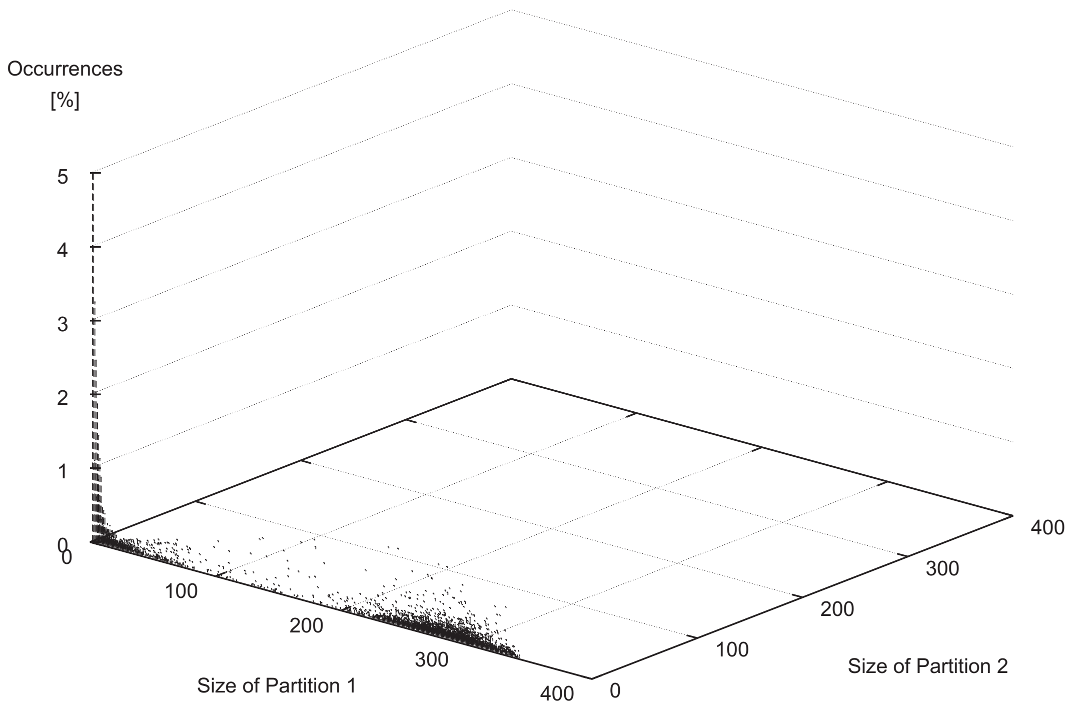


Figure 2.11: Frequency distribution of the partition size ratio PSR(e) for $n = 400$ nodes, $r_{\text{tx}} = 100$ m and the random waypoint mobility model. The isolated dots are located on the x-y-plane and indicate single occurrences of certain partition change ratios.

Therefore, the existence of a well-connected core, backed up by the experimental evidence provided in these graphs, indicates that independently of the mobility model used, important data that needs to be accessed by a large number of nodes can be stored at the core. It might be interesting to determine whether or not there is a geographical correlation between the presence, the size and the location of such a core.

2.2.5.2 Node-centric Partition Metrics

Node Partition Change Rate

Figure 2.12 shows the results of the average node partition change rate $R_{\text{navg}}(T)$ as a function of the number of nodes. The change rate as experienced by individual nodes on average shows the same characteristic behavior as the network-wide partition change rate in Figure 2.9. This indicates that the partition changes across networks of different sizes behave similar as those experienced by individual nodes. On average, $R_{\text{navg}}(T)$ is lower than the network-wide partition change rate, because not every node is affected by every change. Nevertheless, as the size of the network increases, the difference between the node rate and the system-wide rate becomes smaller. This corresponds to the increasing size of partitions observed in Figure 2.6, so that changes in large partitions have an effect on a larger number of nodes.

In Figure 2.13 we have extended the diagram in Figure 2.12 by including the distribution of $R_{\text{navg}}(T)$ across nodes. A closeup for the distribution is shown in Figure 2.14 for the case of

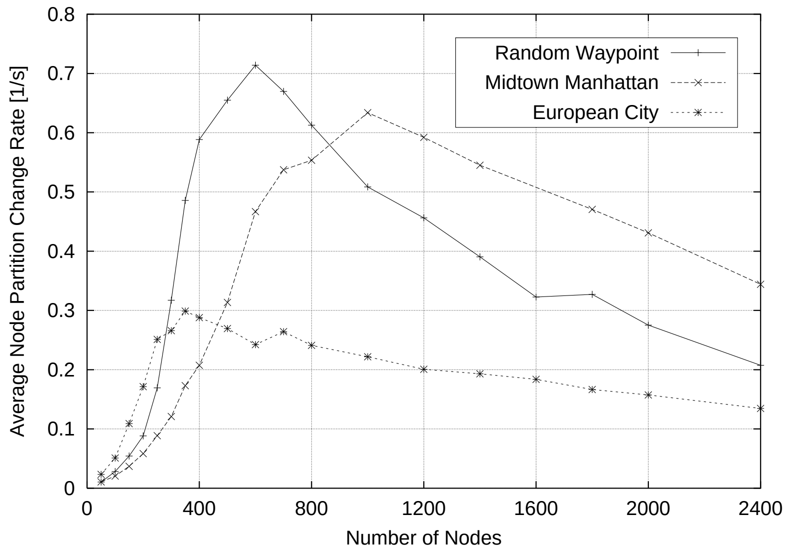


Figure 2.12: Average node partition change rate $\mathrm{R}_{\mathrm{navg}}(T)$ as a function of the number of nodes n for $r_{\mathrm{tx}} = 100$ m and all three mobility models.

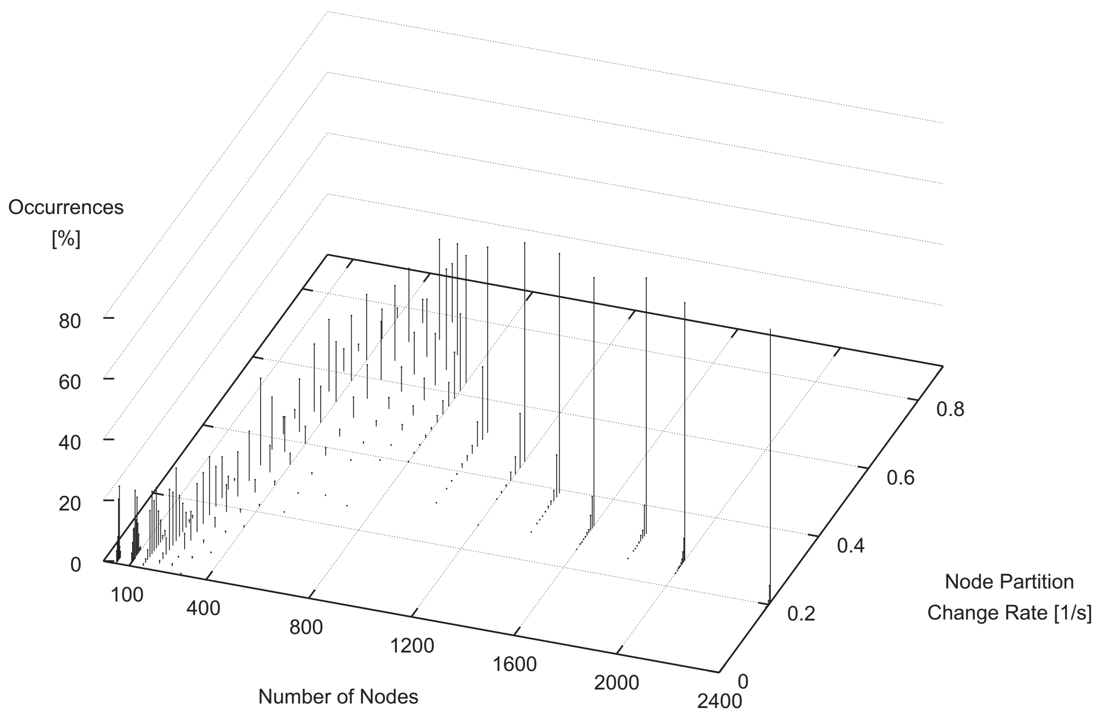


Figure 2.13: Frequency distribution of the node partition change rate $R_{\mathrm{n}}(n_k, T)$ as a function of the number of nodes n for $r_{\mathrm{tx}} = 100$ m and the random waypoint mobility model.

the random waypoint mobility model with 400 nodes. The frequency distribution reveals that $R_{\mathrm{navg}}(T)$ is evenly dispersed towards its average for network sizes of up to approximately 400 nodes. For a larger number of nodes, the node partition change rate distribution is more and more skewed to the right, reaching a peak towards the end of the considered density spectrum. In such networks, large partitions dominate and the node partition change rate of many nodes is affected, if such partitions experience a join or a split. According to our results, developers of distributed algorithms can assume that the partition change rate experienced by individual nodes lies within a small range of frequencies for all nodes in a given network scenario.

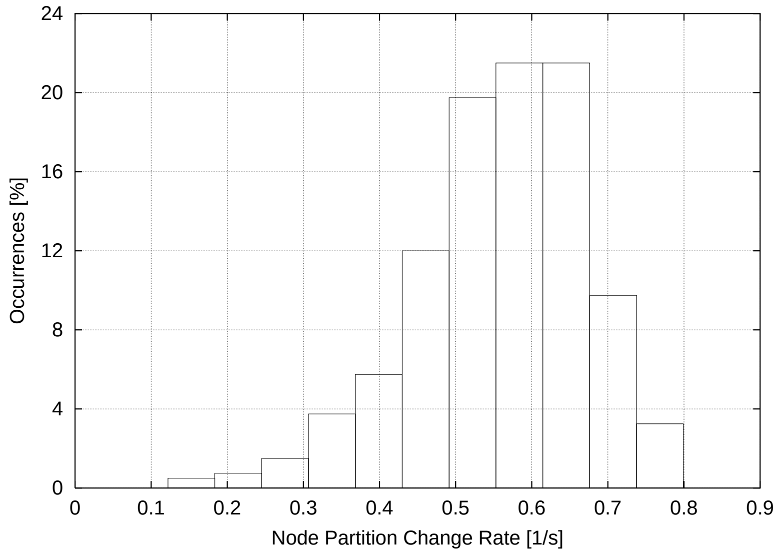


Figure 2.14: Frequency distribution of the node partition change rate $R_{\mathrm{n}}(n_k, T)$ for $n = 400$ nodes, $r_{\mathrm{tx}} = 100$ m and the random waypoint mobility model.

Node Separation and Connection

Figure 2.15 shows the average of the sum of node separation times τ_{sep} over the number of nodes. We selected $n/2$ disjoint node pairs for each experiment at random, i.e. every node is contained in exactly one of the pairs. As the number of nodes in the network increases, the average node separation time decreases. For networks between 300 and 500 nodes the slope is very steep. This is associated with the decreasing partition change rate presented in Figure 2.9 which leads to more stable partitions in networks with a larger number of nodes.

Figure 2.16 provides a detailed view on τ_{sep} for $n = 400$ and the random waypoint mobility model. In this scenario, 50% of the selected node pairs are separated for more than 1500 s.

Figure 2.17 shows the number of separations K_{sep} for the same setting, revealing that more than 50% of the pairs are separated 30 times during the experiment. This leads to the conclusion that the number and duration of separations for node pairs should not be neglected when designing distributed data management algorithms, because nodes may not be available for long periods of time and connection times are limited.

The distribution of node separation times as a function of the number of nodes is shown in Figure 2.18. For a small number of nodes, the node separation times are distributed across the whole time spectrum. For instance, out of 100 nodes, many are separated for the duration of the experiment. This results in a peak at a node separation time of 3600 s, e.g., for 20% of node pairs. As the number of nodes increases, the node separation times become smaller and show a distinct distribution with many occurrences of low separation times between nodes.

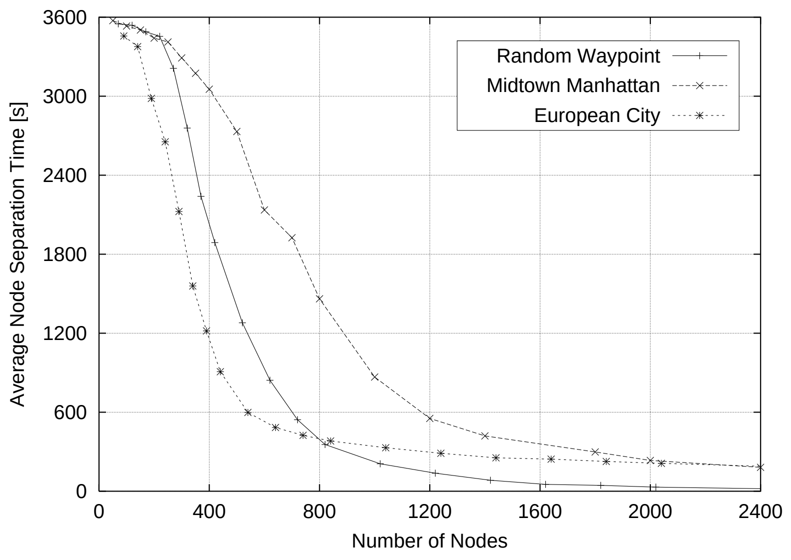


Figure 2.15: Average over the sum of node separation times $\tau_{\text{sep}}(n_1, n_2, T)$ for $n/2$ disjoint pairs of nodes (n_1, n_2) as a function of the number of nodes n for $r_{\text{tx}} = 100$ m and the random waypoint mobility model.

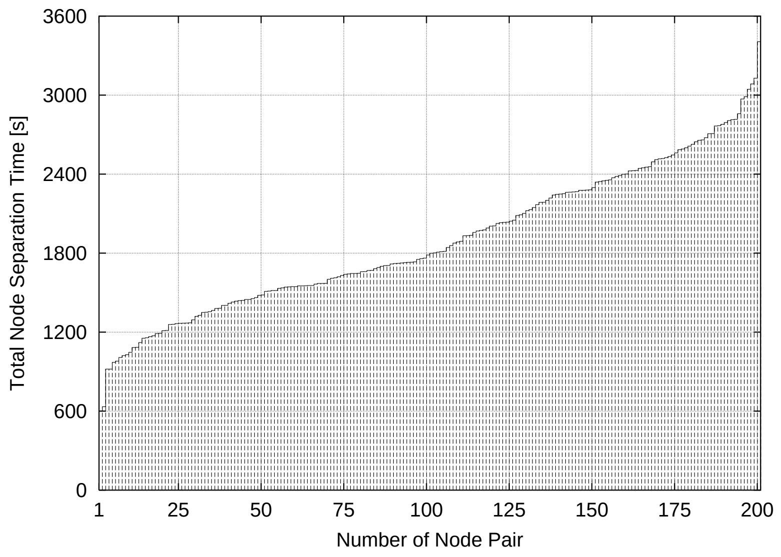


Figure 2.16: Sum of node separation times $\tau_{\text{sep}}(n_1, n_2, T)$ for 200 disjoint pairs of nodes (n_1, n_2), $n = 400$ nodes, $r_{\text{tx}} = 100$ m and the random waypoint mobility model.

Figure 2.19 shows a two-dimensional extraction for the distribution of 400 nodes. About 75% of the node separation times are below 50 seconds, and about 35% are even below 10 seconds. This information is very valuable for algorithms that need to estimate when a node will be reachable again after a communication failure has occurred. For example, a node that has not succeeded in sending information to a particular other node may retry to initiate communication after 10 and 20 seconds, because 35% and 55% of all separations in a network with 400 nodes are shorter than 10 and 20 seconds, respectively.

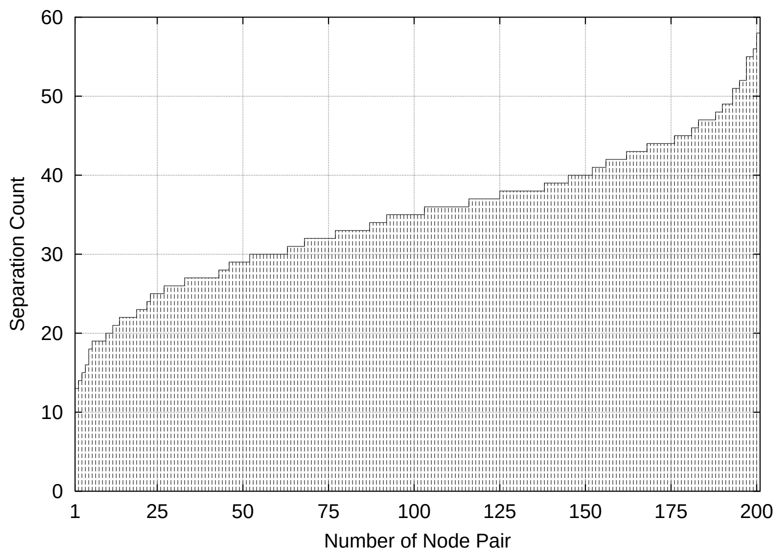


Figure 2.17: Number of node separations $K_{\text{sep}}(n_1, n_2, T)$ for 200 disjoint pairs of nodes (n_1, n_2), $n = 400$ nodes, $r_{\text{tx}} = 100$ m and the random waypoint mobility model.

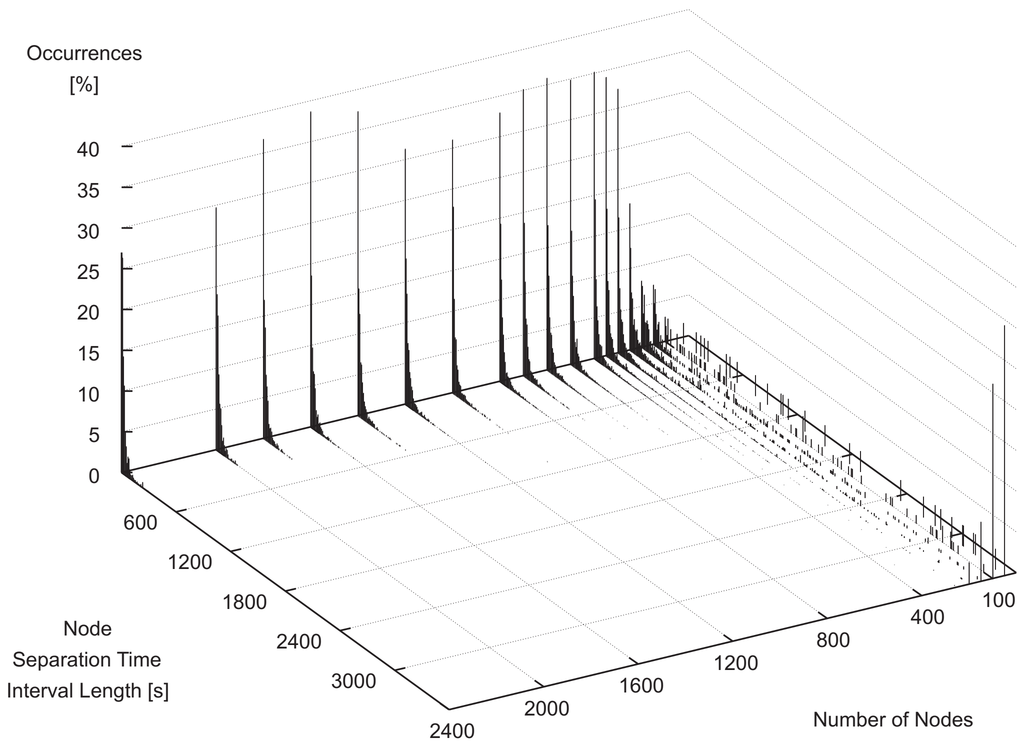


Figure 2.18: Frequency distribution of node separation times as a function of the number of nodes n for $r_{\text{tx}} = 100$ m and the random waypoint mobility model.

Node Visibility Sets

Figure 2.20 and 2.21 show the evolution of the continuous and accumulative node visibility sets, respectively, as a function of time for five representative network sizes. The measurements for the average size of both sets were started in the middle of the experiment at $t = 1800$ s to

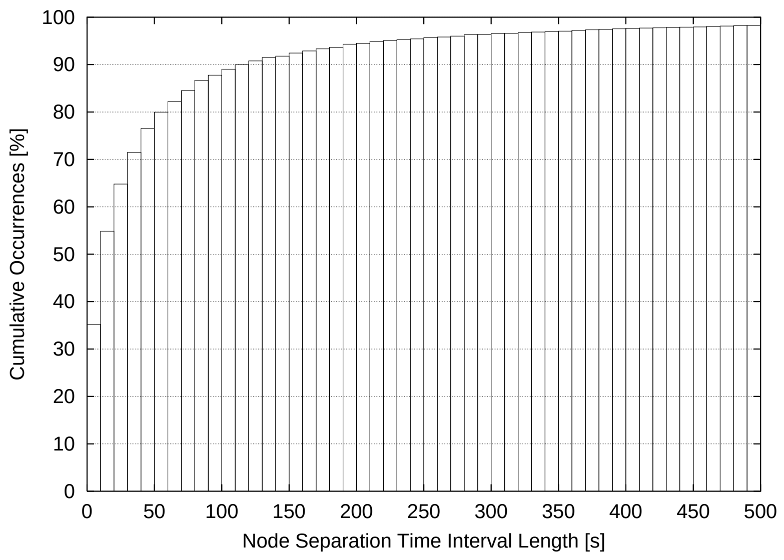


Figure 2.19: Cumulative frequency distribution of node separation times for $n = 400$ nodes, $r_{\text{tx}} = 100$ m and the random waypoint mobility model.

avoid the influence of transient effects of the mobility models. The size of the continuous node visibility set shows a steady decrease over time. Note that at the start of the measurement, the size of this set is equal to the size of the partition the node of this set is located in. The average size of the continuous node visibility set is significantly higher than the average size of partitions, because its average is dominated by the large set sizes of many nodes that are located in large partitions. The average size of the accumulative node visibility set steadily increases over time with the same average value as the continuous node visibility set at the beginning of the measurements. How fast the average size approaches the maximum of 100 % depends on the number of nodes in the network.

For distributed algorithms, the continuous node visibility set gives an approximation if and for how long a given number of nodes remain in the same partition as a particular node. This information may be used for algorithms that share their local data or perform distributed calculations over longer periods of time, e.g., applications in mobile ad-hoc networks that perform distributed data aggregation. The average size of the accumulative node visibility set is an indicator for the amount of time it takes until a node containing a data source for particular information may be queried by a certain number of nodes in the network, e.g., a node that contains data about a particular location.

2.2.5.3 Impact of the Mobility Models

When comparing the results of the different mobility models used in the experiments, i.e. the random waypoint mobility model and the two instances of the graph-based random walk mobility model, it is noticeable that they share the same characteristic behavior for all metrics examined. However, the particular results measured vary significantly in quantitative terms.

In general, it can be observed that the quantities obtained for the random waypoint experiments are between those used for the two graphs (Figure 2.2). Taking the results for the average number of partitions $|\Pi|_{\text{avg}}(T)$ as an example, the Manhattan graph reveals a significantly

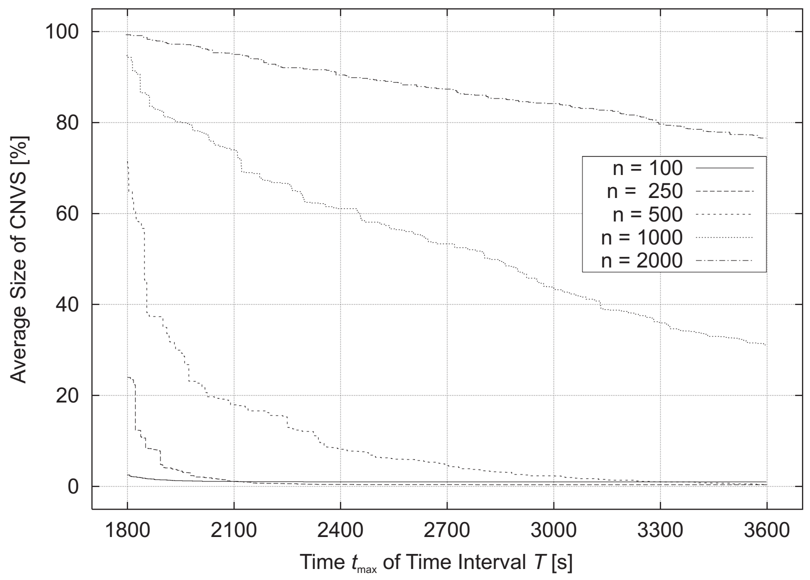


Figure 2.20: Average size of n continuous node visibility sets $N_{\text{cont}}(n_k, T)$ as a function of T for different n, $r_{\text{tx}} = 100$ m and the random waypoint mobility model.

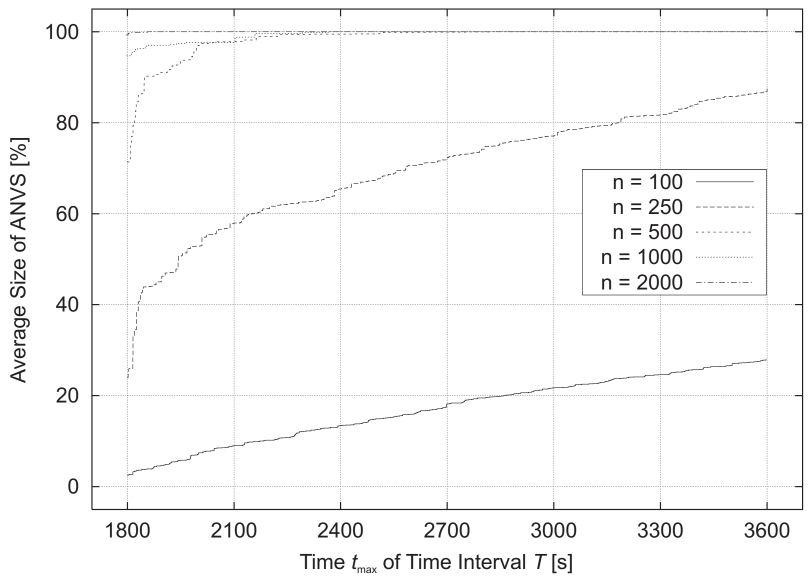


Figure 2.21: Average size of n accumulative node visibility sets $N_{\text{acc}}(n_k, T)$ as a function of T for different n, $r_{\text{tx}} = 100$ m and the random waypoint mobility model.

higher average number of partitions than the other two mobility models, which becomes more prominent for network sizes between 500 and 1500 nodes. The comparison of the two graphs yields a longer total edge length for the Manhattan graph, while it has approximately the same number of vertices. Additionally, the European city graph has a smaller average edge degree. Both graphs have a base area of approximately 4 km^2. In the Manhattan graph the movement of nodes is restricted to an almost regular grid pattern with distances in between those grid rows that are larger than the assumed transmission range. As a consequence, nodes moving on parallel grid rows/columns are separated for longer periods of time.

The results obtained with the two different city graphs underline that the partitioning behavior depends especially on the spatial structure of the system's environment. The analysis of this dimension is out of the scope of this dissertation and left to future work.

2.2.6 Conclusions: Network Partitioning

The analysis of partitioning has shown that network partitions occur frequently even in networks with reasonably high node densities. For example, in a network containing 800 nodes on a 4 km^2 area and a transmission range of 100 m, we found more than 20 partitions on average. For such scenarios, the partitioning situation should not be neglected in the design of distributed data management algorithms in mobile ad-hoc networks.

For the scope of this dissertation, we can draw two main conclusions. First, the fundamental algorithms to be developed shall consider the occurrence of partitioning as a common case that is likely to occur. While the situation of a partitioned network implies that communication is not possible between partitions while they exist, algorithms must still be designed in a way that partitioning does not lead to unresolvable failures in the algorithms' operation. We will address this issue in particular in the design of the core storage approach in Chapter 3.

The second important result is the fact that the mobility of mobile ad-hoc networks leads to the situation where different network partitions eventually join again, assuming some sensible node density that allows the general operation of the network in the first place. This observation has the key implication that the duration of partitions is finite in general. After a partition join, therefore, measures shall be taken to also unify potential temporary inconsistencies in the operation of the algorithms. We will consider this observation specifically in the design of the data migration approach, which we introduce in Chapter 4.

2.3 Location-centric Storage (LCS)

Location-centric storage is a fundamental paradigm in the field of mobile data management (MDM) that lies at the basis of our algorithms. The general goal of MDM is to provide mobile clients (network nodes) with the data they need, when they need it, and where they need it [PJC06]. With respect to these three aspects, MDM addresses all problems related to how data can be obtained, stored, processed, and delivered to the mobile clients.

In its original form, MDM means data management in mobile (nomadic) computing [IB93, IB94] (Section 1.2.1), where mobile clients are connected to the infrastructure via a direct radio uplink. Due to the mobility of clients, many new challenges related to MDM arise, such as the general impact of mobility on data management, disconnected operation (e.g. prefetching, hoarding), energy-efficient data access, and wireless data broadcasting. These issues have been considered by a large body of related work (see, for example, [PS97, Bar99, GS02, Kum06]) and are orthogonal to the scope of this dissertation.

In mobile ad-hoc networks, not only is the data delivered to mobile devices, it is also managed by the mobile devices themselves, according to the peer-to-peer networking paradigm (Section 2.1.1). Intuitively, this is much more challenging due to the characteristics of MANETs as discussed in Section 2.1.2 and the potential occurrence of network partitioning as shown in Section 2.2. For these reasons, besides the requirement of new protocols on link, routing, and transport layer (Section 2.1.1), many new challenges arise for MDM in areas including data acquisition and propagation, caching and replication, as well as aggregation and processing, in order to provide data to the services running in these networks.

In MANETs, one of the key questions that needs to be answered at the basis of mobile data management is the specific location where data is to be stored in the network, which is, optimally, close to where clients will need it. The storage location comes in two flavors: the *where* regarding the geographic region in the network on one side, and the *where* regarding which nodes are recruited to store the data in the end.

Regarding the geographic region, an important hint comes from the primary context of location (Section 1.3.1), which is of utmost importance for context-based applications. In the classes of context-based applications considered in Nexus (Section 1.3.2), we can observe the important property of locality regarding the access to spatial data. For example, in a geographic range query, data items are retrieved based on their inclusion within the specified region, implying a close proximity between the reference region and the items' location. Another example is the observation of spatial events, such as the meeting of two entities, which also naturally involves a close spatial relation, that is, locality, between the involved entities. This locality assumption is true for many context-based applications and can be exploited in algorithm design.

The second issue pertaining to which nodes should store data is challenging in any wireless multihop network due to the following reasons. Assuming classical node-centric routing, each data access implies that a client must know in advance which particular node to address in order to retrieve a specific data item. This advance knowledge requires, in general, the implementation of explicit location servers that provide global information about the whereabouts of each node in the network. Due to the maintenance of a globally distributed location database (e.g. [LJC+00, Bha03, KFWM04]), they incur significant additional communication overhead.

However, due to the information-centric nature of context-based applications, the node identities should be considered secondary, and a more appropriate solution should focus on the data itself. Such a data-centric storage (DCS) approach was introduced by Ratnasamy et al. in [RKY+02, RKS+03, SRK+03] in the context of wireless multihop networks and targets at this problem. In contrast to node-centric storage, the data itself serves as a reference to the geographic location in the network where a specific data item is to be stored. The question which particular node stores that data item is then reduced to a local resolution protocol, such as Greedy Perimeter Stateless Routing (GPSR) in the specific case of DCS. In contrast to node-centric approaches, DCS is highly scalable, because a location service is not required.

Based on a combination of locality and data centricity according to the previous elaborations, we are now in the position to define the paradigm of location-centric storage. By the term spatial data, we designate data that is associated with a geographic position.

Definition 2.1. *Canonical location-centric storage (C-LCS)* refers to the storage of spatial data in a wireless multihop network, where each spatial data item is stored on the network node that is located closest to the geographic position specified by the spatial data item.

By canonical, we refer to the fact that a direct relation exists between pairs of data items and storage nodes. This rather strict definition, however, has two main drawbacks for being of practical value. First, C-LCS implies that data items are associated with storage nodes on an individual basis. This association inherently leads to a very fine-grained storage of data items over virtually all nodes in the network. Second, the mobility of either nodes or the entities modelled by the data items may lead to frequent reassociations, which requires to relocate data items between nodes on a continuous basis.

For these reasons, we define the following important twofold relaxation of C-LCS, which we assume in the remainder of this dissertation and which brings the essential quantum of flexibility to the design of our algorithms.

Definition 2.2. *Location-centric storage (LCS)* refers to the storage of spatial data in a wireless multihop network, where each spatial data item is stored on any network node that is in proximity to a geometric reference location. This location is in turn in proximity to the geographic position specified by the spatial data item.

The relaxation introduced in the definition of LCS in comparison to its canonical form is twofold due to the introduction of a reference location that decouples between storage nodes and data items. Consequently, for both data items and storage nodes, the proximity relation in terms of geometric distance is specified separately. The use of the vague term proximity is deliberate and provides the essential flexibility and tuning knob for algorithm design. Both proximity relations will be specified in detail in Chapter 3 through 5.

At this point, we are, nevertheless, able to formalize the notion of proximity by the term *spatial coherence*, which quantifies the magnitude by which proximity between a data item and its associated reference location is maintained.

Definition 2.3. The *spatial coherence* of a data item is defined as the mean geometric distance between the data item and its associated reference location over time.

Assuming a data item is stored on the same node over time, this definition is effectively applicable to the mean distance that the storage node itself maintains to the reference location. As a general rule of thumb, smaller values of spatial coherence imply more efficient overall storage solutions. We will employ spatial coherence as a performance metric when assessing storage efficiency in detail in the data migration approach (Chapter 4).

2.4 Requirements and Reference Model

Based on the discussions in the preceding sections and Chapter 1, we are now able to state the following set of key requirements that the fundamental storage mechanisms for context-based services in mobile ad-hoc networks shall consider. These requirements bundle the characteristics of mobile ad-hoc networks from Section 2.1.2, the conclusions from network partitioning in Section 2.2.6, and the focus of this dissertation according to Section 1.4.1.

(1) *Communication efficiency* addresses the MANET-specific constraints regarding energy resources and link bandwidth according to Section 2.1.2, Table 2.1. High communication efficiency will enable the execution of diverse context-based applications concurrently, without incurring critical network traffic that would adversely impact user experience, such as decreasing the quality of query results.

(2) *Robustness against significant node mobility* addresses the inherent mobility of nodes in MANETs according to Section 2.1.2, Table 2.1. The mechanisms to be developed shall tolerate node mobility of various degrees, from pedestrian to vehicular traffic in typical urban context-based scenarios. Optimally, robustness shall not be achieved at the expense of communication efficiency.

(3) *Robustness against low node density and weak network topology* according to Section 2.1.2, Table 2.1 is essential to provide uniform performance to the best possible extent, independent of the specific magnitude of these two parameters. As in the case of (2), this type of robustness shall not adversely impact communication efficiency.

(4) *Robustness against temporary network partitioning* addresses the conclusions we have drawn in Section 2.2.6. This requirement ensures the long-term resilience of the storage mechanisms against the forming and merging of network partitions over time. The same behavior regarding the impact on communication efficiency as in (2) and (3) is desired.

(5) *Scalability with network and data model size* is essential to guarantee efficient operation of the storage mechanisms for a potentially large number of nodes or data items. If the storage mechanisms do not scale, context services will eventually show undesired behavior, such as reduced query performance or incomplete query results.

(6) *Support for rich location and query semantics* addresses the intrinsic inaccuracy characteristics of data acquisition as discussed in Section 1.4.1. These inaccuracies must be taken into account in the modelling of location and query semantics and incorporated into algorithm design, in order to provide useful query results to the user.

Figure 2.22 sketches the reference model for the storage mechanisms for context-based services in MANETs that will be implemented in subsequent chapters of this dissertation, comprising Contribution 2 to 6 as stated in Section 1.4.2. Each component in the figure addresses one or more of the preceding requirements and is annotated with the section in which it will be specified in detail. References to the corresponding evaluation sections are also indicated. The client tier is out of the scope of this dissertation and omitted in subsequent discussions. A more detailed account of the reference model in a broader scope is contained in [DWM08].

Routing Tier

The routing tier implements fundamental routing and aggregation functionality that is the basis for algorithms in the storage tier. It addresses several deficiencies of existing protocols in this layer that do not satisfactorily support the stated requirements.

Bidirectional perimeter routing is a customization of geometric routing protocols that we employ in the core data storage in Chapter 3. It addresses Requirement (1), (3), and (4) and is primarily responsible for delivering communication efficiency and robustness against weak network connectivity. Network topology exploration is a mechanism for distributed data aggregation that delivers essential local network information to data migration (Chapter 4) in the storage tier. This information is in turn used by resilient source routing to implement stable temporary routes between nodes on behalf of data migration. In combination, both mechanisms address the robustness-related Requirement (2) to (4).

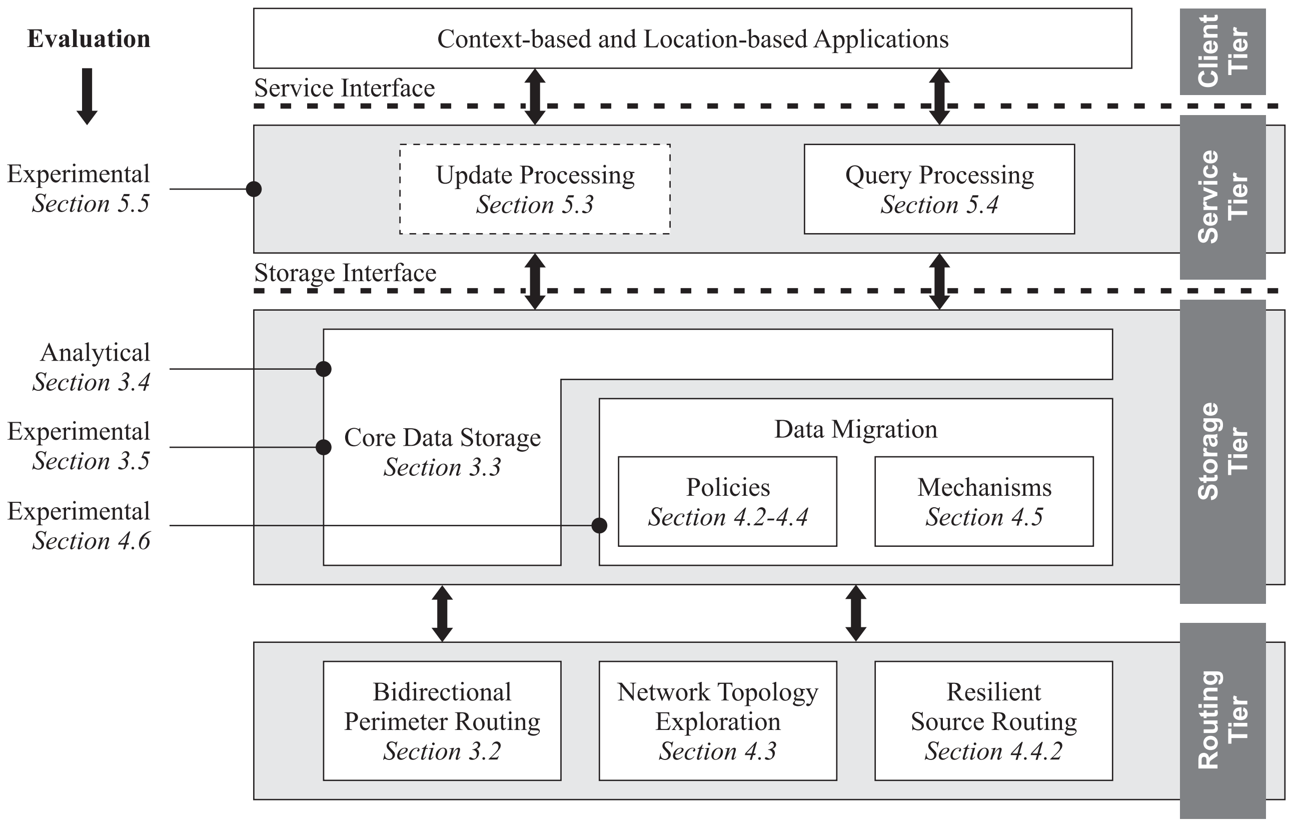


Figure 2.22: Conceptual architecture of the location-centric storage framework.

Storage Tier

The storage tier implements, in collaboration with the routing tier, efficient, robust, and scalable location-centric storage. Both tiers represent the main contribution of this dissertation and are also addressed in [DMR06a] and [DMR07]. Based on lightweight data servers, core data storage provides efficient base mechanisms for data storage by exploiting the location-centric storage paradigm. In conjunction with bidirectional perimeter routing, it specifically addresses Requirement (1), (3), and (4). By breaking down data management into geographic cells of limited size, core storage also implements Requirement (5) on the level of the storage tier.

Data migration constitutes an integral part of core data storage that addresses specifically data server mobility that is due to the movement of network nodes. Migration policies, on one hand, implement, supported by network topology exploration, strategies to decide on the best conditions under which the migration of data is to be performed. On the other hand, data migration mechanisms implement these decisions and translate them into robust network paths, in turn performing data migration via the underlying resilient source routing.

Policies and mechanisms cooperatively achieve continuous spatial coherence (Definition 2.3) under dynamic network density and topology. Thus, data migration is essential in the achievement of Requirement (1) to (3). Data migration also incorporates merging mechanisms to consolidate redundant servers upon the joining of network partitions, thereby addressing Requirement (4). Furthermore, data migration supports core data storage in achieving Requirement (5) in terms of the data size that is supported by individual migration processes.

Service Tier

The service tier contains the base services that context-based applications may exploit to implement their specific forms of context-based interaction (Section 1.3.1). This layer makes extensive use of the mechanisms provided by the storage tier. Contributions in the service tier are published in [DDM05] and [DMR06b].

In the form presented in this dissertation, the service tier comprises services for update and query processing. On one hand, update processing is responsible for delivering the data that is acquired by individual network nodes to the light-weight data servers provided by the storage tier. The essential added value in comparison to the core data storage is the consideration of data semantics in the form of models for inaccurate position information. Thereby, update processing addresses Requirement (6) on the update side.

On the other hand, query processing implements fundamental spatial queries, including range and nearest neighbor queries, which are processed based on the information provided by update processing. Query processing introduces appropriate query semantics, which are compliant with the position inaccuracy model of update processing, thus addressing Requirement (6) on the query side. By exploiting the locality assumption, query processing is also able to provide efficient localized query processing, which leads to the fulfillment of Requirement (1) and (5).

Comparison to the Nexus High-level Architecture

At this point we are able to match the conceptual architecture of the LCS framework in Figure 2.22 with the Nexus high-level architecture in Figure 1.4. A comparison of tiers shows the following two significant differences.

First, the Nexus architecture associates basic services, including query processing, with the federation tier, which in the LCS framework are located in the service tier. This discrepancy is due to the fact that Nexus' assumptions of how context information is provided are more general and include additional challenges that are related to the federation of multiple models each representing possibly overlapping aspects of the physical world. In contrast, the LCS framework assumes a single context model, which only requires basic data merging functionality that update and query processing are able to implement directly.

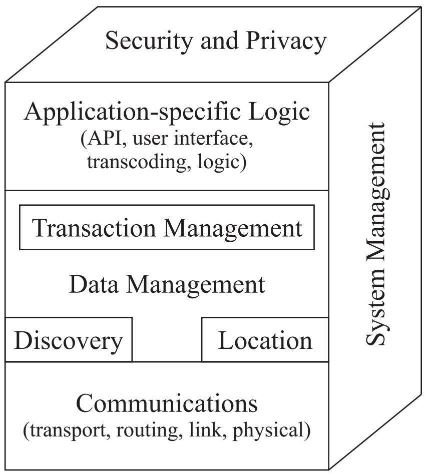
Second, the LCS framework architecture provides the additional storage and routing tier, which collaboratively absorb the MANET-specific characteristics that make these networks distinct from infrastructure-based networks. Specifically, we can observe from the previous discussions that the robustness-related Requirement (2) to (4) are addressed by the storage and routing tier alone and need no further consideration in the service tier. As a result of the encapsulation of MANET-specific characteristics in the service and routing tier, services and applications to be provided in higher tiers of the framework for LCS may draw more easily on results from infrastructure-based context management, such as Nexus.

2.5 Related Work

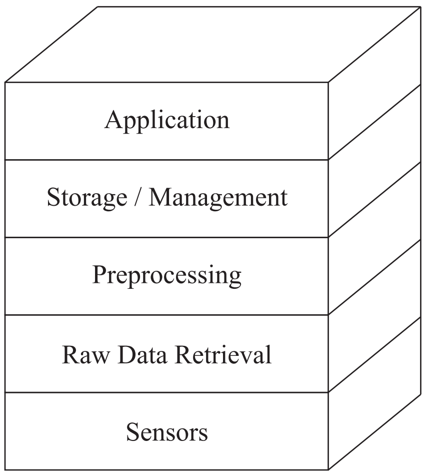
In this section, we review existing work related to our framework of fundamental storage mechanisms for location-based services in mobile ad-hoc networks. We begin with a discussion of related context-aware systems and middlewares in Section 2.5.1, followed by more detailed analyzes of the routing and storage tier in Section 2.5.2 and the service tier in Section 2.5.3.

2.5.1 Context-aware Systems and Middlewares

For the discussion of the spectrum of context management systems and middlewares we consult two reference models presented in [PJC06] and [BDR07]. The data management framework for MANETs in [PJC06], shown in Figure 2.23.a, contains several layers that each address a specific aspect of data management. In this model, the communication and data management layer correspond to the routing and storage tier in Figure 2.22, respectively. The authors of [BDR07] present a layered conceptual architecture for mobile computing environments, shown in Figure 2.23.b. This model contains, among others, layers for raw data retrieval, preprocessing, and storage / management, which together roughly correspond to our storage and routing layer. In both reference models, common services, including update and query processing (Figure 2.22), are included in the data management and storage / management layer.



a. Layered data management framework for MANETs according to [PJC06]



b. Abstract layer architecture for context-aware systems according to [BDR07]

Figure 2.23: Reference models for context management in the literature.

Related system and middleware approaches that consider aspects of context management and which fit into the reference models in Figure 2.23 are abundant. An overview of approaches that are more closely related to the scope of this dissertation can be found in [Khe06, BDR07, DHH07, Kja07]. In the following, we discuss a number of significant representative approaches with a focus on the data management layer and the kind of storage mechanisms used by each approach. The discussion follows a classification by system structure (Figure 1.5).

Mobile Computing Environments

Many context management platforms for mobile computing environments implement the data management layer in Figure 2.23 in the fixed communication network. Apart from the Nexus platform [GBH$^+$05], which we have previously discussed in Section 1.3.2 and 2.4, the following representative approaches shall be considered.

The authors of [LSD$^+$02] describe a middleware for the collection of context information and its dissemination to mobile users. Context information is stored in a centralized Context Service in the fixed network, and distributed to strategic proxies in the infrastructure that are close to where a mobile user's access is expected to take place. The Contextual Information Server introduced in [JS03] implements a virtual database abstraction over contextual information that is supplied by infrastructure-based context providers. A rich query language allows mobile clients to access application-specific context information of the virtual database.

The Context-awareness Sub-Structure (CASS) proposed by [FC06] is a middleware for context-aware mobile applications running on mobile devices. CASS implements mechanisms for gathering low-level sensor data from mobile nodes and for deriving higher-level context information from sensor data. In CASS, all data is stored in the fixed network in a centralized database. The authors of [DHH07] describe the Context-aware Service Platform (CASP), a framework to gather context information from heterogeneous sensors. The approach works mainly on the sensors layer according to Figure 2.23.b and focuses on the gathering of sensor data in periodic time intervals. The authors only give vague information about the location where context is stored, which is presumably the introduced Context Storage central component.

In the describe systems, context information is always stored in dedicated infrastructure-based context servers and made available to clients via a single-hop wireless link to the fixed network. As such, no MANET-specific requirements for context management have to be taken into consideration. Specifically, these systems give no answer on how the robustness-related Requirement (2) to (4) in Section 2.4 can be fulfilled.

Pervasive Computing Environments

The authors of [GPZ05] introduce the Service-Oriented Context-aware Middleware (SOCAM) for building context-based applications in pervasive computing environments. SOCAM consists of three layers, namely, the sensing layer, context middleware layer, and the context application layer. Each layer maps to the corresponding layer in Figure 2.23. The paper has a strong focus on context reasoning based on comprehensive context models, which are stored at a central context interpreter in the context middleware layer.

The authors of [PLK05] introduce an architecture of a context-aware service middleware (CASM) which is targeted at the implementation and support of context-aware services. CASM supports applications by acquiring and interpreting low-level context data to derive and disseminate high-level context information. CASM also supports context storage and querying by means of context managers. However, the ubiquitous environment in which CASM is designed for is not specified in sufficient detail, and solutions to how context storage and querying is performed to satisfy corresponding requirements are left open.

Mobile Gaia [CAMCM05] is a middleware for ad-hoc pervasive computing environments. It builds on clusters of devices, where a number of devices are managed as a whole to provide a single entity of interaction. The middleware supports the development of applications based on such device clusters. Each cluster has a central coordinator that manages devices in the vicinity. In the coordinator, a context server that provides context information to the devices that are participating in the cluster can be identified as the context storage component.

All of the preceding systems for pervasive computing environments consider a central node to store context information, which hinders scalability (Requirement (5)). As in the case of mobile computing environments, the papers do not address mobile multihop networks with their specific characteristics. Furthermore, the previously discussed systems are strongly service-oriented and context-dependent rather than context management systems for a broader spectrum of context-based applications.

Mobile Ad-hoc Network Environments

An early context management system for mobile ad-hoc networks is the Ad hoc Context Aware Network (ACAN) presented in [KK02]. It implements a layered approach for the retrieval of raw data from sensors, the preprocessing of data into higher-level context by a context manager, and the provisioning of context data to applications in the ad-hoc network. While ACAN focuses on context-based service discovery, data management issues are secondary. Further, the presumably mobile ad-hoc network assumed in ACAN addresses scalability, but does not consider MANET-specific characteristics and most of the requirements in Section 2.4.

The authors of [SWS+04] present a context-aware middleware for mobile ad-hoc environments, which is based on cooperating realtime sentient objects. Mechanisms related to context management are implemented by the context component framework (Context CF), including sensor data gathering, the derivation of higher-level context information and reasoning based thereon. The proposed approach does, however, not describe explicit context management mechanisms. Rather, it allows the construction of services that follow event-based interactions between neighboring nodes, such as cars in a vehicular ad-hoc network (VANET). Furthermore, no mechanisms are specified that deal with the MANET-specific requirements in Section 2.4.

MoGATU, described in [PJC06], is a data management framework tailored to mobile ad-hoc networks. It implements many aspects of the reference model in Figure 2.23.a, which is provided by the same authors, including data storage and querying. Similar to our model, the framework explicitly considers the specific characteristics of MANETs and assumes that all nodes are peers. The authors also recognize that the data management and communication layer need to be closely integrated, which in our case is addressed by interactions that take place between the storage and routing tier (Figure 2.22). However, MoGATU is fundamentally different in that it follows the approach of local data storage. In that approach, data is retained on a single device and queries must be distributed to all possible information providers. At this point, we have to refer to Section 3.4, which provides a detailed analysis of the communication cost of local data storage (there termed global partitioning). The results show that the approach does not scale in terms of the frequency by which queries are used to access portions of the stored data. Thus MoGATU does not fulfill an essential part of Requirement (5).

The authors of [PPJ+06] introduce a holistic data management framework for mobile ad-hoc networks with a strong focus on providing trustworthy data exchange between peers. Like MoGATU, the approach implements local data storage and therefore does not scale with query frequency. Additionally, the authors assume that nodes move according to mobility models for which the devices' relative movement is negligible. This assumption is, however, only valid for very specific scenarios, like the considered vehicular ad-hoc network (VANET). In contrast, we allow for any mobility pattern and show how large degrees of mobility up to the speed of vehicles in urban environments can be effectively supported.

Finally, the authors of [CYC07] present a framework for context management to support context applications in mobile ad-hoc networks. The proposed framework builds on clusters of nodes, each containing a single mobile context manager that stores context information that it obtains from context providers located in close vicinity. Multiple clusters are interconnected by constructing an explicit topological structure, referred to as Segment-Tree Virtual Network (STVN). This structure is then used to route queries to those context managers that contain relevant data for each query issued by the context-aware applications.

Due to the mobility of nodes, however, the proposed STVN requires explicit reconstruction and maintenance. The authors provide measurement results that confirm the significant overhead which is even greater than a straightforward publish/subscribe approach based on Ad-hoc On-Demand Distance Vector Routing (AODV) that the authors present for comparison. In contrast, the proposed framework in this dissertation does not require any maintenance of structures, since it relies on a rendezvous-based approach between data updates and queries. This is vital also for achieving robustness against low node density and weak topological structures (Requirement (3)) and temporary network partitioning (Requirement (4)).

2.5.2 Core Data Storage and Data Migration

Let us now discuss previous approaches that relate to the storage tier of our framework in Figure 2.22 with respect to the requirements in Section 2.4. Figure 2.24 presents a classification of related data-centric storage approaches. The close relation to LCS is based on the fact that all DCS mechanisms implement fundamental mechanisms in order to store data at particular locations in the network. Based on this classification, we can distinguish between position-based DCS, cell-based DCS, and miscellaneous DCS approaches.

Position-based DCS

Approaches that can be associated with *position-based* data-centric storage employ geometric routing to determine individual storage nodes. A reference position is circumscribed by a packet that visits subsequent nodes located close to that position until a node is visited twice. By definition, this node is deemed to be the storage node (Figure 2.25.a).

Data-centric Storage (DCS) was proposed for wireless sensor networks in [RKY$^+$02, RKS$^+$03, SRK$^+$03]. DCS uses the Greedy Perimeter Stateless Routing (GPSR) [KK00] protocol to locate a storage node, termed home node in DCS, responsible for storing data. The authors extended DCS by Structured Replication (SR-DCS) to load-balance data associated with the same key to multiple home nodes at different locations. To achieve data consistency in the presence of limited node mobility and node failures, DCS employs the Perimeter Refresh Protocol (PRP) to replicate data around a perimeter in addition to the home node.

The authors of [GGC03] propose an extension to DCS which they call Resilient Data-centric Storage (R-DCS). Using replica and monitoring nodes, they optimize knowledge about data availability in different regions and the exchange of data between different cells to provide a certain degree of replication. Similar to DCS, the authors apply GPSR to locate storage nodes and optionally support PRP to partially recover data from departing or failing nodes.

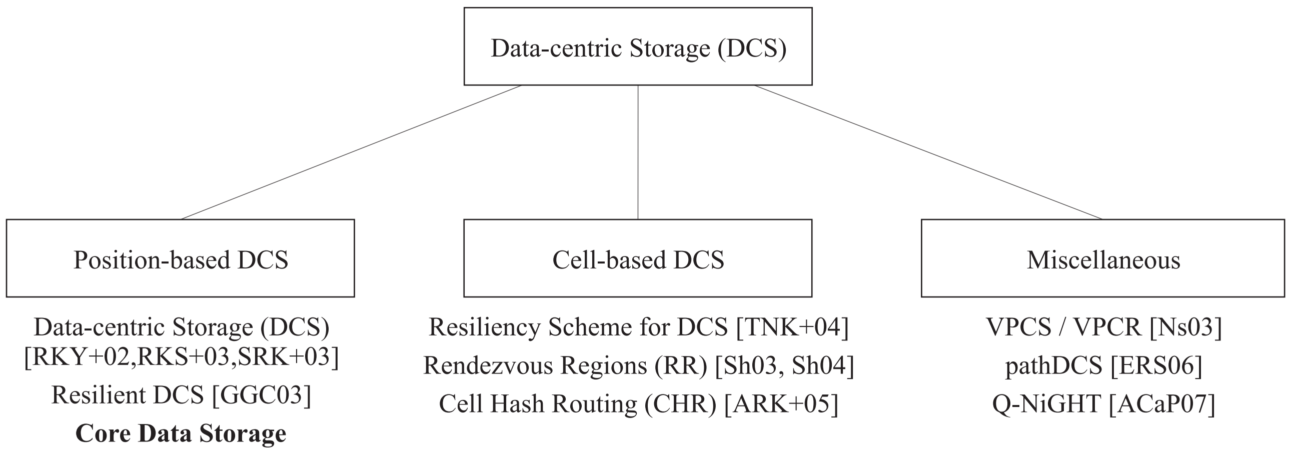

Data-centric Storage (DCS)
Position-based DCS
Cell-based DCS
Miscellaneous
Data-centric Storage (DCS) [RKY+02,RKS+03,SRK+03]
Resilient DCS [GGC03]
Core Data Storage
Resiliency Scheme for DCS [TNK+04]
Rendezvous Regions (RR) [Sh03, Sh04]
Cell Hash Routing (CHR) [ARK+05]
VPCS / VPCR [Ns03]
pathDCS [ERS06]
Q-NiGHT [ACaP07]

Figure 2.24: Classification of mechanisms for data-centric storage.

Besides GPSR, more efficient face routing protocols that can be employed as an alternative were proposed in the literature. Representative approaches include Adaptive Face Routing (AFR) and its extensions GOAFR and GOAFR+ proposed in [KWZ02, KWZ03, KWZZ03] and the Face-aware Routing (FAR) protocol in [HLR04, HBLR05]. To optimize face routing in more practical scenarios where the unit disc graph model does not apply, the Cross-Link Detection Protocol (CLDP) and Greedy Distributed Spanning Tree Routing (GDSTR) were proposed in [KGKS05] and [LLM06], respectively.

Especially in MANETs, DCS and R-DCS incur significant communication cost in the vicinity of storage nodes for each data request and PRP-based refreshing cycle. This is due to the fact that each request and refresh requires the traversal of a closed perimeter, whose length frequently involves a significant number of nodes in comparison to the total route length. While subsequent face routing protocols improve overall routing efficiency and packet delivery ratio, they do not provide mechanisms that would make routing in the vicinity of storage nodes more efficient. The discussed approaches are therefore in opposition to Requirement (1).

In addition to efficiency issues, the presence of malformed perimeters impose challenges on the storage and routing robustness. Figure 2.25.b shows the situation of a malformed perimeter, where a closed perimeter cannot be completed and a packet fails while in transit. This situation occurs for low node densities and weak network topologies (Requirement (3)) and network partitions (Requirement (4)). In the first case, long perimeters lead to timeouts due to the exceeding of the maximum number of hops a packet may travel. In the second case, a closed perimeter around the reference position cannot be traced at all.

All of the previously discussed storage and routing approaches are vulnerable to both situations. Only the authors of [CGP04] propose a customization of geometric routing, termed On-Demand GPSR (OD-GPSR), which targets at the optimization of perimeter traversal in particular. This approach, however, works well only for stationary networks, because OD-GPSR relies on control information that is reused to detect previously traversed perimeters. In mobile environments such control information is of little use due to constant changes of perimeter shapes. Bidirectional Perimeter Routing (BPR), which we introduce in Section 3.2, implements a mechanisms that is resilient to both the situations addressed by Requirement (3) and (4).

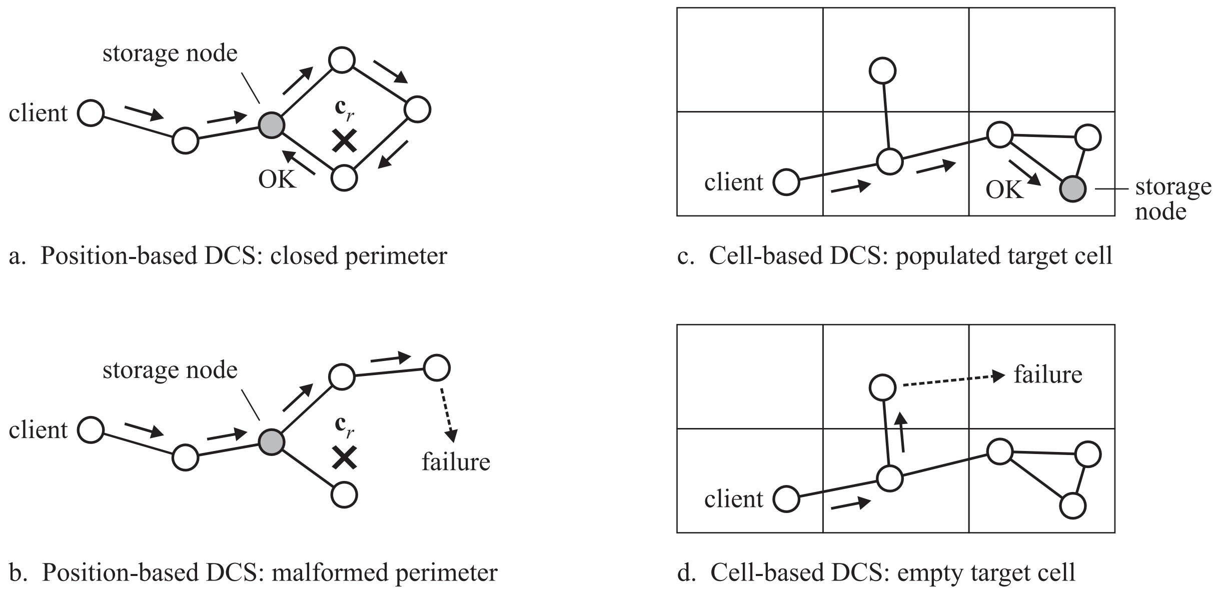

storage node
client
c_r
OK
a. Position-based DCS: closed perimeter
storage node
client
c_r
failure
b. Position-based DCS: malformed perimeter
client
OK
storage node
c. Cell-based DCS: populated target cell
failure
client
d. Cell-based DCS: empty target cell

Figure 2.25: Position-based and cell-based data-centric storage.

Cell-based DCS

Cell-based DCS approaches do not rely on the traversal of perimeters in order to locate a storage node. They instead make use of geometric cells, inside of which a number of nodes are recruited to store data items. Each request is first routed towards a specific destination cell, until it is delivered to one or more nodes based on a local dissemination algorithm (Figure 2.25.c).

The authors of [TNK04] propose a cell-based approach for wireless sensor networks where data items are stored on a single node inside of a cell. The algorithm makes use of multiple destination cells in order to store a data item at diverse locations in the network. This way, the approach increases efficiency on the query side, because each data item can be retrieved from the location closest to the querying node. Rendezvous Regions (RR), proposed in [SH03, SH04], is a very similar cell-based variant for mobile ad-hoc networks. After a data item is routed to a single destination cell, it is flooded to all nodes within that cell. Because all nodes have a copy of each data item, a query only requires to locate any one of the storage nodes.

The authors of [ARK$^+$05] introduce Cell Hash Routing (CHR) for wireless ad-hoc networks, which builds on cells inside of which multiple nodes are recruited for data storage. The authors argue that the particular strategy used to decide which nodes inside of a cell store which data items is a free design choice. However, the authors do not consider node mobility, which has large implications on this choice because the data itself becomes mobile. Furthermore, the authors acknowledge the problem that a cell may not be populated with any nodes. In that case, the approach considers nodes in neighboring cells for storing data.

As noted by the authors of [ARK$^+$05], the population of cells with a sufficient number of nodes is a critical issue for the cell-based approaches. Figure 2.25.d illustrates the situation where a request is destined to the upper right cell, which does not contain any nodes. Both mechanisms in [TNK04] and [SH03, SH04] do not provide a solution for this problem, which leads to request failures in such situations. While [ARK$^+$05] considers this problem, the authors make the decision that a cell is strictly limited in size such that any node inside of one cell can communicate directly with all other nodes in adjacent cells. This is a particularly strong assumption, which works only for the idealized unit disc graph model and the absence of node mobility. Thus, under the conditions of mobile ad-hoc networks, Requirement (3) in Section 2.4 cannot be supported by the aforementioned approaches.

Miscellaneous DCS Approaches

Besides the discussed position-based and cell-based DCS variants, a number of approaches were proposed that apply entirely different strategies in achieving data-centric storage.

The authors of [NS03] introduce a DCS variant for wireless sensor networks that is based on the notion of a Virtual Polar Coordinate Space (VPCS). Different from all other DCS approaches, network nodes are not aware of their physical location. Instead, a virtual polar coordinate system is constructed as an overlay to the network. Virtual Polar Coordinate Routing (VPCR) is then used to route requests to specific nodes that occupy a well-defined location in the VPCS. Similar to [ARK$^+$05], if a node is not available in the supposed location, a more distant alternative node is selected for storing the data item. While the approach also provides recovery mechanisms in case of node failures, the authors acknowledge that it is inefficient for networks with significant node mobility. Hence, VPCS/VPCR is clearly designed for stationary wireless sensor networks and not practical in mobile ad-hoc networks.

An approach to DCS that builds on the abstraction of paths is introduced in [ERS06] and named pathDCS. In this approach, a number of trees are constructed in the network that each root at one of several well-known landmark nodes. Each request is then routed to a specific landmark and then forwarded on a sequence of segments along multiple trees. Similar to PRP in [RKY$^+$02], pathDCS incurs significant overhead in refreshing data in the presence of mobile nodes. Furthermore, pathDCS requires the maintenance of a number of network-wide trees, which incurs significant overhead for dynamic network topologies. The maintenance of such structures, is, in general, not feasible in MANETs, due to potentially low node densities and the risk of network partitioning. In summary, pathDCS is not suitable for MANETs.

The authors of [ACNP07] introduce Q-NiGHT, a DCS mechanism which focuses on QoS control and load balancing. To increase data availability, the approach is based on the concept of a *ball*. A ball is defined as the circular region around a node that contains a specific number of additional nodes. The center node corresponds to the home node as defined in [RKY$^+$02]. A dispersal protocol distributes each data item to the nodes located within this ball. Furthermore, the authors address the issue that perimeter routing according to [KK00] traverses perimeters only in one direction. Q-NiGHT instead dynamically chooses between left-hand and right-hand traversal in dependence of the destination location. The main drawback of Q-NiGHT is that it lacks a suitable protocol to maintain data within a ball in the presence of node mobility. Furthermore, the decision in which direction a parameter is to be traversed is taken before each request and does not solve the problems related to long and open perimeters, as discussed previously. Therefore, Q-NiGHT is unsuitable for mobile ad-hoc networks.

Specific Approaches to Data Migration

In order to maintain spatial coherence according to Definition 2.3, a number of approaches exist that implement explicit approaches to data migration. From the discussed DCS mechanisms, Rendezvous Regions (RR) [SH03, SH04] is the only approach that explicitly maintains data inside of a geographic cell. For that, a dedicated node keeps track of currently active servers inside of a cell. If feedback from one or more servers is missing, an election message is flooded throughout the cell and nodes assume the server role with a given probability. The newly elected servers in turn query for the data stored at previous servers. The main drawback of this cell-based approach is that it is prone to insufficient node population as discussed previously, and thus in violation with Requirement (3).

Several location and query service architectures make use of cell-based structures that require the handover of position information within cells eventually. The Hierarchical Location Service (HLS) by [KFWM04] and the Virtual Home Region-based Distributed Position Service (VDPS) by [Wu05] are closest to our work. In the distributed hash table (DHT) approach proposed in [LGW06] migration of data comes into play when a server reaches some threshold distance, in which case data is transferred to another node. Similar to cell-based structures, the approach assumes sufficient node population so that nodes exist at a certain distance to a reference coordinate. Thus, the aforementioned approaches are vulnerable to Requirement (3).

Data migration based on server abstractions is also used in the realm of geocasting in MANETs. Two representative approaches are the GeoGRID geocasting protocol in [LTLS00] and abiding geocast in [MLS05]. Due to mobility and the requirement to maintain data at a specific geographic location, the authors make use of light-weight migration approaches. However, only a very limited amount of data in the form of routing tables must be transferred. Robustness issues addressing Requirement (2) therefore are insignificant. In particular, [LTLS00] assumes that the data is broadcast within a geometric region, before it is retained at a single node only. This approach is not feasible for larger amounts of data that require a significant number of packets, violating an essential part of Requirement (5).

Common to all mentioned approaches to data migration is the fact that the characteristics of the mobile ad-hoc network are not considered in the selection of migration target nodes. Specifically, if fast-moving nodes are selected, such nodes are likely to depart quickly from the original position, leading to a large number of successive migrations. The data migration approach introduced in Chapter 4 accounts for a number of essential node characteristics to achieve robustness against significant node mobility captured by Requirement (2).

2.5.3 Location Updating and Query Processing

In this section we discuss previous work that is related to the processing of position updates and spatial queries in mobile ad-hoc networks. Due to the dynamics of position information, most of the following approaches implement both update and query algorithms that complement each other to provide the query service. An overview of the discussed approaches, structured by network model, query type, and position model is presented in Table 2.3.

Distributed Object Storage and Location Services

The first group of approaches in Table 2.3 implement object-based queries, which allow to retrieve individual objects by identifer. All of the previously discussed DCS-based approaches in Section 2.5.2, including Data-centric Storage (DCS) [RKY$^+$02], Resilient Data-centric Storage (R-DCS) [GGC03], Rendezvous Regions (RR) [SH04], Cell Hash Routing (CHR) [ARK$^+$05], pathDCS [ERS06], and Q-NiGHT [ACNP07] implement this type of query.

Related Work	**Network Model**	**Query Type**	**Position Model**
Distributed Object Storage for WSNs and MANETs			
DCS [RKY^{+}02],	WSN	object queries	*not applicable*
R-DCS [GGC03]	·	·	·
RR [SH04],	MANET	·	·
GCLP [TV04]	·	·	·
CHR [ARK^{+}05],	WSN	·	·
pathDCS [ERS06],	·	·	·
Q-NiGHT [ACNP07]	·	·	·
Location Services for Mobile Ad-hoc Networks			
GLS [LJC^{+}00],	mobile ad-hoc	position queries	accurate point
RLS [Bha03],	network	·	coordinates
HLS [KFWM04],	·	·	·
LLS [ADM04],	·	·	·
dead reckoning-	·	·	·
based [KD04],	·	·	·
PLS [LCN05],	·	·	·
MLS [FW06]	·	·	·
Spatial Query Processing in Infrastructure-based Networks			
NNQ [RKV95],	centralized	k-NNQ	accurate point
k-NNMP [SR01]	database	·	coordinates
CNN [FSAA01]	·	constrained NNQ	·
RNN [BJKS02]	·	reverse, k-NNQ	·
GNN [PSTM04]	·	group NNQ	·
FNNQ [SIG^{+}04]	federated	k-NNQ	accurate 2D
	databases		locations
Spatial Query Processing in Wireless Sensor Networks			
DIFS [GEG^{+}03, GRS^{+}03]	wireless sensor	1D range queries	accurate point
DIM [LKGH03]	network	multi-dimensional	coordinates
	·	range queries	·
Peer-Tree [DF03]	·	k-NNQ; reverse,	·
	·	constrained NNQ	·
STQP Framework [CNS04]	·	historical range	·
	·	queries	·
KPT [WL04],	·	k-NNQ	·
k-NN [WXL05],	·	·	·
Itinerary-based	·	·	·
[XFLW07, FPL07]			
Spatial Query Processing in Mobile Ad-hoc Networks			
Window queries [XL03]	mobile ad-hoc	range queries	accurate point
DHT-based [ZWS06]	network	·	coordinates
DIKNN [WCCC07]	·	NN queries	·
Probabilistic Spatial	·	range and	probability-based
Query Processing	·	k-NN queries	location model

Table 2.3: Classification of related work in the field of update and query processing.

The Geographic Content Location Protocol (GCLP) [TV04] implements object-based queries by relying on the intersection of object update and query paths. Each update of a specific object is sent into four orthogonal geographic directions in the network and stored at the visited intermediary nodes. In order to request an object, a query is disseminated using the same strategy. Eventually, a query intersects at a node that was updated previously and the object information is returned to the querying node.

The second group of approaches contains representative work on location services for mobile ad-hoc networks, including the Geographic Location Service (GLS) [LJC$^+$00], Randomized Location Service (RLS) [Bha03], Hierarchical Location Service (HLS) [KFWM04]), Locality-aware Location Service (LLS) [ADM04], the dead reckoning-based location service in [KD04], Prediction Location Service (PLS) [LCN05], and MLS location service in [FW06]. All location services target at the efficient provisioning of node position information to other nodes in a mobile ad-hoc network. For that, each location service uses a specific strategy to replicate a node's position information at selected nodes in the network, from which it can be queried by other nodes using an appropriate complementary scheme.

All of the previously mentioned approaches to object-based queries allow objects to be queried based on identifiers only. In contrast, spatial queries retrieve a set of objects based on location information, which requires alternative index structures and query processing algorithms.

Spatial Query Processing in Stationary Networks

Representative work on nearest neighbor queries for infrastructure-based systems is summarized in the third group of Table 2.3. Many variants of nearest neighbor queries were proposed, including the classic k-nearest neighbor query (k-NNQ) found in [RKV95, SR01, BJKS02], constrained NNQ [FSAA01], reverse NNQ [BJKS02], and group NNQ [PSTM04]. Independent of the specific query type, these approaches build on a centralized database, which is fundamentally different from the distributed storage model considered in this dissertation.

More closely related to our work is the system presented in [SIG$^+$04] for processing federated k-nearest neighbor queries (FNNQ) over loosely coupled data sources connected through the Internet. The authors employ a two-phase strategy to optimize the aggregation and achieve fast convergence of query results. The primary difference to our work is the focus on static two-dimensional spatial data, rather than dynamic position information. Furthermore, all of the described approaches in the third group consider a fixed rather than a mobile ad-hoc network, thus occupying a disjoint region in the design space of Figure 1.5.

The fourth group in Table 2.3 contains work on spatial queries in wireless sensor networks. While the Distributed Index for Features in Sensor Networks (DIFS) [GEG$^+$03, GRS$^+$03] and the Distributed Index for Multi-Dimensional Range Queries (DIM) [LKGH03] support range queries in one or more dimensions, the Peer-Tree approach [DF03] is suitable for spatial queries in general. The authors of [CNS04] and [WL04, WXL05] propose algorithms for processing historical range and k-nearest neighbor queries, respectively.

More recently, itinerary-based techniques for processing k-nearest neighbor queries were proposed in [FPL07, XFLW07]. These algorithms work by visiting a sequence of nodes during which the query result is aggregated iteratively based on the computation of specific itineraries. A particular drawback of these approaches is that an itinerary is difficult to trace for low node densities, and impossible to trace in the presence of network partitions. These facts make the approach vulnerable to Requirement (3) and (4). Altogether, the approaches in this group of algorithms have in common that they are designed for stationary wireless sensor networks and thus do not address Requirement (2).

Spatial Query Processing in Mobile Ad-hoc Networks

The fifth group in Table 2.3 contains approaches for processing spatial queries in mobile ad-hoc networks and can be regarded closest to the scope of this dissertation.

The authors of [XL03] propose algorithms for processing spatial window queries in highly mobile sensor networks. Assuming local data storage (Section 2.5.1), a window query is propagated to its target region first, where it is disseminated to all nodes that are located inside of this region. After a query result is aggregated in a distributed way it is returned to the querying node. While the authors do not substantiate their claims with experimental results, the main drawbacks of the proposed approach are, presumably, efficiency and scalability. The reason is that for larger query windows, query dissemination occurs over a large number of nodes. In contrast, our work builds on a set of light-weight data servers, which provide well-defined subsets of data in the network based on which queries are evaluated.

An approach to implementing geographic range queries in mobile ad-hoc networks based on structured distributed hash tables (DHTs) [LCP$^+$05] is pursued in [ZWS06]. The authors assume that Mobile Ad-hoc Pastry (MADPastry) [ZS05, Zah06], an adaptation of the fixed network-based Pastry DHT [RD01], is executing in the network. On top of MADPastry, an implementation of distributed segment trees [ZSLS06] is added to support the processing of range queries. The proposed approach is complex, because it builds on a two-level distributed index structure. On the first level, the DHT requires continuous maintenance to sustain efficient underlay routing in the presence of node mobility. On the second level, the maintenance of the distributed segment tree adds another significant portion to the overall communication overhead. For these reasons, only limited scenarios with small node speed (1.4 m/s) and large transmission range (250 m) are considered. These parameter settings strongly favor the construction of the DHT overlay. In contrast, we address much harsher MANET environments where nodes move up to 15 m/s and a transmission range of only 100 m is assumed.

The authors of [WCCC07] propose an itinerary-based algorithm for processing k-nearest neighbor queries. In contrast to [FPL07, XFLW07], the authors claim that the approach is specifically designed for mobile ad-hoc networks. In terms of mobility, the provided evaluation shows that Requirement (2) is well supported. However, the authors explicitly assume uniform node distribution and sufficient node density, which is not in line with Requirement (3) and (4). This assumption is valid in those WSN scenarios where node placement and movement can be influenced during network deployment. However, in MANETs, the assumption is that nodes move autonomously and thus may lead to any form of network topology.

Consideration of Inaccurate Position Information

All of the discussed approaches in Table 2.3 have in common that position information is considered to be accurate for the processing of position updates and spatial queries. Thus, the approaches do not address Requirement (6), which demands for more general location and query semantics to account for the intrinsic inaccuracies of sensor systems.

In contrast, the consideration of inaccurate data has received much attention in the database community. While these approaches are designed for database systems and thus are not applicable to MANETs, various query semantics have been proposed until recently. Table 2.4 summarizes the approaches we describe in the following and indicates the types of query that each contribution covers by means of probabilistic methods. The table further states whether a threshold probability can be specified to include only objects that obey a given minimum probability in a query result. The last column indicates whether a maximum inaccuracy can be specified for objects that are to be excluded from a query result.

Related Work	**RQ**	**1-NNQ**	**k-NNQ**	**Threshold Probability**	**Maximum Inaccuracy**
[WSCY99]	+	−	−	+	−
[CPK03, CKP04]	+	+	−	−	−
[CKP03]	+	+	−	−	−
[CP03]	+	−	−	+	−
[KKR07]	−	+	−	+	−
[CC07]	+	−	−	+	−
[CCMC08]	−	+	−	+	−
[LC08]		group NNQ		+	−

Table 2.4: Classification of related work in the field of probabilistic query processing.

Wolfson et al. propose a probabilistic model for range queries for moving objects [WSCY99]. A range query returns only those objects that satisfy a minimum threshold probability with respect to being located within a geographic region. The authors of [CPK03, CKP04] consider Probabilistic Range (PRQ) and Nearest Neighbor Queries (PNNQ). Both queries return all objects whose probability either of being located inside of the specified geometric range or of being the nearest object to a specified position is greater than zero.

Cheng et al. propose in [CKP03] general definitions for different types of probabilistic queries over imprecise data. The proposed Probabilistic Range Queries (ERQ) and Nearest Neighbor Queries (ENNQ) are generalizations of the PRQ and PNNQ in [CPK03, CKP04], respectively. In [CP03], Cheng and Prabhakar define generic probabilistic threshold queries. Specifically, the proposed Probabilistic Threshold Range Query (PTRQ) is similar to PRQ in [CPK03, CKP04], but, like [WSCY99], allows to specify a non-zero threshold probability.

The authors of [KKR07] provide a definition for probabilistic nearest neighbor queries. Apart from the uncertainty of the result objects, the authors also assume that the query position itself is inaccurate. In a similar way, Chen and Cheng consider in [CC07] probabilistic range queries that take into account inaccurate reference locations. The provided definitions for Imprecise Location-dependent Range Query over Uncertain Objects (IUQ) and Constrained IUQ (C-IUQ) allow queries to be specified with and without a threshold probability, respectively.

The authors of [CCMC08] propose Constrained Nearest Neighbor Queries (C-PNN) over uncertain data. Apart from a threshold probability, the authors take into account a variance in the threshold. The additional parameter is exploited to provide more efficient query evaluation.

Finally, Probabilistic Group Nearest Neighbor Queries (PGNN) are proposed in [LC08]. The work extends the query definition in [PSTM04] by taking into account position inaccuracies. This query type is, however, different from the ones considered in this dissertation.

In summary, Table 2.4 reveals that none of the proposed definitions supports probabilistic k-nearest neighbor queries. Furthermore, none of the approaches considers that objects shall be filtered out from query results due to their exceeding a specified maximum inaccuracy. This feature is, however, essential for many practical scenarios, because objects with large inaccuracies might not be of any value to the querying user.

2.5.4 Summary of Related Work

In the previous sections, we have analyzed in detail a representative number of approaches that relate to each of the tiers presented in the reference model in Figure 2.22. The assessment in Section 2.5.1 and 2.5.2 show that mobility (Requirement (2)) and robustness (Requirement (3) and (4)) are not sufficiently addressed in combination. Chapter 3 and 4 will introduce in detail the storage tier that addresses these requirements in a combined way.

Section 2.5.3 discussed a large number of approaches that address update and query processing in various system structures. In summary, the approaches for mobile ad-hoc networks are sparse and violate efficiency and scalability (Requirement (1) and (5)). In particular, probabilistic concepts are unavailable for k-nearest neighbor queries, thus Requirement (6) needs further consideration. We will address these issues in Chapter 5.

Chapter 3

Core Data Storage

This chapter elaborates the algorithms that implement the core data storage component according to the storage tier in Figure 2.22. In Section 3.1 we describe more formally the system model that we assume in this chapter and also in Chapter 4. Section 3.2 introduces Bidirectional Perimeter Routing (BPR) that is situated in the routing tier according to Figure 2.22. The core data storage algorithms are presented in Section 3.3.

To analyze the performance of the devised algorithms, we adopt a twofold methodology. In order to understand the performance of core data storage in a broader scope, Section 3.4 presents an analytical study of its communication cost in comparison to a number of more straightforward approaches. A detailed analysis of supplementary performance metrics is presented in the experimental performance analysis of Section 3.5.

3.1 System Model

We consider a mobile ad-hoc network (MANET) deployed inside of a Euclidean 2-space. Let u_i denote network nodes communicating via wireless links, characterized by the nominal transmission range r_{tx}. We assume no specific propagation model, hence the effective transmission range of nodes may vary from r_{tx} with time. Let $\mathbf{r}_i(t)$ denote the trajectory of node u_i. We put no restrictions on how coordinates are obtained, both physical (e.g. WGS84 through GPS [EM99]) or virtual coordinates (e.g. [RPSS03]) are supported by our model.

Let D denote a set of data items $o_i \in D$ stored in the network. Let C denote a set of reference coordinates $\mathbf{c}_r \in C$. By $R \in D \times C$ we denote a relation between data items and reference coordinates, known to all nodes. Relation R induces data subsets $D(\mathbf{c}_r) \subseteq D$, containing data items related to the same reference coordinate. A network node u_i is said to be the data server for data subset $D(\mathbf{c}_r)$ if it is responsible for storing all data items of that subset. A data server may manage several data subsets at the same time.

According to our discussions in Section 2.1.2 we assume that nodes do not fail nor leave the network without explicit notice. Specifically, this allows a data server to take the necessary actions regarding the migration of any stored data prior to logging off the network. We further assume that network partitions may occur at any time, but that the duration of a partition is finite. We assume further that the forming of partitions follows a nondeterministic pattern. In conjunction with finite partition durations, this property implies that any two nodes are eventually able to communicate with each other in the same partition. Finally, we allow messages to be lost, arbitrarily delayed or reordered by the communication system.

Two types of requests are used to access data items that are stored at data servers: updates and queries. Updates are performed by clients with respect to a single data item o_i. An update on data item o_i is considered successful when it is reflected in a write operation on o_i at least at one data server. An update may be lost while being propagated to a data server (best effort), e.g. due to a routing failure, in which case it is considered to have not occurred. Queries are complex requests by clients that lead to read operations on multiple data items at one or more data servers. Processes that update never query data items, and vice versa.

According to the CAP Theorem [Bre00], at most two of the properties of data consistency, system availability, and tolerance to network partitioning can be achieved at any given time [GL02]. Because we explicitly take into account network partitioning, we must therefore decide on a tradeoff between consistency and availability. Under consideration of the many types of conceivable context-based applications that do not require strict data consistency, we decide to relax consistency in favor of availability. Formally, we assume that the storage system obeys eventual consistency, defined according to [TDP+94, GA02, TS06]:

Definition 3.1. A storage system provides *eventual consistency* when it guarantees that if no updates on the data items take place for a long time, any access to any data item will eventually return the most recently updated value of that data item.

Apart from eventual consistency, no further guarantees are given. More concretely, eventual consistency allows that for two consecutive queries for the same data item, the read associated with the first query may occur on a more recent value of the data item than the read that is due to the second query. It is further possible that successive queries that involve the reading of the same data item may not see all updated values of that data item. However, it is not possible that a data item which is successfully updated to a data server is lost at a later point in time, which would otherwise be in violation with Definition 3.1

3.2 Bidirectional Perimeter Routing

In Section 2.5.2 we have discussed how existing approaches to data-centric storage (DCS) are unable to sufficiently support the robustness-related requirements in Section 2.4. We have specifically shown in Figure 2.25 how position-based and cell-based DCS are deemed to fail in situations of low node density and network partitions.

To tackle these shortcomings, we introduce Bidirectional Perimeter Routing (BPR), a customization of geometric routing protocols (Section 2.5.2) that apply face routing while relaying packets towards their destination. BPR adapts perimeter routing in such a way that it allows to deterministically locate practical data servers even in the presence of malformed perimeters. While BPR may be applied to any geometric routing protocol that uses face routing, we take GPSR as an example throughout this chapter and Chapter 4.

When considering the approaches that employ perimeter routing, such as DCS [RKY$^+$02] and Q-NiGHT [ACNP07], we can observe that all approaches make use of only *unidirectional* perimeter traversal. That is, after the perimeter entry node is visited, a packet circumscribes the reference coordinate in only one direction. This rigid method does not allow a packet to exploit potential alternatives that might be given in a perimeter's opposite direction.

Figure 3.1 illustrates the two situations in a mobile ad-hoc network in which malformed perimeters occur. In Figure 3.1.a, reference coordinate $\mathbf{c}_r$ lies inside of a region (sometimes called *void*) that does not contain any network nodes. This may typically occur in urban scenarios, where buildings create obstacles such that voids are formed. In this case, a packet that originates from node u_s is routed to the perimeter entry node u_p first, before is is forwarded along the perimeter around the empty region.

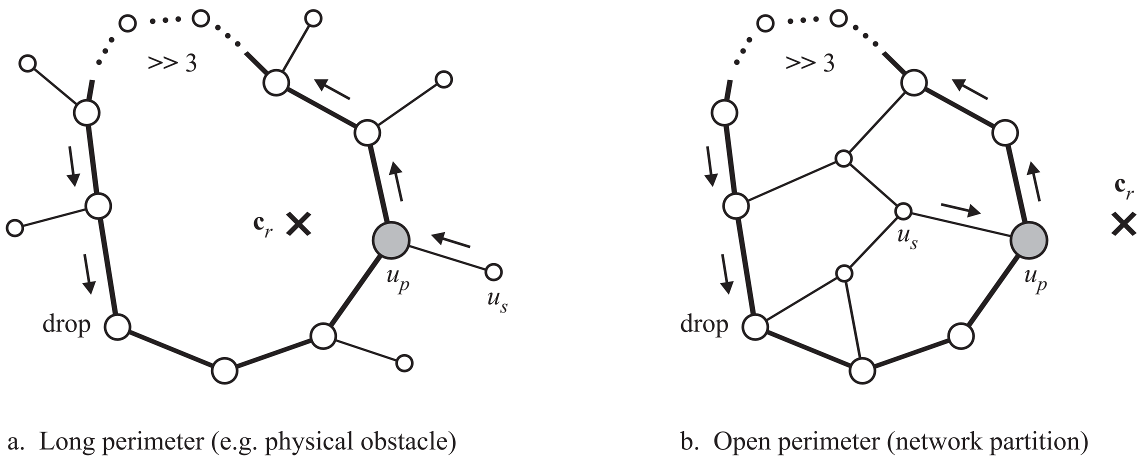
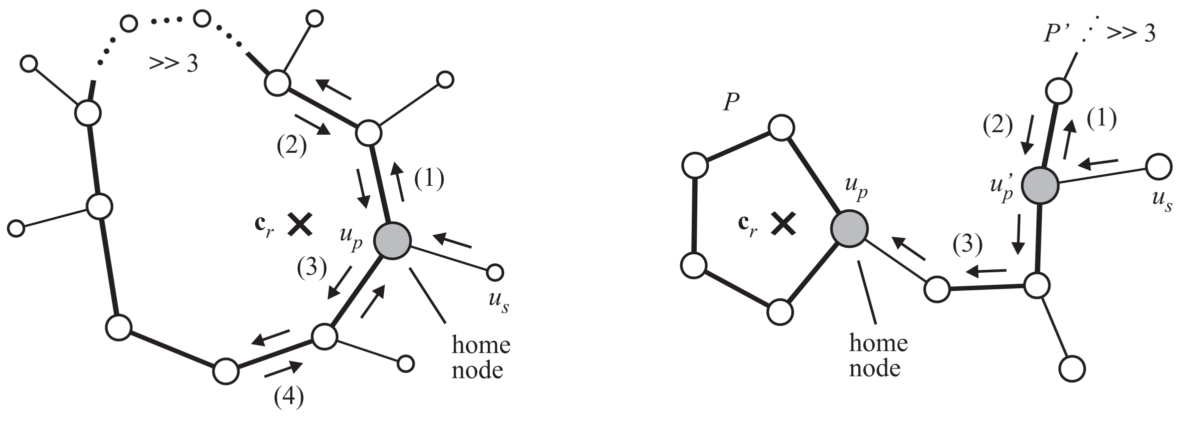

>> 3
$\mathbf{c}_r$
u_p
u_s
drop
a. Long perimeter (e.g. physical obstacle)
>> 3
$\mathbf{c}_r$
u_s
u_p
drop
b. Open perimeter (network partition)

Figure 3.1: Unidirectional perimeter routing in the case of malformed perimeters.

In Figure 3.1.b, $\mathbf{c}_r$ lies outside of the network partition, but the packet originates from node u_s, which is located inside of the partition. This constellation reflects the situation where a temporary partition is formed in the network. As a consequence, the packet reaches perimeter entry node u_p first, then travels on the perimeter that spans the interior of the partition. In both Figure 3.1.a and 3.1.b, if the total perimeter length is greater than the residual time-to-live (TTL) of a packet after starting at u_p, the packet will be dropped before it is able to reach u_p for the second time, which is necessary to decide on the home node.

In order to avoid the forwarding of packets into a single direction that is deemed to fail after a large number of hops, BPR adopts a strategy based on the sequential traversal of a *partial* perimeter in *two* directions. BPR specifies a *perimeter radius* R_p, which defines the maximum number of hops that a perimeter may be traversed in either direction, starting from the perimeter entry node. Thus, the traced partial perimeter comprises at most the set of nodes reachable from the perimeter entry node u_p in R_p hops, taking into account both directions of traversal. This strategy effectively results in the identification of a "practical" home node, rather than a home node that is defined based on a "perfect" full perimeter traversal. The motivation for this strategy is that if a perimeter traversal fails in both directions after the limited number of R_p hops, a network partition exists with high probability.

Figure 3.2.a illustrates how the modified face routing strategy of a geometric routing protocol, such as GPSR, works. Consider source node u_s and reference coordinate $\mathbf{c}_r$. A packet begins to be forwarded in greedy mode, until it is impossible to find consecutive nodes closer to $\mathbf{c}_r$. In this event, BPR switches to *bidirectional* perimeter mode at the perimeter entry node u_p, where a packet is forwarded according to the right-hand rule (1). Each node that is visited on the perimeter is recorded in the packet header. Forwarding continues while each visited node is further away from the reference coordinate than the perimeter entry node u_p and the perimeter radius R_p is not reached. If it is not possible to find a node closer to $\mathbf{c}_r$ in the first direction, the packet is returned to u_p via the previously recorded source route (2). The procedure of perimeter routing is then repeated in the opposite direction (step (3) and (4) in Figure 3.2.a). Again, if a node closer to the reference coordinate than u_p cannot be found, u_p becomes, by definition, the home node with respect to reference coordinate $\mathbf{c}_r$.



a. Convergence in the situation of long perimeter

b. Convergence by exploitation of alternative direction during perimeter traversal

Figure 3.2: Bidirectional perimeter routing in the case of malformed perimeters.

Our choice of using bidirectional perimeter traversal allows, in particular, the consideration of topologies that contain asymmetric perimeters. Figure 3.2.b shows an example of an asymmetric perimeter, denoted P'. BPR starts perimeter traversal at node u_p' in one direction (1) and encounters that P' cannot be completed. After reaching u_p' for the second time (2), the opposite direction is traced. It happens that after a single hop (3), BPR is able to return to greedy mode. Eventually, it reaches perimeter entry node u_p that is related to the closed perimeter around $\mathbf{c}_r$. BPR will terminate at u_p, which becomes the designated home node.

The choice of traversing perimeters in two directions implies that a source route must be taken back to the perimeter entry node before the second direction can be traversed. This redundant visiting of nodes is not present in unidirectional perimeter routing and necessitates a closer look. In Figure 3.3 we have depicted the relation between the communication cost of traversing both unidirectional and bidirectional perimeters for different perimeter shapes.

In Figure 3.3.a, the perimeter length L_P is smaller than or equal to the perimeter radius R_P. In the example, $L_P = 3$, which is the smallest possible length of a perimeter. If $L_P \leq R_P$, BPR and unidirectional perimeter routing behave in the same way, because the perimeter radius is sufficiently large to traverse the full perimeter in one direction. Hence, for $L_P \leq R_P$, BPR and unidirectional perimeter routing incur the same communication cost.

In Figure 3.3.b, L_P is larger than R_P, but smaller than or equal to $2 \cdot R_P$. In this case, the perimeter is traversed in one direction up to the perimeter radius R_P, then travelled back until the perimeter entry node is reached for the second time. In the second direction, BPR must traverse the perimeter only up to the node that terminated the partial perimeter in the opposite direction. The same number of hops must be travelled back to the perimeter entry node. In comparison to unidirectional perimeter routing, BPR incurs significantly larger overhead for $R_P < L_P \leq 2 \cdot R_P$. The total number of hops is $H = 2 \cdot R_P + 2 \cdot (L_P - R_P) = 2 \cdot L_P$, namely, twice as large as for unidirectional perimeter traversal.

In Figure 3.3.c, L_P is larger than $2 \cdot R_P$, thus, BPR must traverse a perimeter in both directions up to the full perimeter radius. In this case, BPR requires the constant overhead of $4 \cdot R_P$ hops, whereas unidirectional perimeter traversal requires L_P hops. Assuming that a perimeter is always closed, the advantage of unidirectional versus bidirectional traversal diminishes with increasing L_P by the difference of $4 \cdot R_P - L_P$. For $L_P = 4 \cdot R_P$, the overhead of BPR and unidirectional perimeter routing breaks even.

The aforementioned considerations indicate that BPR incurs significantly larger overhead than unidirectional perimeter routing for $L_P > R_P$. However, two additional aspects need to be taken into account. Firstly, the arguments regarding Figure 3.3 only apply to closed perimeters. In situations where node density is low, the traversal of a large number of hops may indicate a long or open perimeter. For a malformed perimeter, the *effective* number of hops traversed by unidirectional perimeter routing is limited only by the packet's TTL, which may be large. In contrast, BPR's overhead is limited by the upper bound $4 \cdot R_P$, which is assumed to be significantly smaller than the TTL.

The second aspect pertains to the specific purpose for which BPR is used. In the discussed DCS approaches in Section 2.5.2, every request implies the application of unidirectional perimeter routing in order to determine a home node. In contrast, we will show in the next section how BPR's use of bidirectional perimeter traversal can be strongly reduced by completely decoupling it from request forwarding. We will furthermore revisit BPR's communication cost in Section 3.4, where we will analyze it within the scope of core data storage.

3.3 Core Data Storage Algorithms

Core data storage comprises the mechanisms that allow clients to forward requests to data servers according to the location-centric storage paradigm. In this section, we will discuss the static aspect of location-centric storage. With the mobility of nodes, data migration becomes necessary, which is discussed in detail in Chapter 4.

In order to forward requests, the common strategy in the position-based DCS approaches discussed in Section 2.5.2 is to use a local routing scheme to deterministically select a storage node. For instance, in DCS [RKY$^+$02], GPSR's customized perimeter routing is used to consistently locate a home node. Hence, the identification of a storage node takes place *implicitly*, without the need of node-centric routing. The robustness of node selection, however, depends strongly on the schemes' ability to cope with the characteristics of mobile ad-hoc networks that are addressed by Requirement (3) and (4) in Section 2.4. If a storage node cannot be determined, request accuracy will in turn be adversely affected.

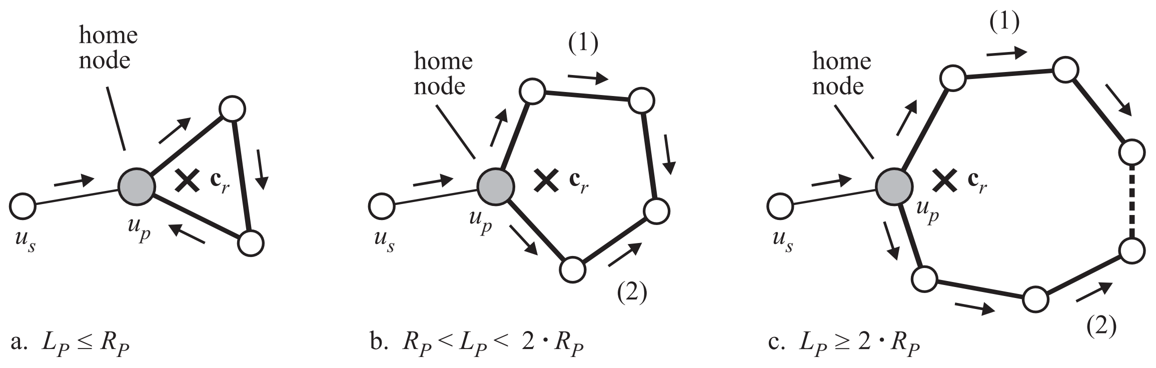


Figure 3.3: BPR: perimeter radius R_P in comparison to perimeter length L_P.

Core data storage acknowledges this inherent drawback of previous DCS approaches and instead employs an *explicit* identification of dedicated data servers. For that, data servers announce their location and association with a reference coordinate via *server advertisement*. Only in this scheme bidirectional perimeter routing is always required to its full extent (Figure 3.3.c). Request forwarding to a data server then takes place in two steps via the distributed advertisement information. The key qualitative difference to previous DCS approaches is that the localization of such information does neither depend on a perfectly deterministic scheme nor on perfectly consistent advertisement information. This design provides the essential flexibility to guarantee compliance with Requirement (3) and (4).

3.3.1 Server Advertisement

Server advertisements are sent by each data server on a regular basis to the server's associated reference coordinate. By regular, we mean that the frequency of server advertisements must be high enough that it remains sufficiently accurate to reliably route packets to a data server. An advertisement contains relevant information to identify currently available servers and their association with particular reference coordinates. That information comprises the server's ID and current position, and the ID of the reference coordinate the server is associated with.

In the first phase, advertisement information is distributed to populate a partial perimeter using bidirectional perimeter routing (BPR). This phase is illustrated in Figure 3.4.a, where a perimeter radius of 1 is used. Hence, nodes u_1, u_2, u_3, which are located on the perimeter that circumscribes $\mathbf{c}_r$, receive the server advertisement. The information contained in a server advertisement is stored at each visited node, together with a lifetime that determines when the information is to be considered stale. Observe that for the distribution of server advertisements, it is not required to return to the perimeter entry node u_2 after the perimeter radius has been reached in the second direction. Furthermore, for small perimeters according to Figure 3.3.a, server advertisements are distributed around the full perimeter.

With the distribution of subsequent server advertisements on a partial perimeter, the mobility of nodes may eventually lead to an alteration in the perimeter's shape. While some nodes will join a perimeter, others will move off to a more distance location. In the case of high node mobility (Requirement (2)), such dynamics may impact the ability to reliably find advertisement information. We provide the following extension to server advertisement for additional resilience to node mobility during request forwarding.

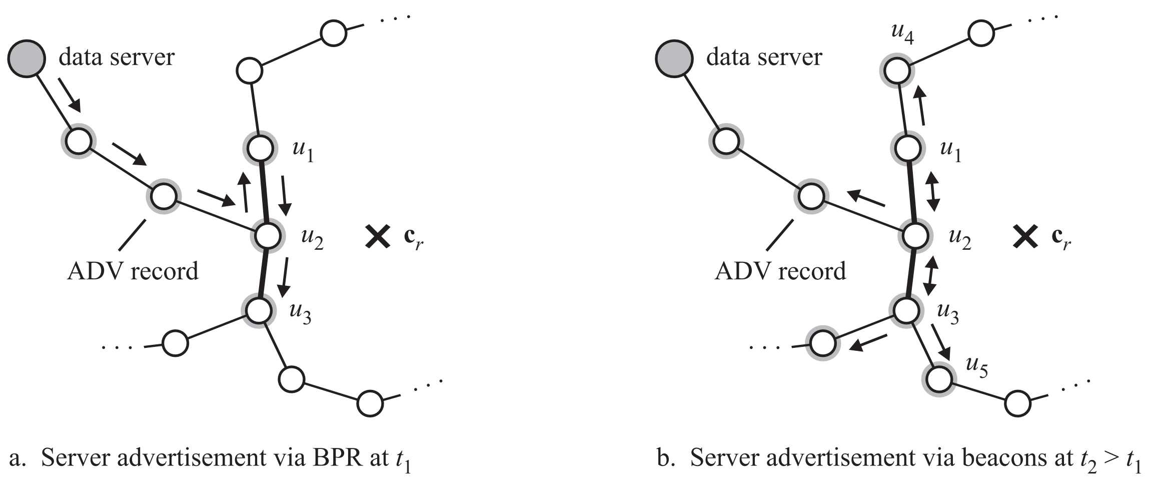


Figure 3.4: Core data storage: server advertisement.

To support the distribution of server advertisements beyond the nodes on the partial perimeter, we exploit existing HELLO beacons that are available in the geometric routing protocol. Hence, no additional packets are required. Two reasons make HELLO beacons particularly attractive. First, nodes send beacons on a regular basis via single broadcasts to inform neighbors about their current position. Second, server advertisements consist of only a small piece of information and are suitable for being piggybacked onto HELLO beacons.

Our approach is to emit a HELLO beacon with server information immediately after a node has received a server advertisement in the first phase. This scheme provides fresh server information to nodes in the 1-hop vicinity of the partial perimeter. Figure 3.4.b shows two nodes u_4, u_5, which have received the information of a server advertisement via a single cycle of HELLO beaconing. Note that the information is stored using a soft state approach and discarded after a timeout period that correlates with the sever advertisement frequency. To avoid the sending of redundant regular beacons, the next beacon is rescheduled after the beaconing interval of the geometric routing protocol. Using a suitable synchronization between server advertisement and regular beacons, *no additional* beacons are required.

3.3.2 Request Forwarding

Client requests, including updates and queries (Section 3.1), are forwarded to data servers in two phases. In the first phase, each client request is forwarded into the direction of the reference coordinate $\mathbf{c}_r$, illustrated in Figure 3.5.a. During this phase, geometric routing is performed towards the reference coordinate in either greedy or perimeter mode, depending on the network topology. For instance, in Figure 3.5.a, the client request reaches a perimeter node u_1 after two hops. This situation is equivalent to the one shown in Figure 3.2.b. After perimeter routing via u_2 and u_3, greedy mode is entered again. At each visited node, a lookup is performed to check whether fresh advertisement information is available for a data server that is associated with the reference coordinate specified in the request.

In the case where fresh server information is found, the request enters the second phase and is forwarded to the data server. In Figure 3.5.a, node u_4 is the first such node. In Figure 3.5.b,

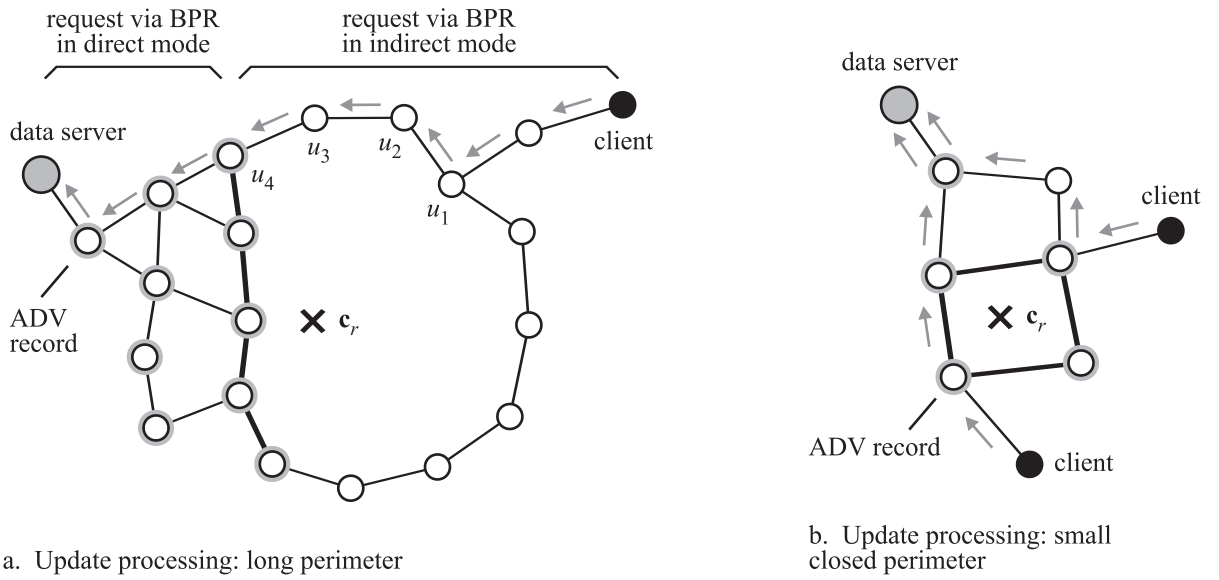

request via BPR in direct mode
request via BPR in indirect mode
data server
client
u_1
u_2
u_3
u_4
$\mathbf{c}_r$
ADV record
a. Update processing: long perimeter
data server
client
$\mathbf{c}_r$
ADV record
client
b. Update processing: small closed perimeter

Figure 3.5: Request forwarding: closed perimeters.

which shows the situation of a small perimeter, advertisement information is found accordingly faster. At this point, the ID and location of the data server are necessary for routing to the server, which can be obtained from the advertisement information. Upon reaching the data server, the request is processed and a reply is sent to the client via geometric routing. The assumption is that the client has previously included its own location in the request, such that the data server is able to return the reply directly to the client.

To emphasize the resilience of core data storage with respect to the robustness Requirement (3) and (4), we illustrate its behavior in the presence of a network partition in Figure 3.6.

In Figure 3.6.a, at t_1, a network partition leads to the situation where no closed perimeter exists around reference coordinate $\mathbf{c}_r$. Because bidirectional perimeter routing is limited by the perimeter radius R_P, the dissemination of server advertisements is restricted to a partial perimeter. For consecutive advertisements, thus, communication cost remain fixed. While

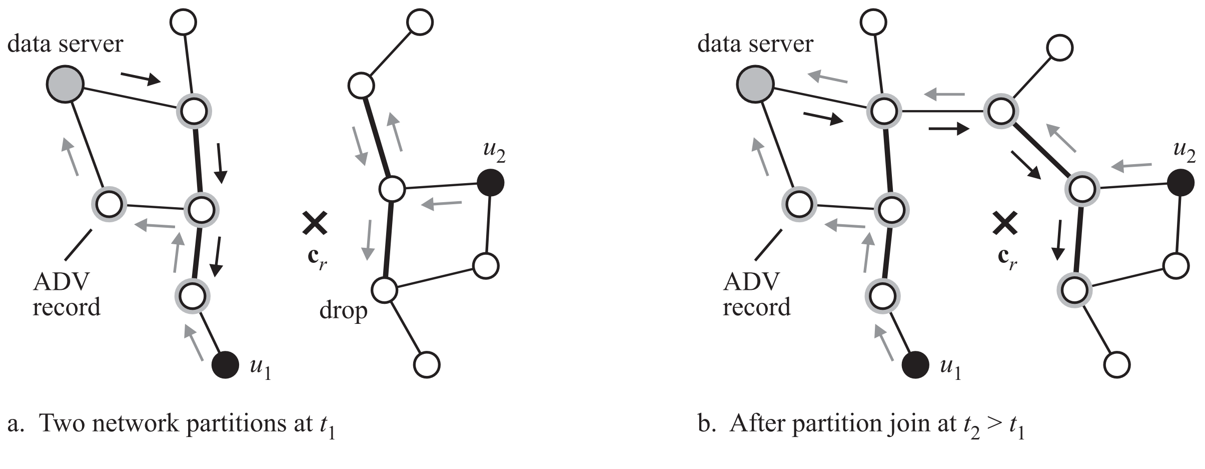


Figure 3.6: Request forwarding: network partition.

clients in the same partition (e.g. u_1 in Figure 3.6.a) are able to locate that information, clients in a different partition (e.g. u_2) are, naturally, not.

Figure 3.6.b shows the joining of the network partitions at $t_2 > t_1$. Observe that the joining of partitions implies the forming of a different partial perimeter along which consecutive server advertisements are distributed. Clients are now able to find server information at a new set of nodes, such as client u_2. In addition, if the previously distributed server information is still considered fresh, it may be used by clients as well (e.g. by client u_1).

In contrast to server advertisements, requests require a full traversal of partial perimeters only in the case of routing failures (e.g. client u_2 in Figure 3.6.a). This essential difference to existing DCS approaches, as discussed in Section 3.2, leads to an overall benefit in terms of communication efficiency, which we will analyze in more detail in the following sections.

3.4 Analytical Study

In this section, we compare the communication cost of the core data storage approach with several alternative approaches that may be considered to implement location-centric storage. Specifically, we justify that despite the indirection scheme used during request forwarding (Section 3.3.2), the overall communication cost incurred by core data storage is generally low in comparison to the other approaches. To this end, we present an analytical study that allows to analyze system parameters on a larger range than simulative methods.

3.4.1 Examined Approaches

A detailed and purely analytical treatment of complex protocols is unfeasible in scenarios with mobility and constantly changing network topologies. Simulative methodologies must be used to treat particular representative scenarios to consider detailed protocol and algorithmic behavior. Nevertheless, analytical considerations with adequate simplifications are useful to discuss a broader range of approaches, which would require significant implementation efforts in a simulation environment. We consider the following storage approaches for a comparative analytical study, whose classification is depicted in Figure 3.7.

In the group of position-based approaches, we can find core data storage, which we denote by LCS/BPR (location-centric storage via Bidirectional Perimeter Routing). Besides server advertisement and request forwarding overhead, we will consider maintenance overhead, which includes data migration overhead that is discussed in Chapter 4. This consideration is necessary to provide a fair comparison to the other approaches. The second approach in the group of position-based approaches is DCS [RKY$^+$02], as described in Section 2.5.2 and abbreviated DCS/GPSR (data-centric storage via Greedy Perimeter Stateless Routing).

In the group of cell-based approaches, we consider three variants. In *cell-based replicated* (C-REP) storage, all data that is associated with the cell is stored on every node inside of that cell. This approach is implemented in Rendezvous Regions [SH03, SH04]. *Cell-based single-copy* (C-SC) storage refers to the situation where all data associated with a cell is stored on a single node. Migration is required to relocate the data once that node leaves the cell. The data-centric storage approach in [TNK04] chooses this variant. In *cell-based partitioning* (C-

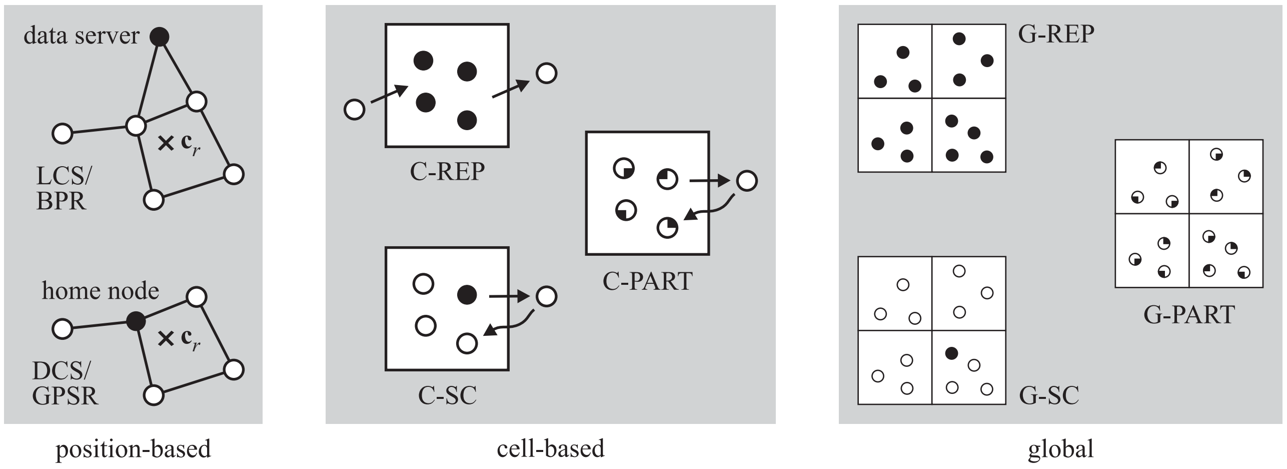


Figure 3.7: Analytical study: classification of considered approaches.

PART) storage, a data item is stored on any node inside of its associated cell. This approach also requires migration of data subsets when nodes leave the cell. This storage mechanism is implemented by Cell Hash Routing in [ARK$^+$05].

We further include the following approaches that make use of global storage strategies. *Global replication* (G-REP) refers to the approach where all data is fully replicated on all nodes in the network. This is used, for example, in [HBR04]. *Global partitioning* (G-PART) denotes the case where data is stored at the node that originally generated the data item. In *global single-copy* (G-SC), all data is stored in a single copy at a single node. G-PART and G-SC are equivalent to local and external storage, respectively, as discussed in the evaluation in [RKS$^+$03].

3.4.2 Analytical Model

We assume a quadratic area A with side length a inside of which the network is deployed. The number of nodes inside of A is denoted by n. Nodes are homogeneously distributed in A and move into uniformly distributed directions. All nodes move at the same and constant speed v. By $S \subseteq A$ we denote the area of a quadratic cell with side length s that is located in the center of A. Cell S contains $n_S = n \cdot (S/A)$ nodes in the average over time.

We assume a spatial data model, hence, each data item is associated with a geometric position. A data item is generated upon observation by a randomly chosen node from the set of n_S nodes and acquires the position of that node. Hence, data items are only generated inside of cell S. Thereafter, an update for the data item is sent according to one of the algorithms in Figure 3.7. The update frequency for all data items is constant and denoted f_u.

Given cell S, we define a data subset of size D_S to comprise all data items whose position is included in S. By $d_s \leq D_S$ we denote the size of a portion of that data subset. We consider an arbitrary query type that retrieves a query result containing a data subset of size d_S. In order to evaluate this query, we require that a range query over S must first retrieve the full data subset of size D_S at a single node. At this node, the arbitrary query is evaluated, before d_S data items are returned in the final query result to the query client. Finally, queries are issued with a frequency of f_q, in units of queries per second.

Notation	Definition
A	quadratic region from which queries originate
$a = \sqrt{A}$	side length of square A
n	mean number of nodes in square A
v	constant speed of all network nodes
$S \subseteq A$	quadratic cell in the center of A
$s = \sqrt{S}$	side length of square S
n_S	mean number of nodes in square S
f_u	update frequency per data item
D_S	total number of data items per data subset
$d_S \leq D_S$	number of queried data items
f_q	query frequency relative to D_S
f_a	server advertisement frequency
C	packet capacity in number of data items
R	migration threshold in LCS/BPR

Table 3.1: Analytical model: notations.

We further assume that the capacity of a single data packet is C, which is the maximum number of data items that can be carried in the packet. For LCS/BPR, we define the migration threshold R. This is the distance from the data server to the associated reference coordinate that triggers data migration. Upon reaching R, the data server migrates all data items it currently stores in its local database. A summary of the used notations is given in Table 3.1.

Mean Communication Cost

We consider the analytical quantification of the communication cost for each of the approaches in Figure 3.7. The mean communication cost, denoted K, that a single data subset incurs, are measured in number of packets per second. K is composed of a number of individual contributions, which depend on the considered approach. Table 3.2 lists the contributions that each approach involves. On the update side, K_u denotes the mean communication cost of data updates in number of packets per second. On the query side, K_q denotes the cost for querying data subsets of size d_S. On the management side, K_m denotes the mean cost for maintaining the data subset of size D_S over time. K_a denotes the cost incurred by server advertisement.

3.4.3 Analytical Derivations

In this section we analytical derive mean communication cost for each of the considered approaches in Figure 3.7. The results will be discussed in Section 3.4.4.

3.4.3.1 Preliminary Results

First, we require some preparatory results to analytically quantify basic communication primitives that are required for our model. First of all, flooding comes at asymptotic communication cost, that is, for k nodes, flooding overhead is $\mathcal{O}(k)$. In the case of simple flooding, which is assumed subsequently, $\mathcal{O}(k)$ corresponds to k packets.

		G-REP	G-PART	G-SC	C-REP	C-PART	G-SC	DCS/GPSR	LCS/BPR
Update Side									
K_u	data update	×	0	×	×	0	×	×	×
Query Side									
K_q	data subset query	0	×	×	×	×	×	×	×
Management Side									
K_m	data maintenance	0	0	0	×	×	×	×	×
K_a	server advertisement	0	0	×/0	0	0	×/0	0	×

Table 3.2: Individual communication cost terms.

Next, we need to quantify the cost for geometric routing. As noted in the beginning of this section, we focus on the operation of any of the considered storage approaches under normal network conditions. That is, we assume a node density where routing succeeds in the large majority of cases, and perimeters are not malformed. Only in this case, geometric routing cost can correctly be assumed to be in the order of $\mathcal{O}(\sqrt{k})$ for k nodes. It is then possible to derive valid correlations between distance and the number of required hops.

Let us first identify the node density range where virtually no malformed perimeters occur. For that, we use Figure 3.8, which displays the fraction of malformed perimeters. Note that the figure uses logarithmic scale on the ordinate. In the figure, a malformed perimeter is defined where a timeout occurs during routing without making a full perimeter traversal. In the figure, we consider area $A = 1200 \cdot 1200$ m^2.

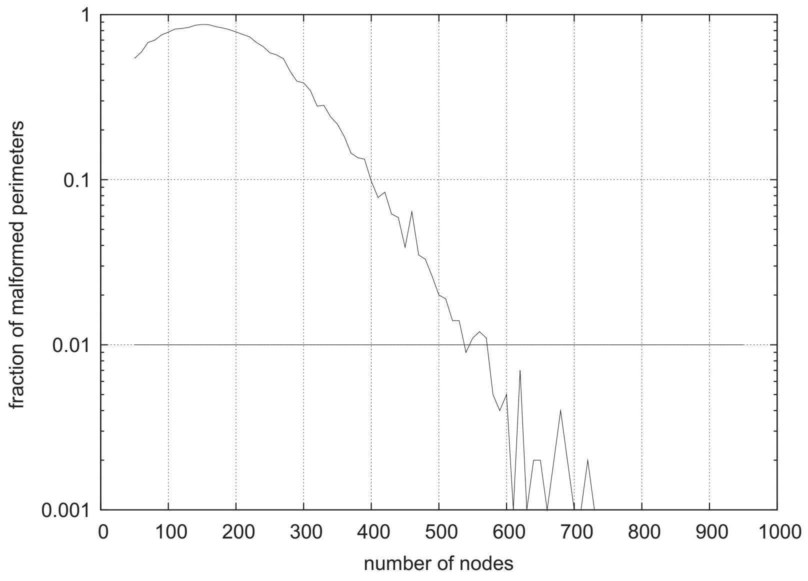


Figure 3.8: Fraction of malformed perimeters.

From the figure, we can identify right away the lower range of node densities from which on malformed perimeters do not play a significant role. We fix the limit where the fraction of malformed perimeters is below one percent. From around 600 nodes and above, the fraction is safely achieved. Thus, we will assume 600 nodes for our reference scenario.

In the second step, we need to establish the relation between network topology and geometry. That is, we need to determine the mean number of hops that are required between a pair of nodes that is separated by a given geometric distance.

Figure 3.9 shows the mean number of hops as a function of the distance between two nodes. Each point captures an interval of 1 m, which is the assumed computational granularity. We have performed a nonlinear least-square (NLLS) fitting, shown along with the data points. In the lower range (approximately below 400 m), there are in fact noncontinuous transitions. Due to the large number of 10^6 iterations, the profile of the curve is sufficiently smooth to be mapped to a continuous function. Above 1000 m, the number of data points for computing the mean becomes too small, because individual distances occur rarer with increasing length. In the range above 1000 m, we therefore assume linear interpolation.

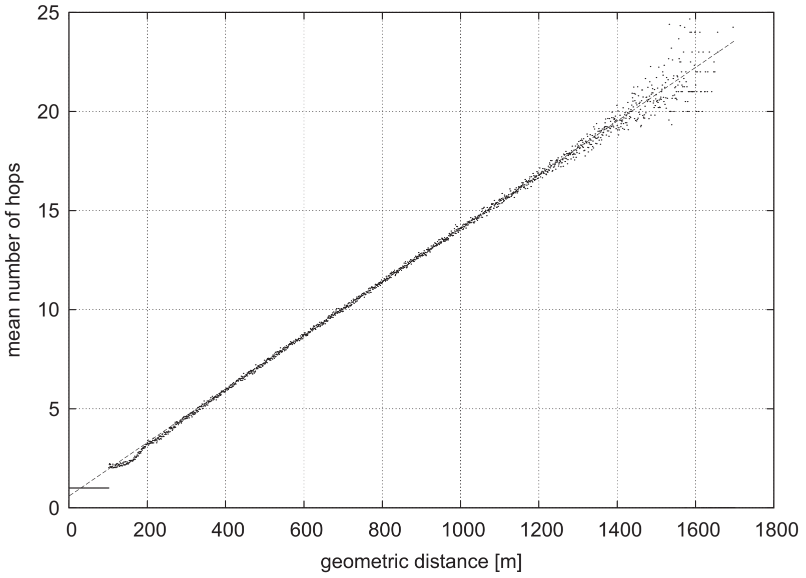


Figure 3.9: Mean route length $H_0(d)$ as a function of the distance d between nodes.

3.4.3.2 Global Approaches

Global Replication (G-REP)

Global replication assumes that each data item of a considered data subset is stored on every node in the network. Consequently, queries can be evaluated on any node immediately. Therefore, only update cost are required (Table 3.2). Since each update must be flooded in the whole network, the mean communication cost for G-REP are:

$$\boxed{K_{\text{G-REP}} = f_u \cdot D_S \cdot n} \tag{3.1}$$

Global Partitioning (G-PART)

In G-PART, each data item of a data subset is only stored at the node that generated the update in the first place. This implies that no cost are required on the update side. Because of global storage, the data items remain within the network at no additional management cost. Because the data of a single data subset is potentially distributed over many nodes, server advertisements do not make sense to locate each node of the subset and are omitted.

We therefore have to consider only the query cost f_q, that is, the cost for requesting a partial data subset with $d_S \leq D_S$ data items. Initially, all nodes that may potentially hold data items that belong to the requested data subset must be identified. For that, we have to consider the speed v of nodes and the update frequency f_u of data items. From the latter, it follows that the time between two updates on the same data item is $1/f_u$. The maximum distance a data item can travel from its original position before the next update is v/f_u.

Because all the nodes of one data subset need to be contacted, the total size of the area S' inside of which nodes may be located that hold relevant data items is

$$S' = \left(\sqrt{S} + \frac{2v}{f_u}\right)^2 \tag{3.2}$$

Based on the assumption that nodes are homogeneously distributed in area A, the number of nodes n', located inside of area S' is

$$n' = n \cdot \frac{S'}{A} \tag{3.3}$$

Efficient querying in terms of overall communication cost works in four steps, where a proxy node is considered that is located in the center of region S'. First, the query is sent from a node inside of region A to the proxy node. In step two, the proxy node distributes the query inside of region S'. In the third step, data items are collected at the proxy node inside of S', and sent back by the proxy to the querying node in step four.

While the second step can be readily quantified by n' packets per query, the other portions require numerical integrations based on the results in Figure 3.8 and 3.9. For that, we will use function H_1 provided in Appendix C.1.1, Figure C.1 (left). The first step requires $H_1(a)$ with $a = \sqrt{A}$ packets (hops). The fourth step, returning the reply from the proxy to the querying node, requires $\lceil d_S/C \rceil \cdot H_1(a)$ packets.

Bringing together the model at the proxy node requires each node to send back its data items that belong to the requested data subset. The total size of the data subset is D_S data items (Table 3.1). We assume that the data items are distributed among the n' nodes homogeneously, that is, the distribution of data items to nodes follows a uniform random distribution. The number of packets required, denoted $d' = d'(n', D_S, C)$, is a function of the number of nodes, the number of data items in the data subset, and the packet capacity C.

Let us first consider a single node. We can state the number of packets required from the viewpoint of a single node, d'', based on the Bernoulli distribution's expectation value:

$$d'' = \sum_{k=0}^{\lceil D_S \rceil} \left\lceil \frac{k}{C} \right\rceil \binom{\lceil D_S \rceil}{k} \frac{1}{n'^k} \left(1 - \frac{1}{n'}\right)^{\lceil D_S \rceil - k} \tag{3.4}$$

The total number of packets that are required in the mean is the product

$$d' = n' \cdot d'' \tag{3.5}$$

Each of the d' packets is then returned to the proxy with cost $H_1(s')$, where $s' = \sqrt{S'}$, thus requiring a total of $d' \cdot H_1(S')$ packets. We can now state the total communication cost for G-PART, which is composed of the four contributions of querying only:

$$\boxed{K_{\text{G-PART}} = f_q \cdot \left[n' + d' \cdot H_1(s') + \left(1 + \left\lceil \frac{d_S}{C} \right\rceil\right) \cdot H_1(a)\right]} \tag{3.6}$$

Global Single-Copy (G-SC)

In G-SC, all model data is stored on a single network node in A. Server advertisements may or may not be used to advertise the location of the server in the network. This choice will influence performance on the query side. We will in the following address both variations.

Let us consider the case without server advertisement first. Each update and each query require flooding at cost n to reach the server, because the location of the server is not available. Since all model data is stored at the server forever, no maintenance cost are required.

Returning the query result depends on the amount of data to be returned and the mean number of hops between the querying node and the server. The amount of data can be quantified as in the case of G-PART, and is equal to $\lceil d_S/C \rceil$ packets. The number of hops requires function H_2, an intermediary result we provide in Appendix C.1.1, Figure C.1 (right).

The mean communication cost for G-SC without advertisement sum up to:

$$\boxed{K_{\text{G-SC}} = f_u \cdot D_S \cdot n + f_q \cdot \left(n + \left\lceil \frac{d_S}{C} \right\rceil \cdot H_2(a)\right)} \tag{3.7}$$

Let us now consider the case with server advertisement, which adds cost of $f_a \cdot n$ for a single server in the network. With the availability of the server location information at every node, the cost for both updating and the first phase of query forwarding can be significantly reduced. Because queries can originate from anywhere inside of region A, each query incurs cost of $H_2(a)$. Updates, however, originate only from within S. For that, we can apply function H_5 from Figure C.3 (left), hence, each update requires $H_5(s)$ packets. The total cost in the case where server advertisement is used are:

$$\boxed{K^{\text{adv}}_{\text{G-SC}} = f_a \cdot n + f_u \cdot D_S \cdot H_5(a) + f_q \cdot \left[\left(1 + \left\lceil \frac{d_S}{C} \right\rceil\right) \cdot H_2(a)\right]} \tag{3.8}$$

3.4.3.3 Cell-based Approaches

In the cell-based storage approaches, some or all nodes inside of the geometric cell of size S, with which the data subset D_S is associated, store parts of the data subset.

Cell-based Replicated (C-REP)

In C-REP, all nodes inside of the cell of size S store the complete data subset. Let us consider update cost first. Based on the association between node position and cell, a node is located inside of the cell where it performs an update. Thus, the update does not have to be routed to the cell, but only needs to be flooded to all nodes inside of the cell. A cell contains n_S nodes in the mean. Thus, the update cost are $f_u \cdot D_S \cdot n_S$.

Maintenance cost are required to retain the data subset inside of the cell. The assumed node population of 600 nodes in A is sufficiently high to assume that new nodes entering the cell will virtually immediately encounter other nodes that are already located inside of the cell. Nodes entering the cell must request a copy of the model from a node already located inside of the cell. Nodes leaving the cell can simply drop their stored data items.

Two magnitudes are required, the amount of data to be transferred and the rate at which nodes enter the cell. The size of a data subset is D_S (Table 3.1). This amount of data must be transferred via a single hop to the entering node, which requires $\lceil D_S/C \rceil$ data packets.

Based on the assumption that movement occurs in every direction at the same probability, and a node is moving in a straight line, we can determine the average time a node is located inside of a cell. By t_{cell} we denote the mean time a node stays within a cell. From (C.8) in Appendix C.1.2 we know the average distance a node will travel in the same cell, which is

$$\bar{l}_1 = -\frac{2\sqrt{2}s}{\pi} \ln\left(\sqrt{2}-1\right) \tag{3.9}$$

From this result, we can immediately derive the magnitude of t_{cell}:

$$t_{\text{cell}} = \frac{\bar{l}_1}{v} \tag{3.10}$$

Conversely, the frequency with which a single node leaves a cell is $1/t_{\text{cell}}$, and the mean cell leave rate for n_S nodes is $n_S \cdot 1/t_{\text{cell}}$. This frequency dictates the rate at which data subsets are to be transferred between nodes. Thus, the maintenance cost for one data subset are

$$\frac{n_S}{t_{\text{cell}}} \cdot \left\lceil \frac{D_S}{C} \right\rceil \tag{3.11}$$

While updates are restricted to within the cell according to our notion of location-centric storage, requests are executed from anywhere in the network area A. Thus, each query must be forwarded to any node inside of the respective cell if the request originates from outside the cell, and the reply is returned via the same route. If a request originates from inside of the cell, no cost are incurred, similar to the case of global replication.

The probability that a node is located inside the cell is S/A. With a probability of $1 - S/A$, thus, the node is located outside and additional routing cost are required. The average of these cost depend on S. We have numerically integrated the mean number of hops required for a node to reach S if it is located in $A \setminus S$. For that, we use function H_3, defined in Appendix C.1.1, Figure C.2 (left). Returning a request is at the same cost, but depending on the requested number of data items a function of d_S. The communication cost for queries sum up to

$$f_q \cdot \left(1 - \frac{S}{A}\right) \cdot \left(1 + \left\lceil \frac{d_S}{C} \right\rceil\right) \cdot H_3(s) \tag{3.12}$$

The total cost for cell-based replication are thus:

$$\boxed{K_{\text{C-REP}} = \frac{n_S}{t_{\text{cell}}} \cdot \left\lceil \frac{D_S}{C} \right\rceil + f_u \cdot D_S \cdot n_S + f_q \cdot \left(1 - \frac{S}{A}\right) \cdot \left(1 + \left\lceil \frac{d_S}{C} \right\rceil\right) \cdot H_3(s)} \tag{3.13}$$

Cell-based Partitioning (C-PART)

In C-PART, all nodes located inside of a cell store the data subset associated with that cell. Different to G-PART, migration is required of the data that a node stores while leaving the cell to another node that is still located inside of the cell. As in the case of global partitioning, we assume that the data subset is homogeneously distributed, this time among the n_S nodes in the respective cell. Therefore, we can state the expected number of packets, $d^{(4)}$, based on the Bernoulli distribution's expectation value again, similar to (3.4):

$$d^{(4)} = \sum_{k=0}^{\lceil D_S \rceil} \left\lceil \frac{k}{C} \right\rceil \binom{\lceil D_S \rceil}{k} \frac{1}{n_S^k} \left(1 - \frac{1}{n_S}\right)^{\lceil D_S \rceil - k} \tag{3.14}$$

From C-REP, We know that nodes leave their cell at the rate of n_S/t_{cell}. Each time that a node leaves the cell the data it stores must be migrated by one hop to any other node inside of cell S. This results in the following migration cost:

$$\frac{n_S}{t_{\text{cell}}} \cdot d^{(4)} \tag{3.15}$$

Note that the data is migrated to a node that is already inside of the cell, thus its mean time to leave the cell is below the global average. However, transferring data back into the cell does not affect the rate at which nodes leave the cell.[1]

As in the case of G-PART, no update cost occur because updates are performed within the cell and occur only locally at the node that generates the update. However, this comes at a price on the query side, similar to the case of G-PART. Again, four contributions add up to the query cost. This time, the second step, query dissemination within S, comes at n_S packets. For the other three portions, we make use of H_1 again. In addition, we take (3.14), to state the number of packets required to transfer the data subset back to a proxy located in the center of S:

$$d''' = n_S \cdot d^{(4)} \tag{3.16}$$

Thus, a total of $d''' \cdot H_1(s)$ packets are required in this third step. Altogether, the request cost plus management cost add up to the mean communication cost for C-PART:

$$\boxed{K_{\text{C-PART}} = \frac{n_S}{t_{\text{cell}}} \cdot d^{(4)} + f_q \cdot \left[n_S + d''' \cdot H_1(s) + \left(1 + \left\lceil \frac{d_S}{C} \right\rceil\right) \cdot H_1(a) \right]} \tag{3.17}$$

[1]Strictly speaking, it is possible that data aggregates towards the border of a cell, and thus is not perfectly homogeneously distributed in a cell. However, the presented calculations are more optimistic and thus lead to smaller total communication cost for C-PART, which is valid for comparison to LCS/BPR.

Cell-based Single-Copy (C-SC)

In C-SC, the complete data subset associated with a cell is stored on a single server inside of that cell. As in the case of G-SC, server advertisements may be used or not, leading to an influence on the query side. We will again examine both alternatives.

We first consider the case without server advertisements. For each update, flooding at cost n_S is required inside of the cell because the location of the single server is unknown.

Migration of the complete data subset is required when the node leaves the cell. For that, we make the idealization (in favor of C-SC) that the target node of a migration will always be located optimally inside of the cell for migration, that is, in the center of the cell. For that, we require the result from (C.12) in Appendix C.1.2, which states the average distance a node will travel to leave a cell, starting from the center of the cell:

$$\bar{l}_2 = \frac{2s}{\pi} \ln(\sqrt{2} + 1) \tag{3.18}$$

At the constant speed v, the mean time it takes to leave the cell is

$$t_{\text{cell2}} = \frac{\bar{l}_2}{v} \tag{3.19}$$

This value determines the frequency of migrations. In addition, we need to quantify the distance, i.e. the number of hops over which migration will be carried out, to the center of S. For that we assume that migration occurs from the border of S to the center, and we can use H_4, shown in Figure C.2 (right). The migration cost sum up to

$$\frac{1}{t_{\text{cell2}}} \cdot \left\lceil \frac{D_S}{C} \right\rceil \cdot H_4(s) \tag{3.20}$$

Requesting a partial data subset requires to locate the single storage node inside of S. For queries originating from outside the cell, the query must first reach the interior of S, as in the case of cell-based replication. Locating the node inside the cell can only be achieved by flooding in the absence of server advertisements. Returning the reply depends on the requested data subset's size. In addition, the number of hops over which the reply is returned can be obtained from H_5, depicted in Figure C.3 (left). The total request cost are

$$f_q \cdot \left[\left(1 - \frac{S}{A}\right) \cdot H_3(s) + n_S + \left\lceil \frac{d_S}{C} \right\rceil \cdot H_5(s) \right] \tag{3.21}$$

Thus, the total costs for C-SC without advertisement are:

$$\boxed{\begin{aligned} K_{\text{C-SC}} = f_u \cdot D_S \cdot n_S + \frac{1}{t_{\text{cell2}}} \cdot \left\lceil \frac{D_S}{C} \right\rceil \cdot H_4(s) \\ + f_q \cdot \left[\left(1 - \frac{S}{A}\right) \cdot H_3(s) + n_S + \left\lceil \frac{d_S}{C} \right\rceil \cdot H_5(s) \right] \end{aligned}} \tag{3.22}$$

Let us now consider the case with advertisements. Then, each advertisement is flooded inside of cell S which requires n_S packets. Like in the case of G-SC, advertisement results in smaller cost of update and query propagation in the cell.

Let us consider updates first, which are generated by nodes inside of the cell only. We use function H_2 to quantify the communication cost for each update sent, because a node is able to immediately route to the sever based on available advertisement information.

In contrast to updates, queries may also origin from the exterior of cell S. For queries that origin inside of S, which is with probability S/A, we can use function H_2. For queries originating from the cell's exterior, we first use H_3 to quantify the number of hops for a request to reach any node inside of A. Then, we use H_6, shown in Figure C.3 (right), which gives the mean number of hops for the request being forwarded inside of S to the server. Thus, a single request incurs cost of $H_3(s)+H_6(s)$. The result of a query is then routed directly to the querying node, which comes at H_5. While a query fits into a single packet, the number of packets required for a query result depend on d_S again. Query cost in the case of advertisement add up to:

$$F_q = f_q \cdot \left[\left(1-\frac{S}{A}\right)\cdot(H_3(s)+H_6(s)) + \frac{S}{A}\cdot H_2(s) + \left\lceil\frac{d_S}{C}\right\rceil \cdot H_5(s)\right] \tag{3.23}$$

Altogether, the total cost for C-SC with advertisement are:

$$\boxed{K^{\text{adv}}_{\text{C-SC}} = f_a \cdot n_S + \frac{1}{t_{\text{cell2}}}\cdot\left\lceil\frac{D_S}{C}\right\rceil \cdot H_4(s) + f_u \cdot D_S \cdot H_2(s) + F_q} \tag{3.24}$$

3.4.3.4 Position-based Approaches

Data-centric Storage (DCS/GPSR)

Data-centric storage according to [RKY⁺02] proposes to locate a node for both queries and updates via the full traversal of a closed perimeter.

The mean length of perimeters that can be expected in a network depends on the topology and the planarization algorithm used. Due to complexity reasons, the analytical derivation is unfeasible, thus, we have assessed the perimeter length by means of measurement. For that, we have produced a large number of topologies with homogeneous node distribution and applied the Gabriel Graph planarization to the network topology. We have then determined the mean length of the perimeter in the center of the region. Figure 3.10 shows the distribution of perimeter lengths in the selected scenario of 600 nodes, measured on an area of $1200 \cdot 1200$ m^2. For a sufficiently small error below one percent, we have performed over 10^6 measurements.

Let $F_p(k)$ denote the function that returns the fraction of occurrences of a particular perimeter length k according to the graph in Figure 3.10.

We begin with determining the update cost. Again, updates occur only within a cell of area S, and we assume that the reference coordinate of DCS/GPSR is located in the center of that cell. Function H_1 gives the mean number of hops required to reach the center of cell S. In addition, the full perimeter needs to be traversed. For that, we must determine the expectation value L_u of the perimeter length based on $F_p(k)$, which is:

$$L_u = \sum_{k=0}^{31} k \cdot F_p(k) \tag{3.25}$$

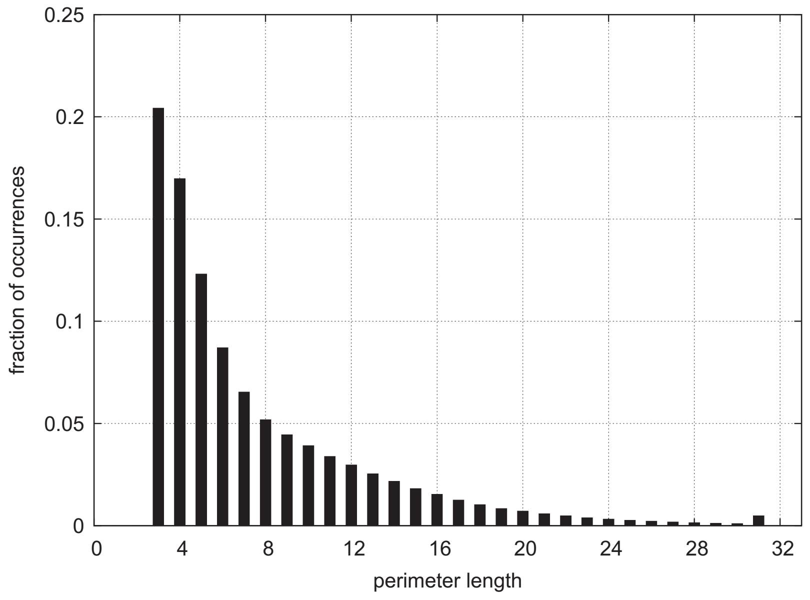


Figure 3.10: Fraction of occurrences of unidirectional perimeters for DCS/GPSR.

The cost for updating thus sums up to

$$K_u = f_u \cdot D_S \cdot (H_1(s) + L_u) \tag{3.26}$$

Queries may originate from anywhere within A. In addition to updates, a reply must also be sent to the querying node, for which H_1 applies as well. Thus, the total query cost sum up to:

$$K_q = f_q \cdot \left[\left(1 + \left\lceil \frac{d_S}{C} \right\rceil \right) \cdot H_1(a) + L_u \right] \tag{3.27}$$

On the management side, DCS/GPSR implements the perimeter refresh protocol to retain data in case of movement of individual nodes. For that, a home node periodically sends refresh messages around the perimeter. According to [RKY+02], a refresh message must contain all data associated with the key, which means in our case the complete data subset. Therefore, management cost occur at the following magnitude:

$$K_m = f_a \cdot L_u \cdot \left\lceil \frac{D_S}{C} \right\rceil \tag{3.28}$$

Note that we have used the advertisement frequency for the frequency of the perimeter refresh protocol, because advertisement and refresh messages both relate to the mobility in the network. The total cost for DCS/GPSR are thus

$$\boxed{\begin{aligned} K_{\text{DCS/GPSR}} &= f_u \cdot D_S \cdot (H_1(s) + L_u) \\ &\quad + f_q \cdot \left[\left(1 + \left\lceil \frac{d_S}{C} \right\rceil \right) \cdot H_1(a) + L_u \right] + f_a \cdot L_u \cdot \left\lceil \frac{D_S}{C} \right\rceil \end{aligned}} \tag{3.29}$$

Location-centric Storage (LCS/BPR)

LCS/BPR has four contributions to the overall communication cost. We will start by update cost again, and we assume that no additional distribution by HELLO beacons is used. Therefore, any node on a perimeter must be located first to find information about the location of the server. This phase can be quantified with H_1 again. Next, geometric routing to the server is required. Because a server will be located within a circle, we can use function H_7 given in Figure C.4 (left) in Appendix C.1.1. Assuming radius R, the number of hops required for a single packet to reach a server from the circle's center is $H_7(R)$. The update cost are

$$K_u = f_u \cdot D_S \cdot (H_1(s) + H_7(R)) \tag{3.30}$$

Next, we consider the cost for a single query, originating in A. We need function H_8 shown in Figure C.4 (right), which quantifies the mean distance between nodes located inside of the circle and square A. Query cost sum up to

$$K_q = f_q \cdot \left(H_1(a) + H_7(R) + H_8(R) \cdot \left\lceil \frac{d_S}{C} \right\rceil \right) \tag{3.31}$$

Regarding the cost for server advertisement, each advertisement is distributed using bidirectional perimeter routing. The cost for a single advertisement depend on the length of the perimeter, as shown in Figure 3.10. Instead of unidirectional traversal, bidirectional perimeter traversal is used. Let R_P denote the perimeter radius setting of BPR. The number of hops required for a particular perimeter length k is given by the following equations:

$$H_b(k, R_P) = \begin{cases} k & \text{if} \quad k \le R_P \\ 2 \cdot R_P + (k - R_P) = k + R_P & \text{if} \quad R_P < k \le 2 \cdot R_P \\ 3 \cdot R_P & \text{if} \quad k > 2 \cdot R_P \end{cases} \tag{3.32}$$

Note that a perimeter is never traced back in the second direction, which is sufficient for a server advertisement. Thus, the expectation value $L_b(R_P)$, which gives the expected mean number of hops for bidirectional perimeter routing for a perimeter radius of R_P, is

$$L_b(R_P) = \sum_{k=0}^{31} H_b(k, R_P) \cdot F_p(k) \tag{3.33}$$

In addition to L_u, each advertisement requires to be sent from the server to the perimeter, which we already quantified by $H_7(R)$. Altogether, server advertisement cost are

$$K_a = f_a \cdot (H_7(R) + L_b(R_P)) \tag{3.34}$$

Finally, we derive the migration cost. For that, we assume that a server is initially located at the center of the circle, thus, using (C.13), the time required for it to leave the circle is

$$t_{\text{cell3}} = \frac{R}{v} \tag{3.35}$$

Further, the number of hops required for the migration of the data is determined by the radius only, and we can use the graph from Figure 3.9, whose function we have denoted by $H_0(d)$ with distance d. Migration incurs the following cost:

$$K_m = \frac{1}{t_{\text{cell3}}} \cdot \left\lceil \frac{D_S}{C} \right\rceil \cdot H_0(R) \tag{3.36}$$

Written in an abbreviated form, the total cost for LCS/BPR are

$$\boxed{K_{\text{LCS/BPR}} = K_u + K_q + K_a + K_m} \tag{3.37}$$

3.4.4 Discussion

Based on the analytical derivations we now discuss the performance of LCS/BPR in relation to the other approaches from Figure 3.7. We use the following default parameter settings: Let $A = 1200 \cdot 1200$ m^2, $S = 400 \cdot 400$ m^2, $n = 600$, and $v = 1.5$ m/s, which corresponds to typical pedestrian speed. Let $D_S = 67$, which results in one data item per node with the aforementioned setting of S. Let $C = 32$, which corresponds to approximately 50 bytes per data item for a packet payload of 1600 bytes. This setting allows to include all necessary information in the data packet, like object identifier and position, and the time of observation. For the migration radius of LCS/BPR, we assume the default value $R = 100$ m.

We further assume the update frequency $f_u = 0.3$, that is, an update on an object occurs every 3.3 seconds. This case reflects an accuracy of 5 m in the position of pedestrians moving at the speed of 1.5 m/s, using a time-based update protocol. The query frequency is $f_q = 10$, which corresponds to one query every 60 seconds per node. The number of data items requested by default corresponds to the size of the data subset, that is $d_S = D_S$. This reflects the case where the complete data subset is requested by a client for further processing.

3.4.4.1 Update and Query Performance Relative to Global Approaches

We begin by examining the relation of LCS/BPR with respect to the global approaches. We will use the default settings and vary the update and query frequency individually. Figure 3.11 and 3.12 display the variation of f_u and f_q, respectively. Note the use of a logarithmic scale, because our analytical model allows us to evaluate the approaches over a wide range of value settings. We also indicate in the following four figures the parameter range where core data storage operates best in comparison to all other approaches.

Both diagrams show the extremal cases where global replication and partitioning have their advantage. In Figure 3.11, the performance of global replication is best when the update frequency is low. However, performance quickly degrades. Based on our analytical model, the intersection with LCS/BPR occurs at about $f_u = 0.007$, which corresponds to about one update every 2.5 minutes. This is well below the degree of dynamics we are interested in.

Figure 3.12 reveals the strength of global partitioning for small values of the query frequency f_q. Above approximately $f_q = 0.5$, LCS/BPR performs better than global partitioning. Note that $f_q = 0.5$ corresponds to one query every two seconds, that is, for 600 nodes, one query

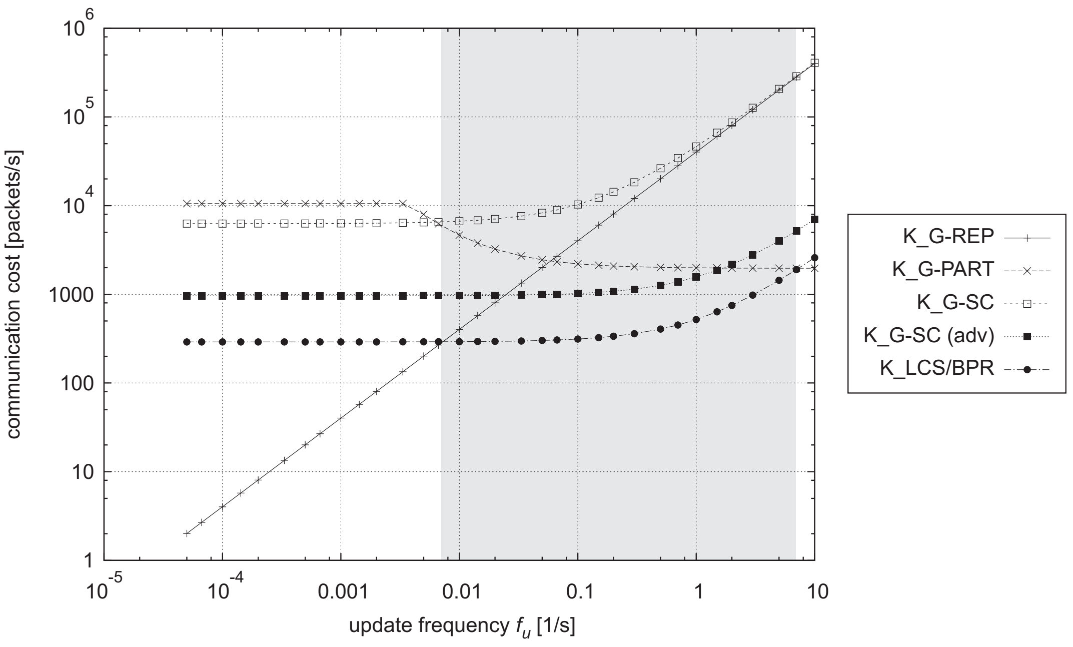


Figure 3.11: Global approaches: variation of the update frequency f_u.

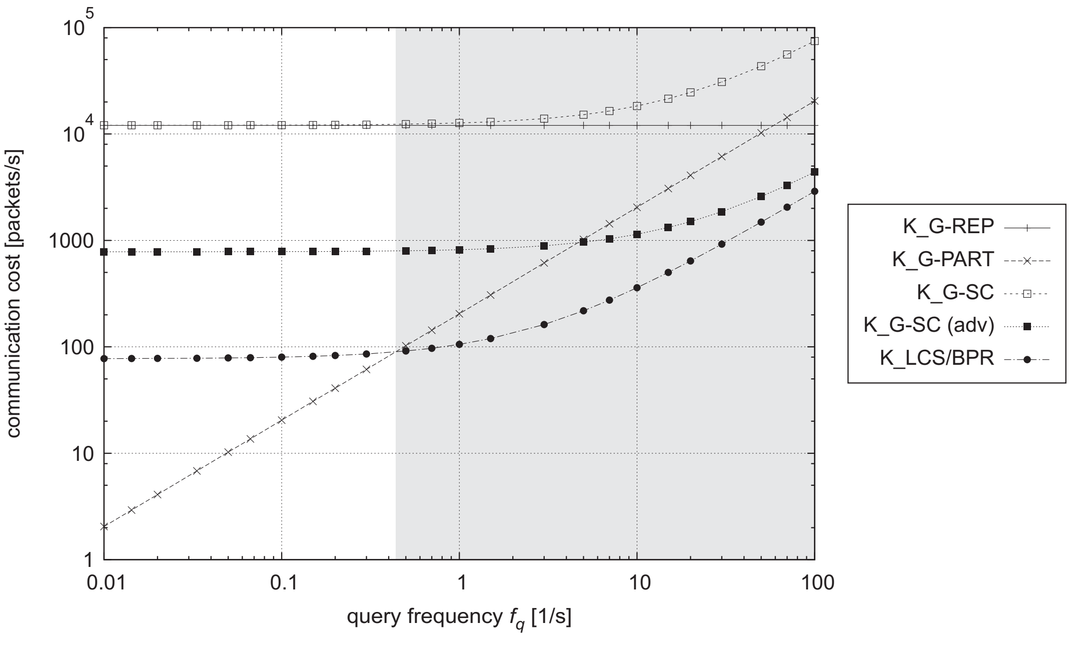


Figure 3.12: Global approaches: variation of the query frequency f_q.

every 20 minutes per node. This is also safely below the dynamics we consider in the scope of location-centric storage. Observe in Figure 3.11 the decreasing slope of global partitioning from about 1/300 and above. The performance is increasing because with higher update frequency, the area in which queries must be resolved decreases.

Global single-copy without advertisement is never adequate, because both updates and queries need to be flooded in the network. With advertisements, global single-copy becomes significantly more efficient, because both routing and update cost are reduced significantly. However, due to the global scope, communication cost are still higher by an order of magnitude, and the approach is outperformed in general by LCS/BPR.

3.4.4.2 Update and Query Performance Relative to Cell-based Approaches

Figure 3.13 and 3.14 show the analogous results for the cell-based approaches as in Figure 3.11 and 3.12, respectively. We can observe that the behavior of cell-based replication and partitioning is qualitatively very much like the global replication and partitioning case, respectively. This is because only the area has changed from A to S, which adds a small amount of management cost (not visible due to its small magnitude).

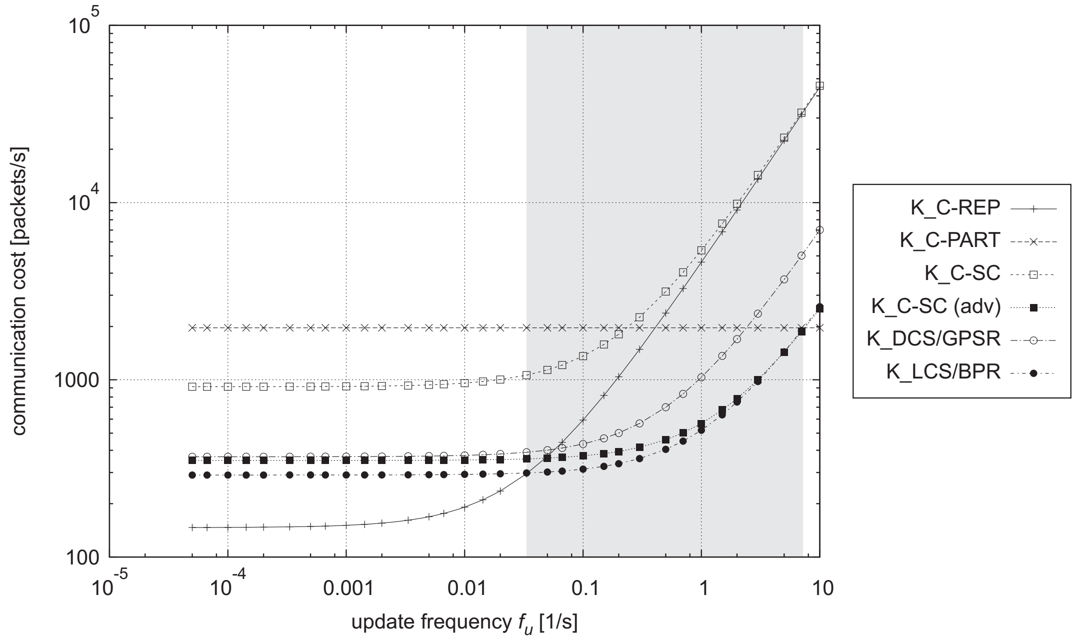


Figure 3.13: Cell-based approaches: variation of the update frequency f_u.

Cell-based single-copy (C-SC) storage without advertisement has again low performance due to the many instances of flooding required for each update and query. In both Figure 3.13 and 3.14, the cost for C-SC without advertisement are larger by one order of magnitude.

In contrast, C-SC with advertisement performs comparatively well with respect to LCS/BPR. This is clear from the fact that this time only advertisements are flooded, and all queries and updates can be routed more efficiently. Because the fraction of cost for advertisements is small compared to the overall cost, both C-SC with advertisement and LCS/BPR perform similar. We will detail on the effects of a variation in the size of S below.

Also observe that LCS/BPR performs significantly better than DCS/GPSR. The reason is that for each update and query, a full perimeter traversal is required. In contrast, LCS/BPR only requires to distribute advertisements on a partial perimeter, and both updates and queries can be efficiently routed to the server even with the indirection in place.

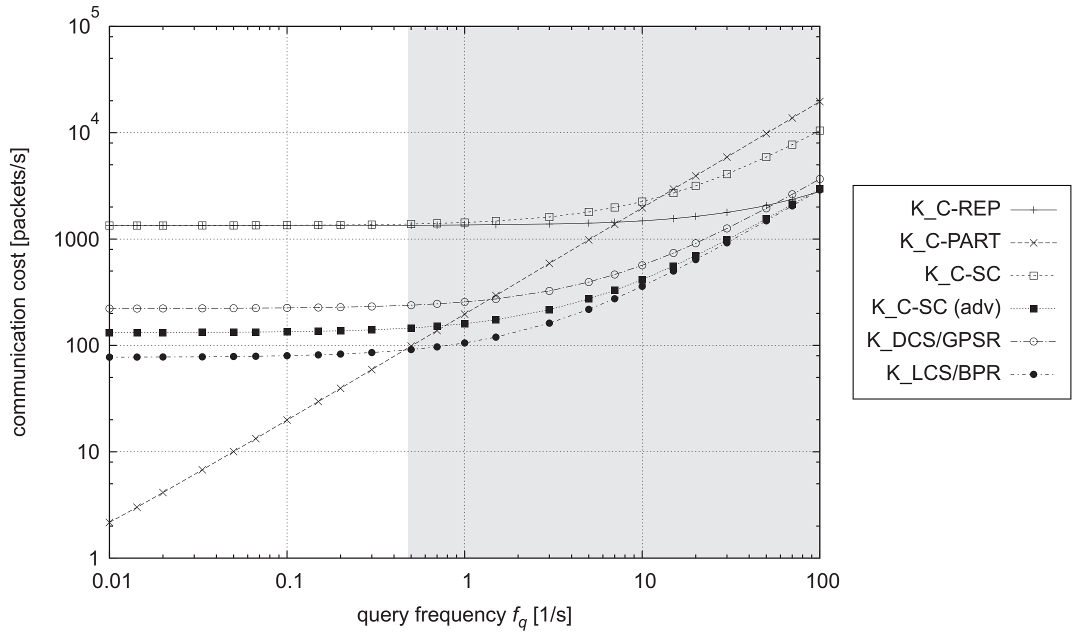


Figure 3.14: Cell-based approaches: variation of the query frequency f_q.

3.4.4.3 Performance of Cell-based Approaches in Idle State

In the following, we restrict our analysis to the cell-based approaches. The idle state is defined as the state where no updates and no queries occur, that is, $f_u = f_q = 0$, and only management cost exist. Due to the fact that model data is retained in a geometric cell in all cell-based approaches, management cost also occur for all of the cell-based approaches.

Figure 3.15 and 3.16 show the mean communication cost in idle state, with a variation of D_S and v, respectively. Let us consider Figure 3.15 first. First of all, we can see that the approaches C-REP, C-PART, and C-SC without advertisements generally incur smaller management cost. However, this fact is responsible for the worse performance in Figure 3.13 and 3.14, because management cost effectively enhance update and query costs. Therefore, management cost are essential to enhance overall performance in the long term. For times of idle state, it is thus important that management cost remain within reasonable limits while they do not bring any benefit. Clearly, this requirement is confirmed by the given results.

Concerning the approaches where management cost imply the highest performance gain relative to C-REP, C-PART and C-SC without advertisement, LCS/BPR remains within reasonable performance limits in idle state. LCS/BPR also performs best compared to DCS/GPSR, which requires to maintain the perimeter refresh protocol, and to C-SC with advertisement, which involves flooding that incurs cost of one additional order of magnitude. Note that 10 packets per second and per data subset are distributed among all nodes in the mean. For 67 nodes being located in cell S, only approximately 1 packet every 7 seconds occurs per node.

Observe the significantly steeper gradient for C-REP and DCS/GPSR in relation to C-SC with advertisements and LCS/BPR. While management cost in idle state are a function of the data subset in all four approaches, C-REP's and DCS/GPSR's idle cost increase much faster because the complete data subset must be either migrated frequently in the case of C-REP, or transmitted with the perimeter refresh protocol in DCS/GPSR. In contrast, both LCS/BPR

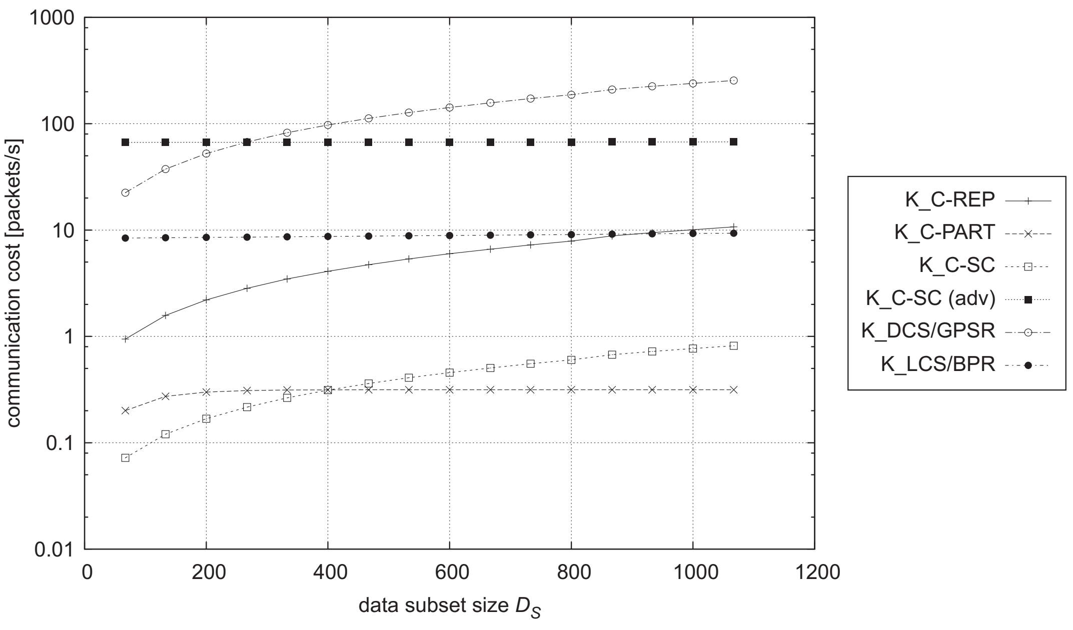


Figure 3.15: Cell-based approaches: idle mode ($f_u = f_q = 0$) with a variation of D_S.

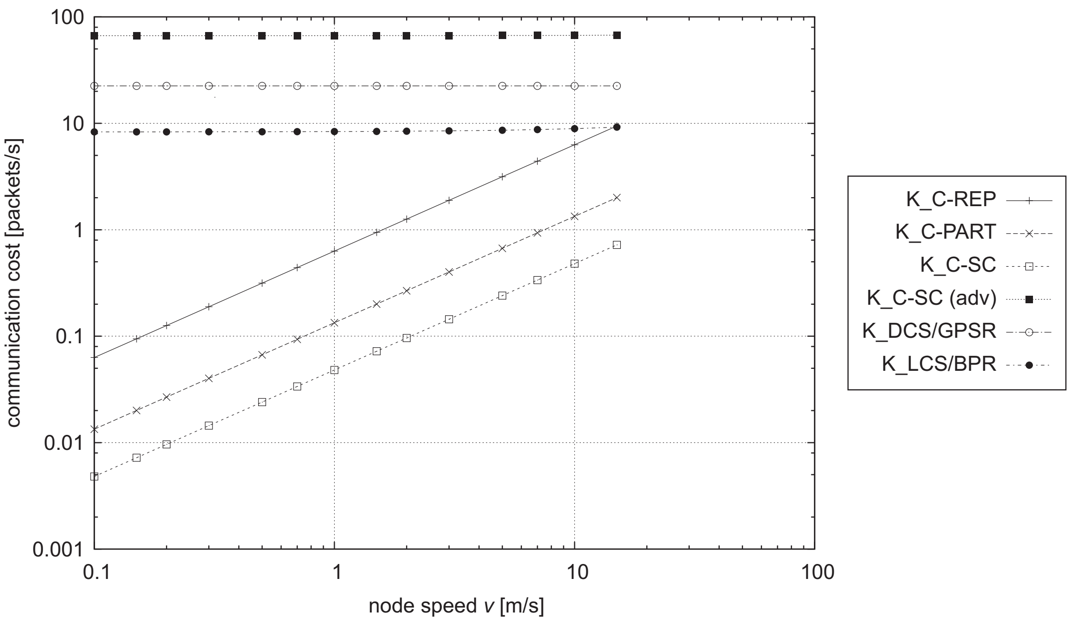


Figure 3.16: Cell-based approaches: idle mode ($f_u = f_q = 0$) with a variation of v.

and C-SC with advertisement only involve occasional migrations of the data subset.

Let us now consider Figure 3.16, which shows the idle cost as a function of the node speed. We can immediately see that a variation of the speed even across several orders of magnitude does not have a significant influence on LCS/BPR, C-SC with advertisement and DCS/GPSR. Still, the influence is highest for LCS/BPR, whose increase can be observed in the upper range at around 10 m/s. The reason is that a relatively small migration radius R leads to a larger number of migrations than in the case of C-SC for $S = 400 \cdot 400$ m^2.

3.4.4.4 Impact of Variation in the Number of Data Items

We will now focus on the variation in the number of data items in a data subset, while keeping S as its default value. This effectively means an increase in the density of data items.

Figure 3.17 and 3.18 show the communication cost as a function of the number of data items in a data subset for the global and cell-based approaches, respectively. In both diagrams, the complete data subset is also returned to the querying node. The figures show that LCS/BPR has the best performance in comparison to the other approaches.

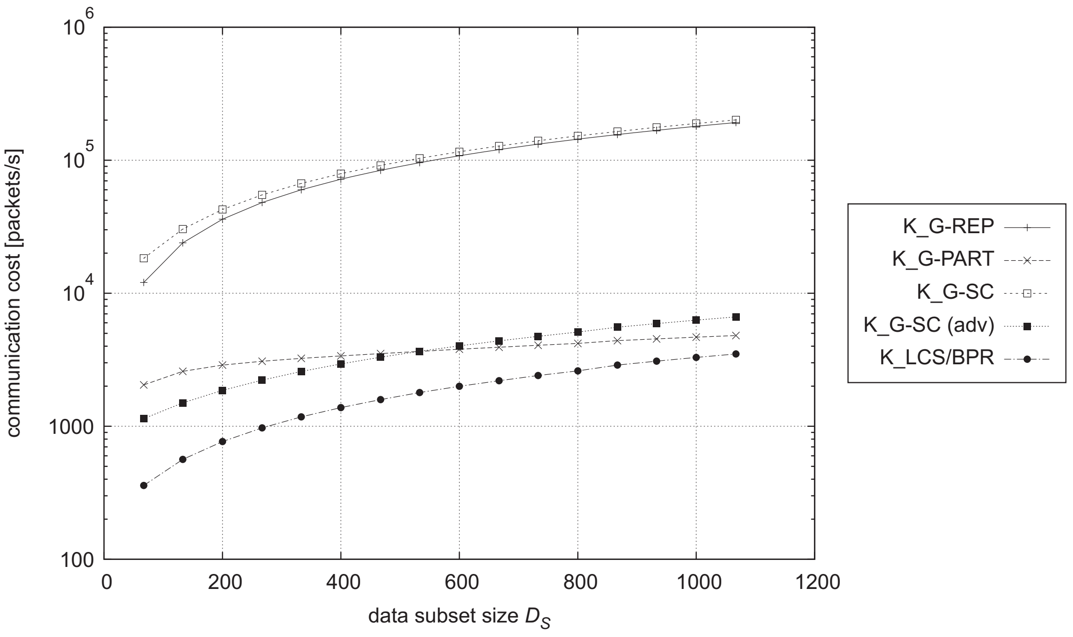


Figure 3.17: Global approaches: variation of the number of data items D_S and returning the full data subset D_S to the querying node.

Figure 3.19 and 3.20 show the results for the same settings as in Figure 3.17 and 3.18, respectively, with the difference that only a single packet is returned in the reply to the querying node. A single packet corresponds to up to 32 data items for a packet capacity of $C = 32$. For larger data subsets, returning only a few data items leads to a significant decrease of the overall cost. This case is particularly relevant for such queries that resolve to a few data items only, such as k-nearest neighbor queries or range queries with small target range.

3.4.4.5 Impact of the Cell Size

The relation of the cell size S to the migration radius R is critical because it decides on the performance gain or penalty for LCS/BPR with varying cell size. In the following, we keep the migration radius of LCS/BPR constant at 100 m. This corresponds to a mean number of 13 nodes within the so-defined circle, leaving a sufficient number of candidates for migration. In the cell-based approaches, no individual migration limit and node selection criteria exist. However, we will elaborate in Chapter 4 that LCS/BPR does not depend on any nodes being located within radius R, while still functioning fine. In contrast, cell-based approaches always require reasonable node population for proper operation.

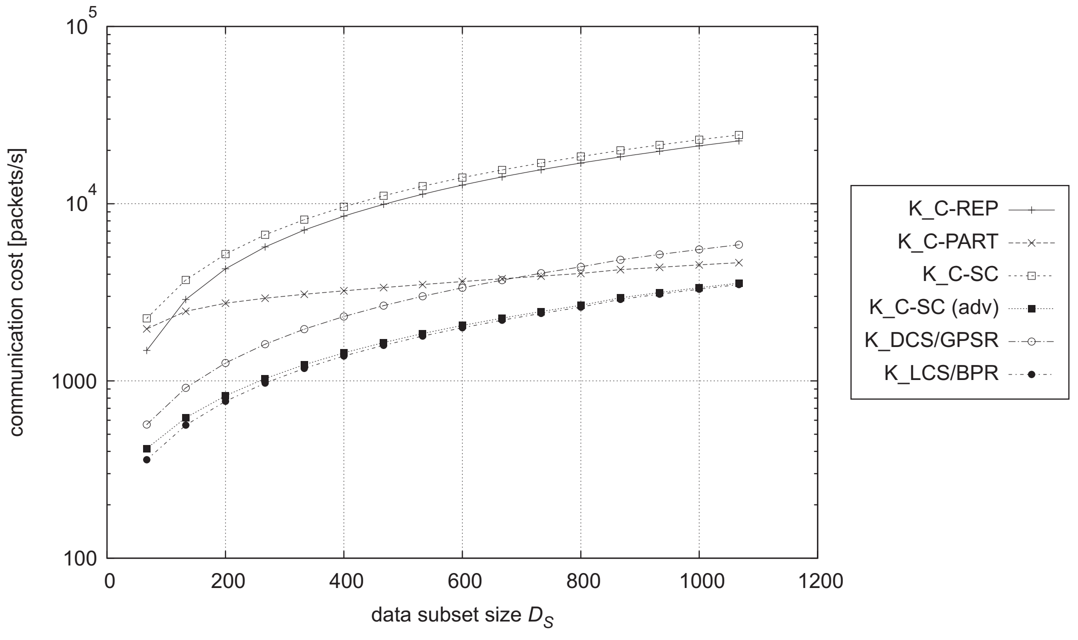


Figure 3.18: Cell-based approaches: variation of the number of data items D_S and returning the full data subset D_S to the querying node.

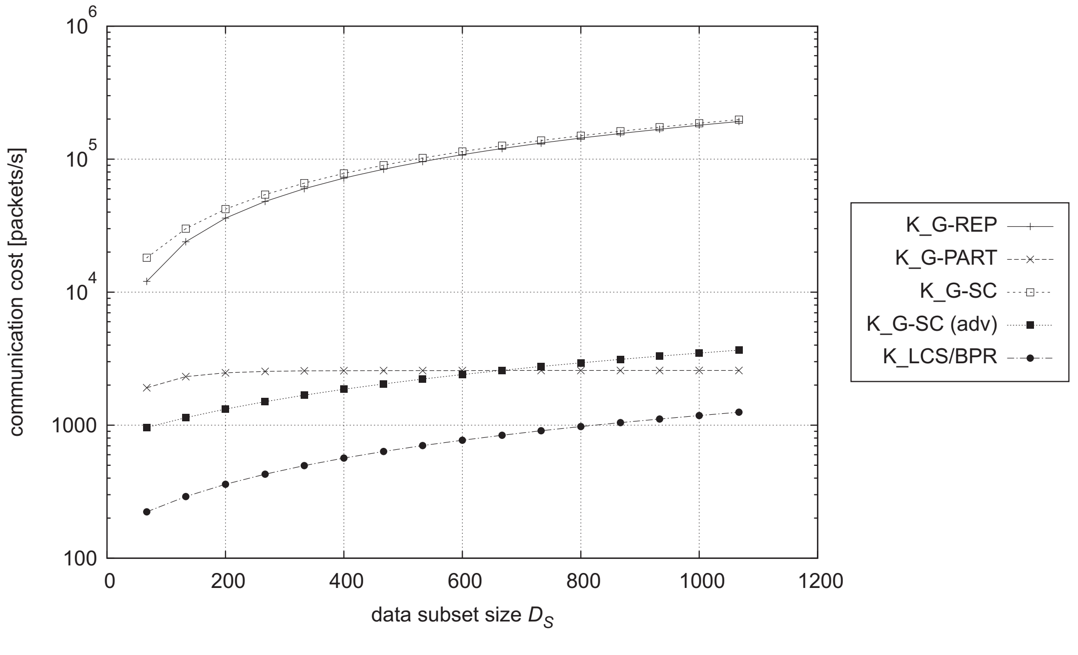


Figure 3.19: Global approaches: variation of the number of data items D_S and returning up to 32 data items in a single packet.

Figure 3.21 and 3.22 show the communication cost for the default setting and a variation of the cell size. Figure 3.21 shows the complete spectrum of cell sizes considered, while Figure 3.22 presents a magnification of the most significant part. Also observe the noncontinuous transition at a cell width of around 275 m. This is due to the packet capacity of $C = 32$ data items, because at this setting more than 32 items are part of a data subset. Note that this shape

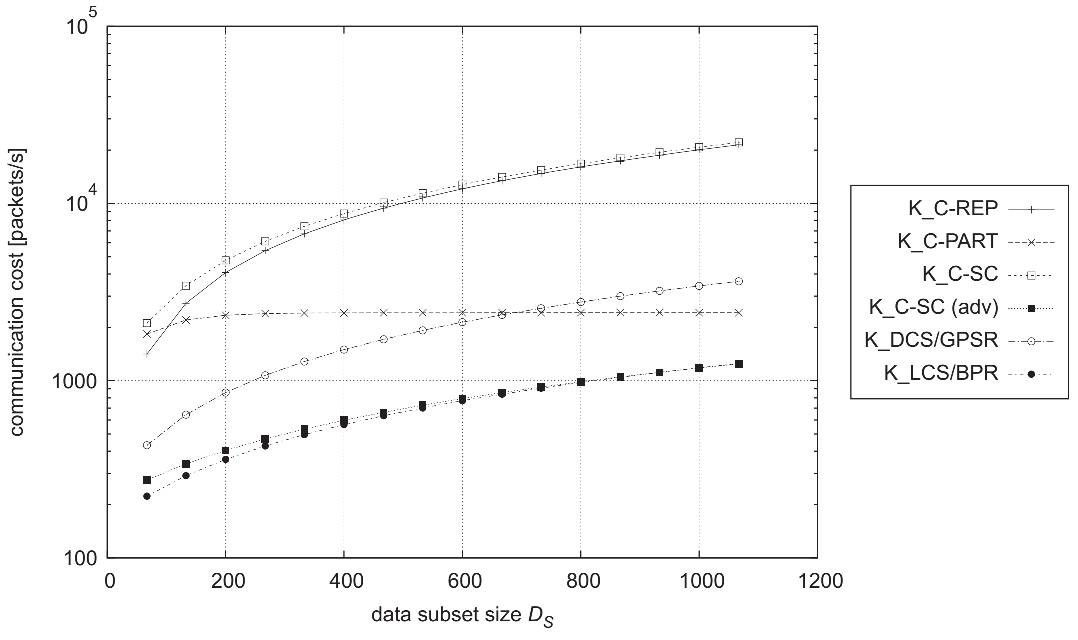


Figure 3.20: Cell-based approaches: variation of the number of data items D_S and returning up to 32 data items in a single packet.

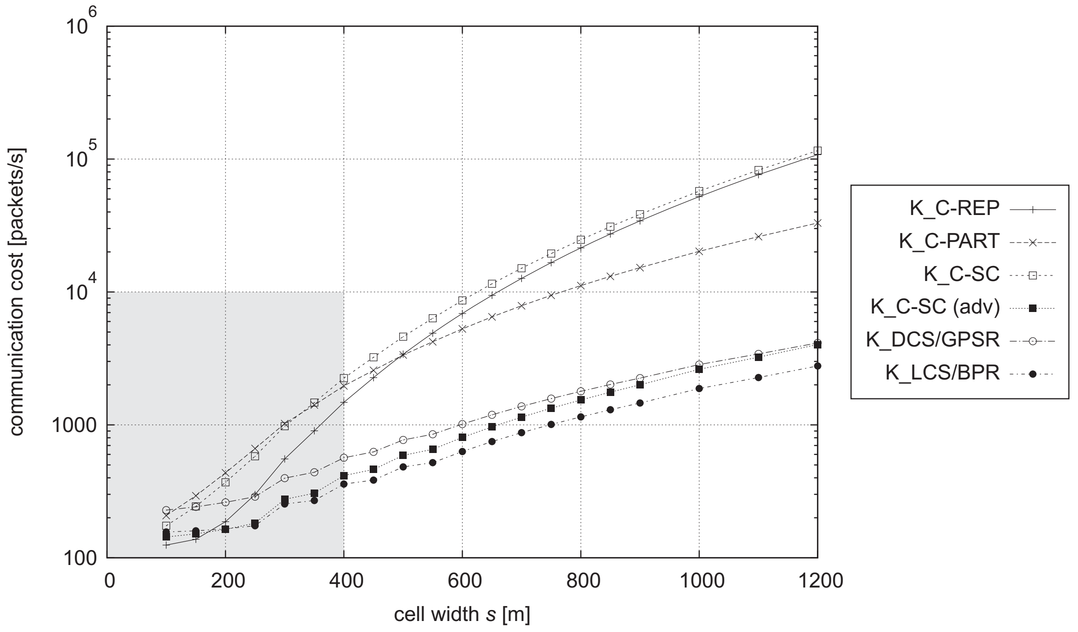


Figure 3.21: Cell-based approaches: variation of the cell size S with default values, in particular, constant query rate f_q across the complete considered range.

appears less distinct for higher communication cost only due to the logarithmic scale.

Let us first consider LCS/BPR in relation to cell-based storage with advertisement. The performance of both approaches according to Figure 3.22 is very similar. This is also true in Figure 3.23 and 3.24. A break-even point can be identified slightly above 200 m. At this point, both the advertisement mechanism's performance and the query/update performance are very

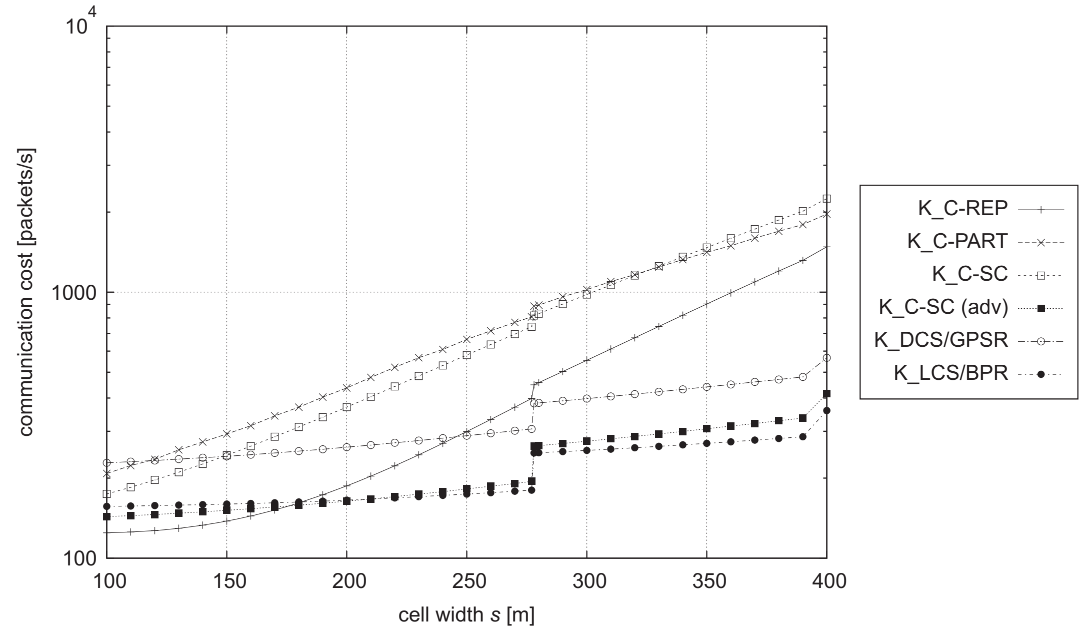


Figure 3.22: Cell-based approaches: variation of the cell size S with default values, in particular, constant query rate f_q in the magnified range.

similar. First, LCS/BPR has better performance above the break-even point. Second, LCS/BPR's performance increases only slowly. These facts make the selection of cell sizes more flexible than for the cell-based storage approach with advertisement.

Let us further examine the intersection between C-REP and LCS/BPR, occurring at approximately 180 m. This is the point where replication is more efficient for the given default settings of $f_u = 0.3$ and $f_q = 10$. However, this advantage of C-REP will rapidly diminish with an increase of f_u, as discussed in conjunction with Figure 3.11 and 3.13.

In Figure 3.21 and 3.22, we have assumed default values for all settings except the single varied parameter. However, we have neglected the more subtle fact that the query frequency per model subset is in general a function of the data subset size, which itself depends on the cell size S. Let us examine this fact in more detail and assume that the size of a cell is decreased. Then, the number of data items managed in the cell also decreases. In the case of queries that require access to a small portion of a data subset only, which will generally be the case in a varied query mix, queries will now be distributed over multiple subsets, naturally leading to a *decrease* in the query rate at an individual subset.

When the query rate decreases per cell the benefit of replication decreases with larger ratios between query rate and update rate. Therefore, deviating from the maximum query rate obtained in the default case, we have illustrated two more variations in Figure 3.23 and 3.24. In Figure 3.23, the query rate f_q is a function of the ratio between cell size S and area size A. That is, for twice the cell width s, the update frequency is quadrupled. In Figure 3.24, the query rate f_q is changed proportionally to the ratio between cell width $s = \sqrt{S}$ and area width $a = \sqrt{A}$. That is, for twice the cell width s, the query rate is doubled.

Both diagrams indicate that LCS/BPR is generally to be preferred over C-REP. Also note

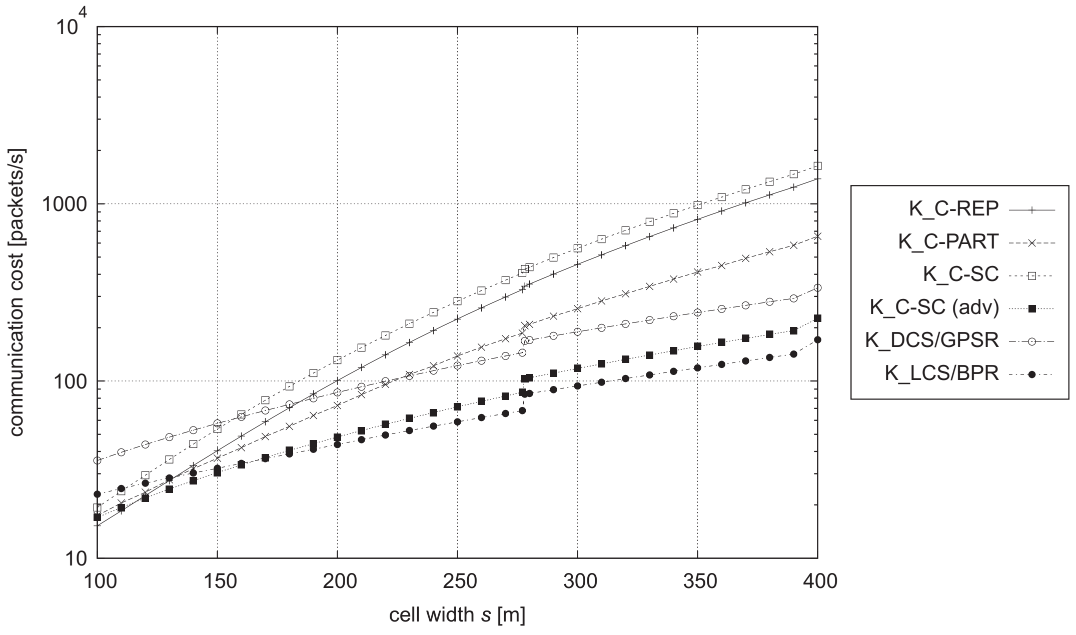


Figure 3.23: Cell-based approaches: variation of the cell size S with default values, but with changing query rate f_q according to a square-root variation.

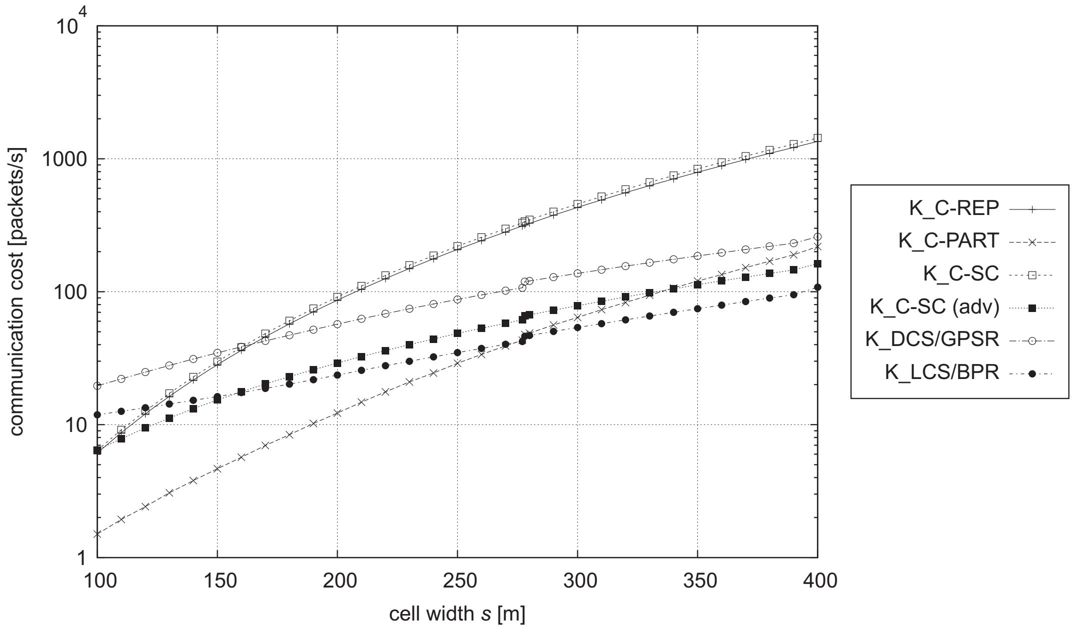


Figure 3.24: Cell-based approaches: variation of the cell size S with default values, but with changing query rate f_q according to a linear variation.

in Figure 3.24 that C-PART is more efficient than LCS/BPR for smaller settings of the cell width. However, we have argued in conjunction with Figure 3.12 and 3.14 that the query rate f_q adversely impacts both the global and cell-based partitioning approaches. Hence, the arguments that apply to C-REP and f_u also hold for C-PART when f_q is changed instead.

3.4.5 Analytical Study: Summary

The analytical study shows that LCS/BPR is an efficient base mechanism in comparison to the considered alternative approaches. Specifically, LCS/BPR's performance remains high for a large range of update frequencies f_u and query frequencies f_q. This behavior makes LCS/BPR very predictable in scenarios where both of these parameters may change significantly over time. At the same time, LCS/BPR incurs only small communication cost in idle state, thus, it does not stress the network in times of low update and query load.

Only the approach of cell-based single-copy (C-SC) with advertisement shows similar performance in terms of communication cost in comparison to LCS/BPR. However, in our study, we have considered a network with a relatively large node population, in order for providing valid basic analytical assumptions. In contrast, for small node densities, cell-based approaches are suboptimal, as we have discussed previously in Section 2.5.2. We will analyze the robustness-related Requirement (2) to (4) in more detail in the next section.

3.5 Performance Analysis

This is the first of three sections whose purpose is to rigorously analyze the performance of the LCS framework in Figure 2.22 using simulative methods. The objective is to give evidence for that the LCS framework implementation provides sufficient performance in terms of the requirements in Section 2.4. On one hand, we investigate in detail the performance characteristics of the algorithms that implement location-centric storage, including core storage (this section) and data migration (Section 4.6). On the other hand, we show the efficient processing of probabilistic spatial queries in Section 5.5 using the concepts of Section 5.3.

An exhaustive performance study that accounts for all conceivable parameters that might have an impact on the various algorithms is unfeasible. This is true especially for the analysis of algorithms in MANETs, where the large space of algorithm parameters is complemented with the many-dimensional space of system parameters. We therefore take particular care in analyzing a subset of parameters that adequately describe the algorithms' key performance characteristics and reveal both their strengths and limitations in an unbiased way.

An additional aspect of algorithm evaluation that is of special importance in MANETs is the fact that both absolute and comparative quantitative performance statements are limited. This is true not only because of simplifications in the model, which every evaluation is subject to, but also because of the significant impact that individual system parameters may have on the magnitude of certain performance metrics. For example, small changes in the network topology or variations in mobility patterns may heavily impact accuracy performance metrics due to the sudden degradation of the underlying routing performance.

We therefore carefully select sensible default values for those parameters that are fixed in each simulation run, in order to provide a solid base configuration to allow for meaningful absolute performance statements. In each simulation, we modify only a single parameter at a time, in order to be able to properly differentiate the performance of multiple algorithms with respect to one another on both a microscopic and macroscopic quantitative scale.

The overall performance analysis is preluded with an overview and justification of the generic evaluation methodology in Section 3.5.1. This generic part of the evaluation will be assumed in the remainder of this section and also in Section 4.6 and 5.5. In Section 3.5.2 and 3.5.3, we discuss the core data storage-specific evaluation methodology and performance metrics, respectively. Section 3.5.4 through 3.5.6 discuss the detailed evaluation results for core data storage. We conclude the evaluation with a brief summary in Section 3.5.7.

3.5.1 Generic Methodology

The analysis of algorithms can be principally accomplished by any of the four methods of real system-based evaluation, emulation, simulation, and mathematical analysis. While we have applied the latter in Section 3.4 to the detailed analysis of communication cost, we have chosen a simulation-based methodology in this section due to the following desired properties.

In order to obtain statistically useful and deterministic evaluation results, a sufficiently large geographic area and number of nodes need to be considered, while at the same time a large number of evaluation runs need to be performed. With respect to these facts real systems are impractical, since they require large setup efforts and runtime overhead, sequential execution of individual evaluation trials, and thus lead to a significant expenditure of time. Real systems furthermore make it virtually impossible to repeat the same experiment under similar conditions, which naturally may lead to inconclusive results.

In order to be able to approximate the performance behavior that algorithms would exhibit in a real system, a comprehensive protocol stack implementation that mimics its real system counterpart is indispensable. Due to the shared nature of wireless media and the extremely high timing requirements in the emulation of shared medium access protocols, emulation environments provide only simplified versions of the MAC sublayer. This fact makes current emulation systems unsuitable for our purposes.

Network topology and node mobility are two of the fundamental system parameters that will significantly impact algorithm performance and therefore must be considered in any mobile ad-hoc network-based evaluation. While real systems and emulation are able to support these parameters, analytical approaches become overwhelmingly complex when attempting to capture these parameters by the analytical model. They may, therefore, only be used to provide a picture under specific constraints and individual performance metrics, which was our goal in the analytical study of the storage tier in Section 3.4.

For these reasons, we have chosen a simulation-based methodology. All evaluations were based on version 2.29 of the ns-2 simulator, using the wireless extensions introduced by Broch et al. in [BMJ+98]. These extensions provide an adequate implementation of the CSMA/CA[2] protocol with the RTS/CTS[3] mechanism according to the IEEE 802.11 standard.

A total of three experiments, corresponding to core data storage, data migration (Section 4.6) and query processing (Section 5.5) were conducted. Each experiment comprises a number of simulations, which depends on the number of system parameters (e.g. node speed, number of nodes) that were modified. All simulations were set up with the following general parameters and their default values, also listed in Table 3.3.

[2]Carrier Sense Multiple Access / Collision Avoidance

[3]Request to Send / Clear to Send

System Parameter	**Default Value**
Simulation runs	10
Simulation area	$600 \cdot 600$ m^2
Number of nodes	150
Mobility model	random waypoint
Node speed (fixed for all nodes)	1.5 m/s
Pause time (fixed for all nodes)	30 s
Communication model	unit disc graph
Communication range	100 m
Propagation model	two-ray ground
Medium bandwidth	11 Mbit/s (IEEE 802.11b)
Beaconing interval	1 s
Beacon information	node identifier and position

Table 3.3: General system parameters for all experiments.

For each simulation, a total of 10 simulation runs was carried out with different mobility and traffic patterns. All performance metrics were taken as the mean over the 10 simulation runs. This number was maintained in all simulations and accounts for a sensible compromise between statistical determinacy and accuracy, and the feasibility to carry out a variety of different simulations in a reasonable amount of time.

All simulations were set up in a geographic region whose size is $600 \cdot 600$ m^2. Inside of this region, 150 nodes were placed that move according to the random waypoint mobility model [BMJ+98]. The mobility model is parameterized with a fixed node speed of 1.5 m/s and a fixed pause time of 30 s, applicable to all nodes. Nodes communicate according to the unit disc graph model, for which a communication range of 100 m is assumed for all nodes. The two-ray ground propagation model applies. The channel bandwidth is set to 11 Mbit/s, equivalent to the specification of the IEEE 802.11b standard. It is further assumed that each node performs MAC-layer beaconing to announce its identifier and current position to neighboring nodes that are within communication range, which occurs in regular time intervals of 1 second.

3.5.2 Core Data Storage: Methodology

We make use of comparative performance analysis to discuss the performance of core data storage with two other approaches. The core data storage approach is considered in the form described in Section 3.3, based on location-centric storage over Bidirectional Perimeter Routing (LCS/BPR). We compare LCS/BPR with a variant thereof, which we term LCS over geocast (LCS/geocast). The third approach is Data-centric Storage over Greedy Perimeter Stateless Routing (DCS/GPSR), as proposed by Ratnasamy et al. in [RKY+02].

LCS/geocast corresponds to cell-based single-copy (C-SC), introduced in the analytical study of Section 3.4.3.3. It differs from LCS/BPR in the type of mechanism used to disseminate server advertisements. In LCS/geocast, a server employs simple flooding to distribute advertisements inside of the cell that is associated with that server via a reference coordinate. Requests are routed the same way as in LCS/BPR, however, during routing, server advertisement information is likely to be found on the first node encountered inside of the respective cell.

DCS/GPSR is implemented according to the approach in [RKY+02] and was considered analytically in Section 3.4.3.4. A data server is implicitly defined as the node being located nearest to the reference coordinate. We recall that GPSR achieves the selection of this node via a full perimeter traversal around the reference coordinate. The same node is reached by requests that are forwarded using GPSR. To account for mobility, the Perimeter Refresh Protocol (PRP) is applied to retain data on the nearest node to the respective reference coordinate.

Table 3.4 lists the default system parameters that are used in the evaluation of the core data storage algorithms in addition to the general system parameters given in Table 3.3.

System Parameter	**Default Value**
Simulation time	360 s
Number of cells	1, centered at (x,y) = (300,300)
Cell size	$200 \cdot 200$ m^2
Number of requests	1000
Request execution interval	$[30\text{ s}, 330\text{ s}]$
Request frequency	3.33 requests/s
BPR perimeter radius	3
LCS server advertisement interval	1 s
PRP refresh interval	1 s
Number of data items	1000
Total data size	32 kB (32 bytes / data item)

Table 3.4: Core data storage: system parameters.

The total simulation time is set to 360 s, which is sufficient to process a large number of requests in order to provide adequate statistical stability. To show the performance of core data storage on a large range of cell sizes, we limit the setup to a single cell that is located at the center (300,300) of the overall simulation area. We set the default cell size to $200 \cdot 200$ m^2, which corresponds to a cell width that is equal to twice the communication range.

By default, 1000 requests are issued during the interval $[30\text{ s}, 330\text{ s}]$ of the simulation time. Requests are executed in regular time intervals, resulting in 3.33 requests per second. For each request, one node is chosen randomly from all nodes located in the single cell based on a uniform distribution. The initial delay of 30 seconds guarantees that the system is in a state of stability regarding mobility patterns and the distribution of initial server advertisements. The delay at the end of the simulation takes into consideration that requests might still be processed and are able to be completed before the simulation ends.

For both variants of LCS, the perimeter radius is set to 3, which is in particular applicable to the way server advertisements are distributed in LCS/BPR. In both LCS/BPR and LCS/geocast, the server advertisement interval is constant at 1 second. To allow for a fair comparison, the refresh interval of DCS/GPSR's PRP is also set to one second.

We further assume that a default fixed number of 1000 data items is stored on the single data server that is associated with the only reference coordinate. Each data item has a payload of 32 bytes, resulting in a total of 32 kB of data. This parameter is relevant due to the fact that core data storage must run with data migration in place. Thus, this amount of data corresponds to the data that is transferred during a migration process. A precise analysis of the impact of data size on migration performance follows in Section 4.6.

3.5.3 Performance Metrics

The performance metrics considered in the evaluation of the core data storage approach are request accuracy, request latency, and request cost.

Request Accuracy

For LCS/BPR and LCS/geocast, *request accuracy* is defined as the fraction of requests successfully delivered from the node that issued the request to the current data server. In the case of a migration in progress, it is not essential whether the migrating node or the target node is reached by the requests, since this situation is handled by the migration process.

For DCS/GPSR, request accuracy is defined as the fraction of requests successfully delivered to the perimeter entry node of the perimeter that encloses the reference coordinate. We assume it is sufficient that a closed perimeter around the reference coordinate can be traced, independent of whether the respective perimeter entry node is up to date according to the PRP. This simplification is in favor of DCS/GPSR and provides an upper bound for its performance, thus, it can validly be assumed in the comparative performance analysis.

Request Latency

Request latency is defined as the mean time between the sending of a request at the requesting node and the reception of that request at the target node, taken over all successful requests. More specifically, for both LCS variants, only requests that are delivered to the current data server are considered. For DCS/GPSR, only those requests that are delivered to the perimeter entry node after the successful traversal of a closed perimeter are considered.

Request Cost

Request cost are defined as the mean number of packets sent per second over all requests of the simulation, including both successful and unsuccessful requests.

In addition to the request cost, we also state the cost incurred by server management (data migration, perimeter refresh protocol, topology exploration) and server advertisement. These cost will be further elaborated in the evaluation of data migration in Section 4.6.

3.5.4 Request Accuracy

Figure 3.25 through 3.30 show the simulation results for the request accuracy as a function of different system parameters. Let us start by considering Figure 3.25, which shows the request accuracy as a function of the cell width. The default cell width is 200 m. Depicted are two bars for LCS/BPR (left bar) and LCS/geocast (right bar), overlayed with a graph for DCS/GPSR. For the LCS variants, two colors indicate whether a request is received while a data server is in *active* or *receiving* state (cf. Section 4.5 for a discussion of these states).

We firstly observe that the request accuracy of all approaches drops gently and with a similar slope with increasing cell size. The cause for this behavior is that requests are only performed

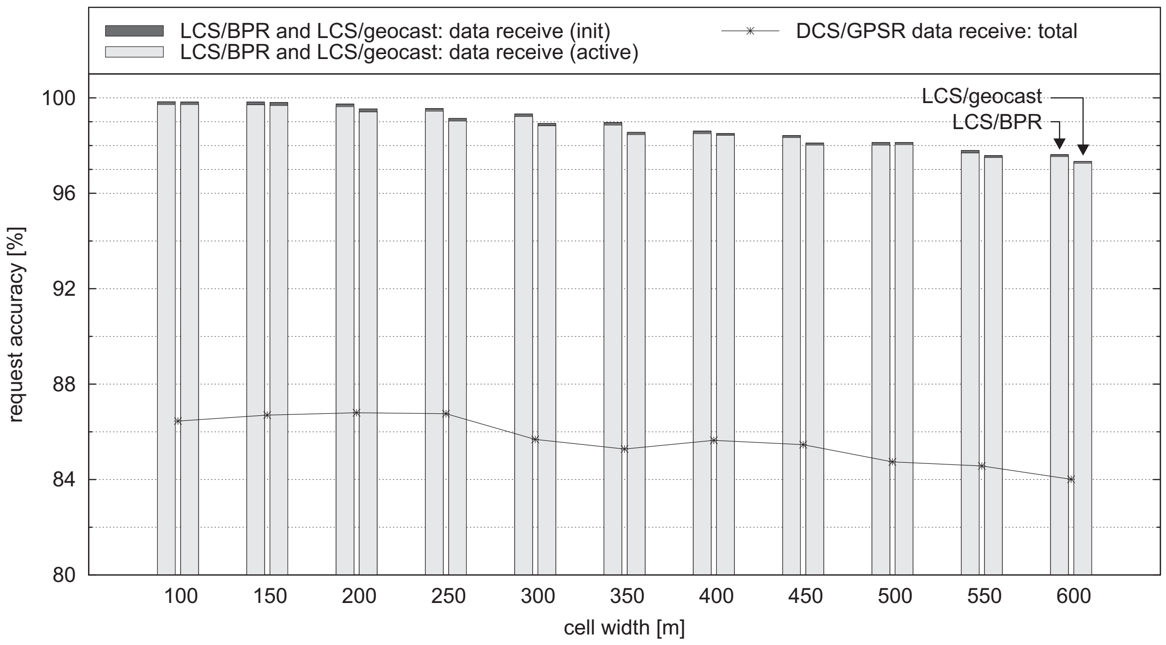


Figure 3.25: Request accuracy as a function of the cell size.

by nodes from inside of the single cell used in the experiment. With increasing cell size, the routing distance becomes larger, leading to an increase in the number of routing failures.

The second observation is that the request accuracy of both LCS variants clearly outperforms that of DCS/GPSR. This can be attributed to DCS/GPSR's difficulty to trace a full perimeter in a significant number of cases, in which a perimeter entry node cannot be identified.

Third, we observe that the request accuracy of LCS/BPR is slightly larger than that of LCS/geocast, which becomes apparent especially with increasing cell size. This is in particular interesting because intuition suggests that a larger distribution of server advertisements should make the localization of data servers during requests more successful. The reason for the performance differences can be understood from the two phases of request routing.

During the first phase, a request is routed towards the reference coordinate. Any node that holds advertisement information is a qualifying target node. Thus, this phase is very likely to succeed. In the second phase, unicast routing is used to finally reach a data server. This phase fails more often due to the addressing of a specific target node, which may not be reached if its position information is too outdated for the underlying geometric routing to locate it. This argumentation yields the fact that the first routing phase is generally more successful than the second routing phase if the same geometric distance has to be traversed.

Looking at the two LCS variants, we observe that in LCS/BPR, server advertisements are distributed always in the same way from the data server. In LCS/geocast, however, advertisements are distributed more and more distant from the server with increasing cell size. Whereas in LCS/BPR the ratio between the routing length in the first and second phase is constant in the average, in LCS/geocast the routing length in the second phase becomes larger in comparison to the first phase. Based on our argumentation in the previous paragraph, the overall routing failures thus increase more quickly for LCS/geocast.

Figure 3.26 shows request accuracy as a function of the number of nodes. Note that 150 nodes is the default value. In this figure, we have decreased the number of nodes down to 50. This is very low concerning the transmission range of only 100 m, where partitions begin to form (Section 2.2). We observe that even in the lowest density range, both LCS variants achieve a greater than 94% request accuracy. We also observe a small advantage of LCS/BPR over LCS/geocast which is due to the same reasons discussed in the previous figure.

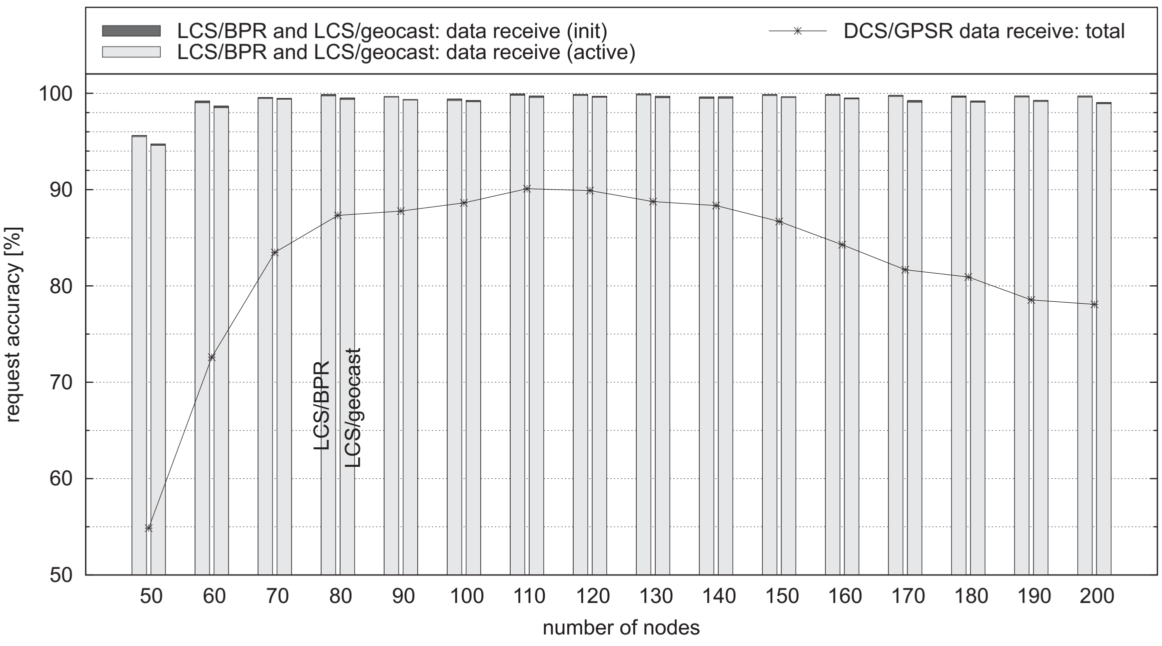


Figure 3.26: Request accuracy as a function of the number of nodes.

In contrast, DCS/GPSR's performance drops sharply starting at around 70 nodes and below. This is because the length of perimeters increases significantly with lower node density, and it becomes more difficult for DCS/GPSR to trace a closed perimeter. Mobility complicates this situation, where a small change of a node's position will lead to many inconsistencies between the planarized graph used for perimeter routing and the actual physical network topology.

With increasing node density, once the request accuracy of the LCS variants has reached its maximum at around 80 nodes already, it remains virtually constant. An additional observation is that starting at around 120 nodes, the performance of DCS/GPSR begins to drop again. This behavior is due to the graph planarization required by GPSR that is the basis for perimeter routing ([KK00]). With increasing node density, the number of nodes eliminated from a node's neighbor list to build its view of a planarized graph also increases. At the same time, mobility leads to the situation where each node's neighbor set fluctuates more strongly. Thus, with increasing node density, the inconsistencies increase between a node's planarized view on the graph and the physical graph, as well as between the planarized views on the graph of different nodes. In turn, perimeter routing is operating on a more inconsistent overall planarized graph with increasing node density, making it more difficult to terminate on a closed perimeter.

Figure 3.27 depicts request accuracy as a function of node speed. The first observation is the strong performance of both LCS variants up to the maximum tested speed of 15 m/s. At this speed, LCS/BPR is roughly 1% better than LCS/geocast due to the reasons discussed

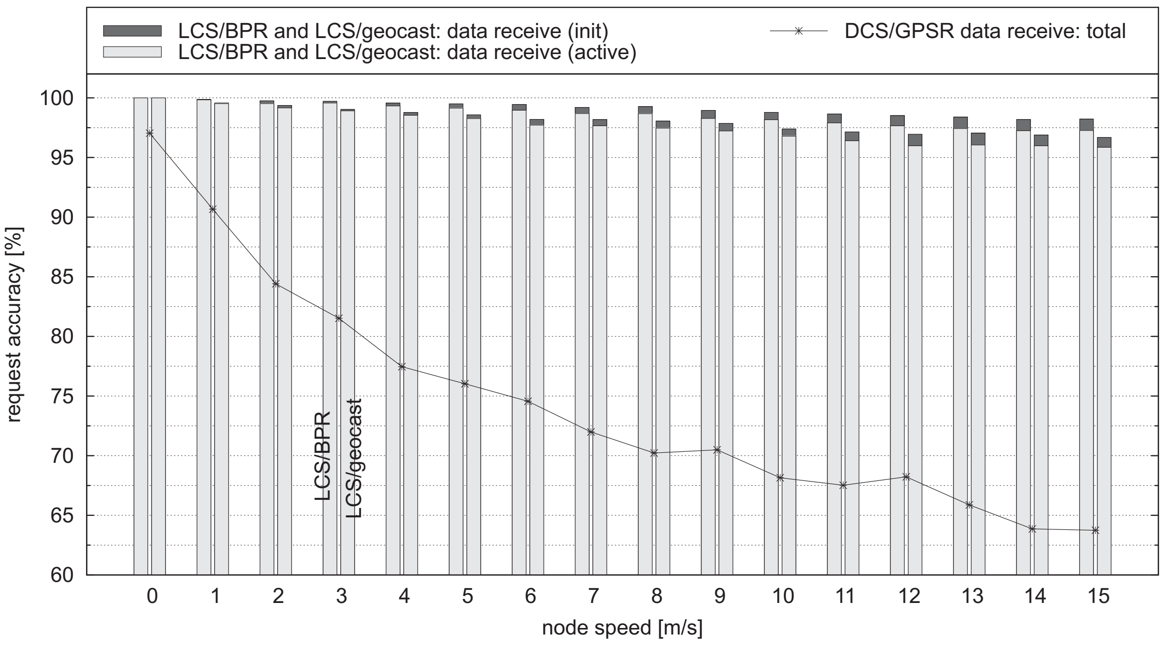


Figure 3.27: Request accuracy as a function of the node speed.

previously. This extremely good performance can be attributed to the fact that the rendezvous approach implemented in LCS is a resilient mechanism for request routing.

The performance gain of both LCS variants over DCS/GPSR is also significant, starting at small speeds already and increasing rapidly for larger node speeds. The reason is that the perimeter strategy is not able to cope well with node mobility. We further observe that with increasing mobility, requests are more frequently received at a data server while in *receiving* state. This is because migrations occur more frequently when a server is moving at a higher speed. This behavior is more pronounced in Figure 3.29, where we discuss it in more detail.

Figure 3.28 shows request accuracy as a function of node speed for LCS/BPR in direct comparison to DCS/GPSR. Larger cell sizes clearly have a negative impact on request accuracy, which increases significantly for larger node speeds for both LCS/BPR and DCS/GPSR. This is due to the limitations of geographic routing in general where neighbor information cannot be kept sufficiently accurate. The figure indicates clearly that smaller cells should be preferred over larger cells, especially when nodes moving at higher speeds can be expected.[4] However, we will show in Section 5.5 that smaller cell sizes adversely impact query performance. This implies a tradeoff that we will take up again in Section 5.5.4.

Figure 3.29 shows request accuracy as a function of the number of data items that are stored at a node. For convenience, also the data size in bytes is annotated below the abscissa. The figure reveals the impact of data migration on LCS.

Up to a relatively small number of about 2000 data items, corresponding to 64 kB of data, migration is able to complete quickly. Thus, most of the requests are received while the migrating

[4]Observe that for extremely small cells, request accuracy will eventually reach 100%, since every node will be a server and deliver requests to itself. However, this case corresponds to cell-based partitioning data storage (C-PART), treated analytically in Section 3.4, where we showed that it is not appropriate in our scope.

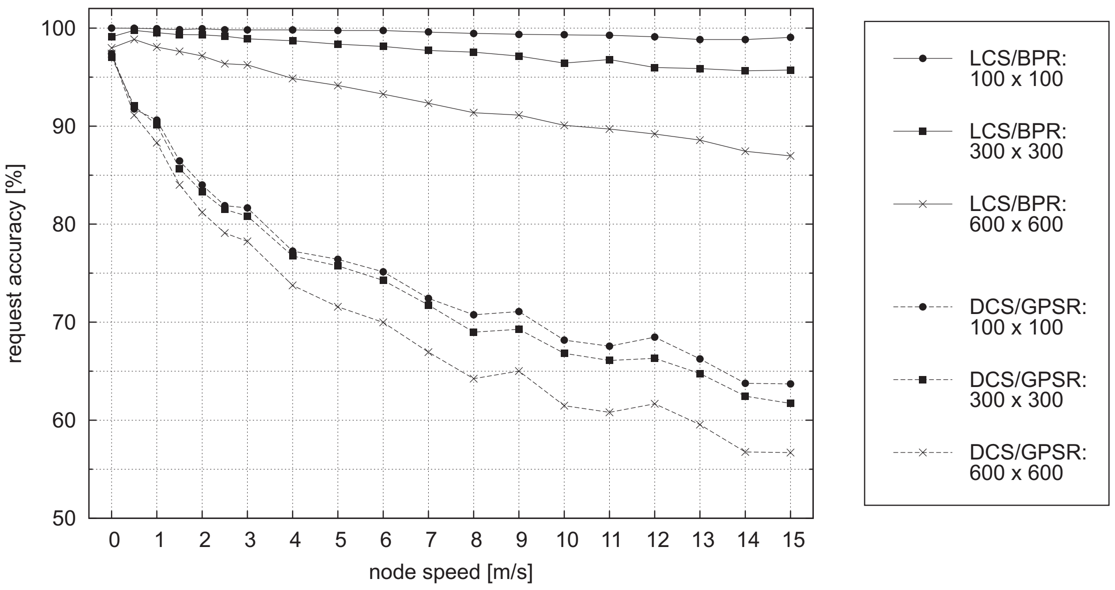


Figure 3.28: Request accuracy as a function of the node speed for different cell sizes.

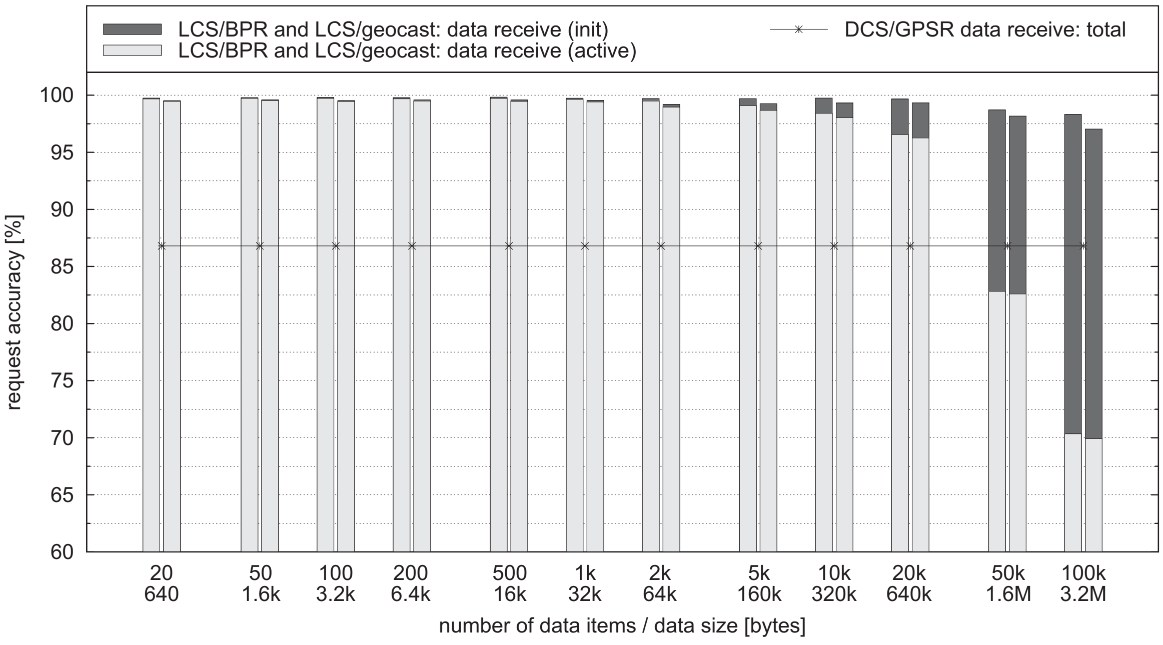


Figure 3.29: Request accuracy as a function of the number of data items.

data server is in *active* state. For about 5000 data items and above, a significant number of requests is received at the target server, which is in *receiving* state, during an active migration. Considering the additional indirection involved during a migration, it is remarkable that up to the maximum considered 100000 data items, corresponding to 3.2 MB of data, the request accuracy of LCS/BPR is about 98%. About 25% (the dark colored portion of the right-most column of LCS/BPR) of the successful requests can be attributed to the forwarding of these requests by our data migration approach during a migration in progress.

For DCS/GPSR, data migration is implicitly performed by the perimeter refresh protocol.

However, this mechanism operates differently and does not have a direct impact on the request accuracy, which is determined by the reaching of the correct perimeter entry node at a given point in time. Therefore, the accuracy performance of DCS/GPSR can be depicted as a straight line. Nevertheless, the performance of both LCS variants is well beyond that of DCS/GPSR for all considered settings of the data size.

Figure 3.30 shows the request accuracy as a function of the number of requests. This result is of significance because with an increasing number of requests, a data server will eventually become a bottleneck due to the many requests routed to the same spot. The immediate observation is that while both LCS variants perform without any decrease in the request accuracy across the whole considered spectrum, the performance of DCS/GPSR starts to drop significantly above 3000 requests, corresponding to a rate of 10 requests per second.

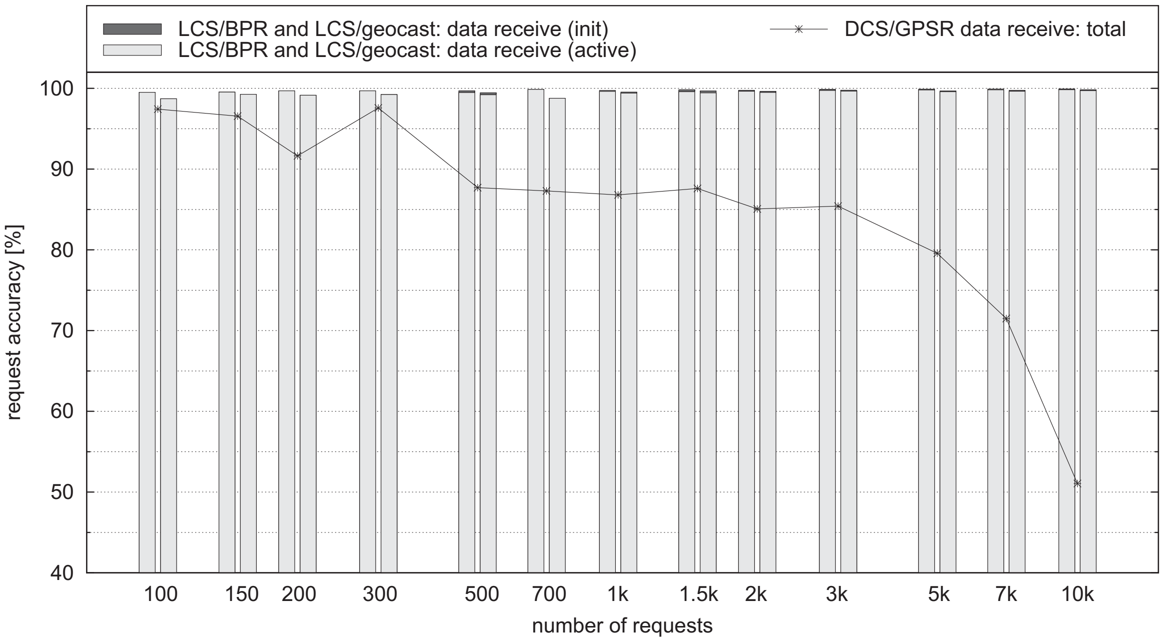


Figure 3.30: Request accuracy as a function of the number of requests.

This can be explained based on the characteristics of routing in the vicinity of the data server. In both LCS variants, a request can immediately be routed after any server advertisement has been found on some node other than the server. Thus, only the directly routed packets during the second phase of request forwarding (Section 3.3.2) aggregate in close proximity to the server. In DCS/GPSR, for every request it is necessary to perform a full perimeter traversal, which leads to the aggregation of a large number of packets along the perimeter.

3.5.5 Request Latency

The results for request latency are shown in Figure 3.31 through 3.35. Let us start with Figure 3.31, in which the request latency is shown a a function of the cell width. A key observation is that the latency of both LCS variants is much smaller than that of DCS/GPSR. Additionally, the difference between both location-centric storage variants and DCS/GPSR remains roughly constant across the whole considered spectrum. This is due to the fact that routing paths become proportionally longer in all three approaches.

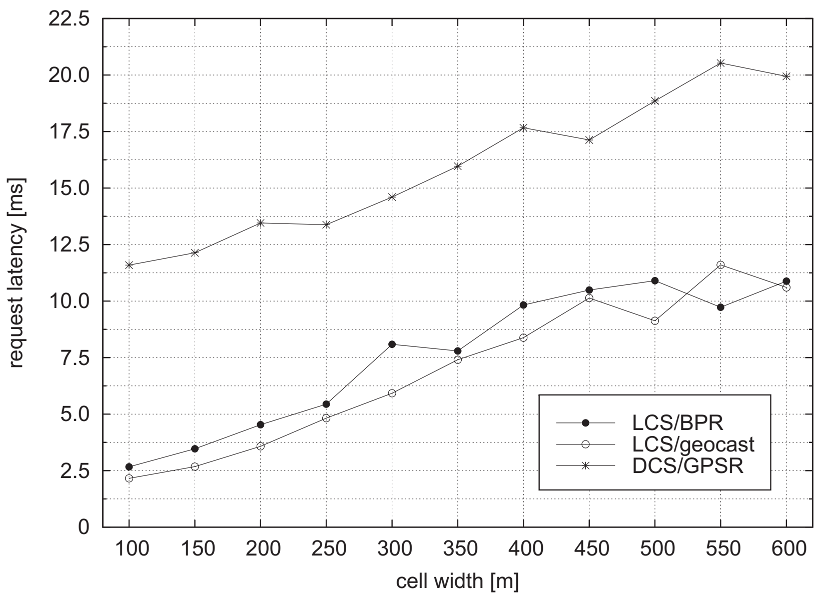


Figure 3.31: Request latency as a function of the cell size.

We further observe that the linear slope of all approaches leads to an increased performance gain for the LCS variants over DCS/GPSR. While at 600 m the latency of DCS/GPSR is about twice as large as that of LCS/BPR, at 100 m it is by a factor of approximately 5 larger. This is because DCS/GPSR incurs an initial overhead, which is due to the full perimeter traversals that are independent of the cell size and always required.

The generally very low latency incurred by the LCS variants leads to another important conclusion. When thinking of requests as data updates related to some particular context information, low latencies are essential to provide high quality of provisioned data at a low level of storage as a basis for the context services that build on top of it.

To see that LCS/BPR meets this requirement at a fundamental level, consider the following example. Assume that a request corresponds to a position update. Let the initial position information be acquired by the update initiator based on GPS with a typical inaccuracy of 5 m. Assume further a node speed of 15 m/s (which is the maximum speed we assume in our experiments), and an update latency of 5 ms (which corresponds to the cell width default value of 200 m), Based on these values, an update leads to an inaccuracy of $5 \text{ ms} \cdot 15 \text{ m/s} = 0.075 \text{ m}$ that can be attributed to update latency. Compared to GPS' inaccuracy of 5 m, this is only a small contribution that will have no significant impact on overall data accuracy.

Figure 3.32 and 3.33 show the request latency as a function of the number of data items. In Figure 3.32, the request latency of DCS/GPSR is depicted as a straight line, due to the same reasons related to the perimeter refresh protocol as in Figure 3.29. Note that the request latency of both LCS variants is well below that of DCS/GPSR as discussed in Figure 3.31.

We focus specifically on the impact of a large number of data items on the performance of the LCS variants. Up to 20000 data items, the request latency is roughly constant. Above this value, it increases significantly. This is due to the migration mechanism that incurs more time to complete for a very large number of data items. Note that it takes between 20000 and 50000 data items to match the request latency of DCS/GPSR. This is a remarkable number

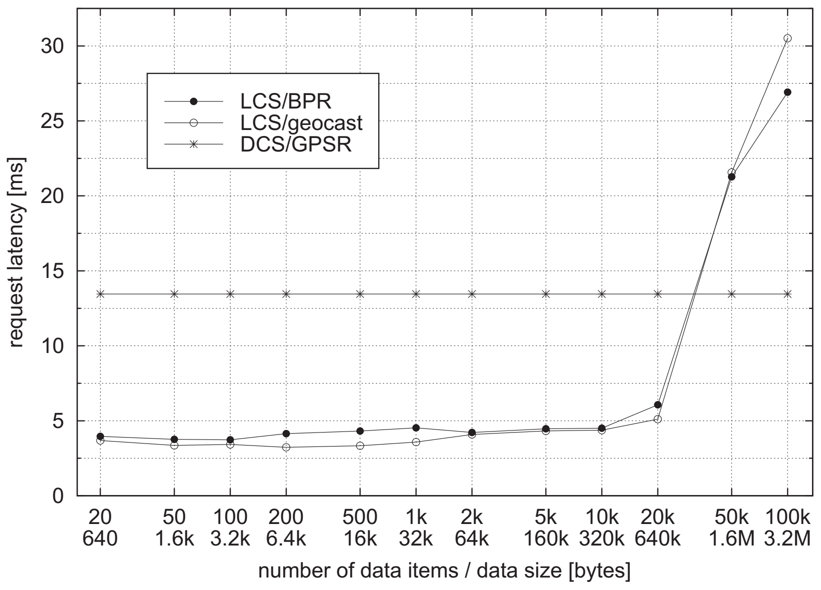


Figure 3.32: Request latency as a function of the number of data items.

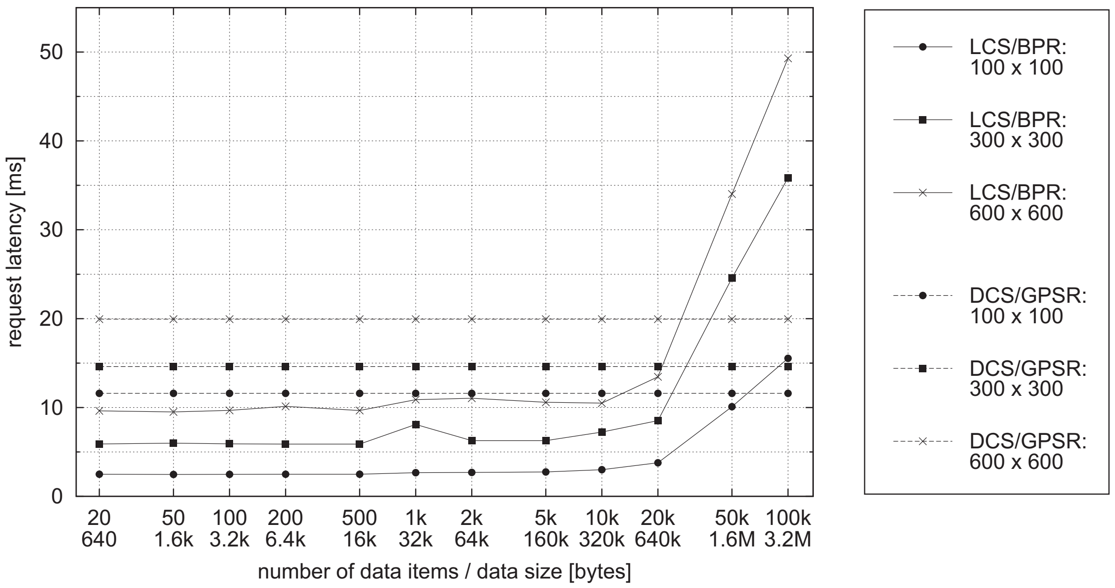


Figure 3.33: Request latency as a function of the number of data items for different cell sizes.

of data items. In terms of data size, this number corresponds to 1 MB of data, well within the envisioned data characteristics of the core data storage approach. In the case of an even larger data size, it is advisable to split a single cell into multiple smaller cells to account for a distribution of the load. Alternatively, in the case of static data, it is recommended to apply a different kind of storage approach, e.g., global replication as discussed in Section 3.4.4.

Figure 3.33 shows the request latency of just LCS/BPR and DCS/GPSR for different cell sizes. We observe that for smaller cell sizes there is a clear performance increase for both approaches in terms of request latency. Especially in the region where data migration incurs a larger time overhead, the increase in latency over a cell size of $100 \cdot 100$ m^2 is about 20 ms for $300 \cdot 300$ m^2

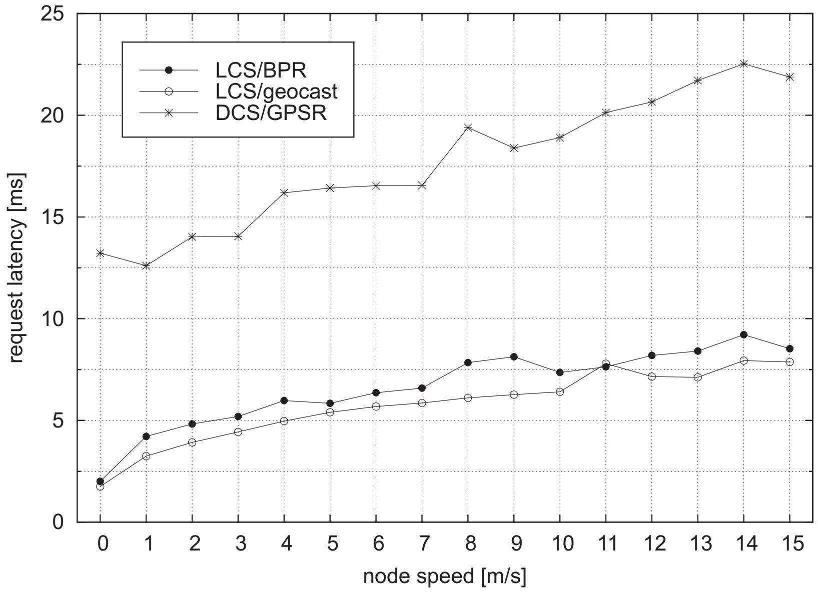


Figure 3.34: Request latency as a function of the node speed.

and 30 ms for $600 \cdot 600$ m^2 at 100000 data items.

An additional observation is that the intersection of curves between LCS/BPR and DCS/GPSR for the same cell size shifts to the right with decreasing cell size. In other words, with decreasing cell size, LCS/BPR is more efficient than DCS/GPSR for a larger number of data items. For example, at $300 \cdot 300$ m^2, the intersection of the LCS/BPR and DCS/GPSR graph occurs roughly at 30000 data items, whereas in the case of $100 \cdot 100$ m^2, this happens not until about 60000 data items. This shift is due to the constant perimeter overhead always present in DCS/GPSR as discussed in Figure 3.31, becoming more influential at smaller cell sizes.

This shift is helpful when considering the discussed option to split a single data server into several ones. Assume, for instance, a cell size of $100 \cdot 100$ m^2 and consider the case where a data server storing 100000 data items is split into two servers each holding 50000 data items. In this case will not only the request latency on both data servers decrease, but will also the performance of LCS/BPR surpass that of DCS/GPSR.

Figure 3.34 and 3.35 show request latency as a function of the node speed. In Figure 3.34, besides the clear performance advantage of both LCS variants over GPSR, we focus on the slight disadvantage of LCS/BPR over LCS/geocast. The difference can be explained based on the characteristics of the two-step routing applied in LCS/BPR and LCS/geocast.

Because in LCS/geocast an advertisement is likely to be found more quickly than in LCS/BPR, the second routing phase towards the data server can be initiated faster. This results in a slightly more optimal route of LCS/geocast compared to LCS/BPR in terms of total geometric distance. However, in the evaluation we assume that no additional advertisement distribution takes place. While we consider the performance in Figure 3.34 sufficient, additional advertisement based on beaconing can also be used to increase the availability of advertisement information within a scope similar to LCS/geocast, but at no additional cost.

In Figure 3.35 we have displayed the comparison of LCS/BPR with DCS/GPSR for different

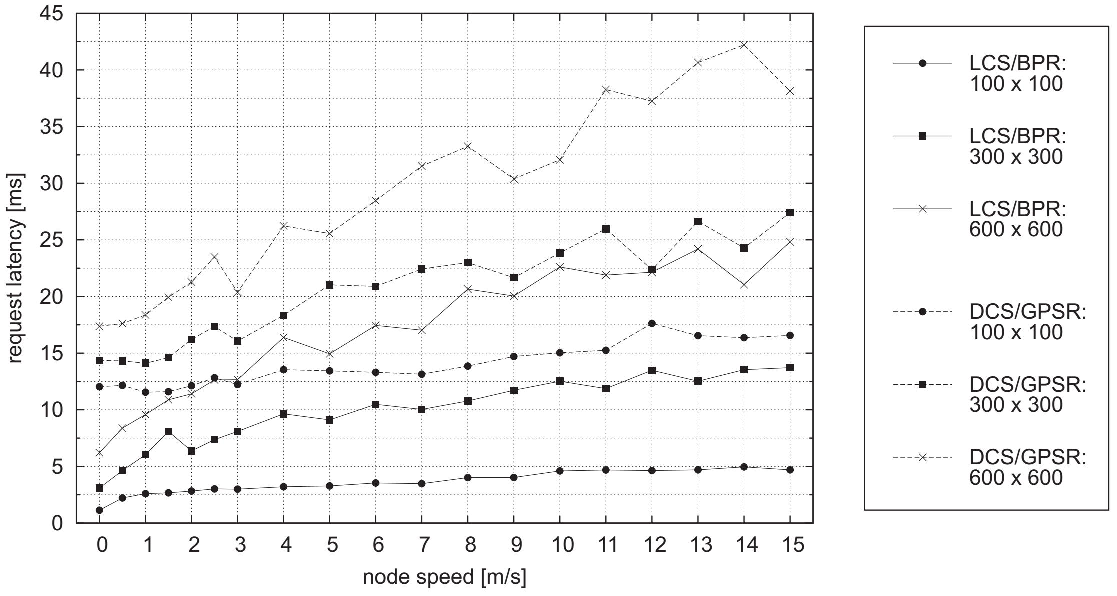


Figure 3.35: Request latency as a function of the node speed for different cell sizes.

cell sizes for a variation of the node speed. Small cells are clearly to be preferred over large cells, and for the same cell size, LCS/BPR outperforms DCS/GPSR in all cases.

3.5.6 Request Cost

Figure 3.36 through 3.41 show the request cost as a function of different parameters. Let us consider Figure 3.36 and 3.37 first, which show the request cost as a function of the cell size. Figure 3.36 provides a detailed view on the request cost together with all other maintenance and advertisement cost that occur in the different approaches.

The immediate observation is that the overall communication cost of LCS/BPR remains at a very low level throughout the considered spectrum of cell sizes. This is due to the fact that the particular scheme employed by LCS/BPR is independent of cell sizes, thus maintenance cost, including migration cost and the cost for topology exploration (Section 4.3), remain constant. Only the cost for requests increase due to the larger routing path length that itself increases with cell size, shown in more detail in Figure 3.37.

This fact is important for the following reason noted previously. We will show in Section 5.5 that in contrast to the core storage approach, larger cell sizes lead to a more efficient processing of spatial queries. Therefore, a tradeoff in the cell size needs to be achieved. The fact that the overall communication cost for request processing changes only insignificantly means higher flexibility in modifying the cell size for the benefit of more efficient query processing.

Figure 3.36 also reveals the major drawback of LCS/geocast, which was pointed out previously in the analytical study in Section 3.4. Because the advertisement mechanism of LCS/geocast is based on cell-wide flooding, advertisement cost increase significantly. Note that it is in principle possible that LCS/geocast applies a smaller cell area to disseminate server advertisements. This, however, is the very problem at the root of this approach due to the dependability on a specific node population within a cell, which is unknown beforehand. In contrast, this problem

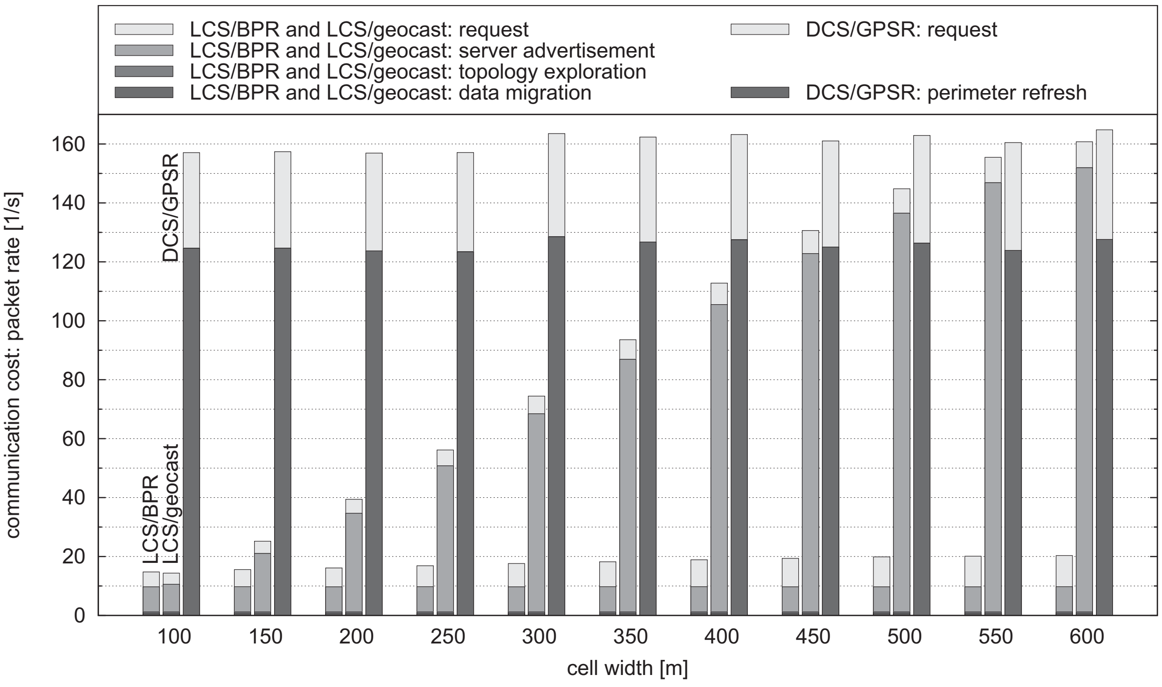


Figure 3.36: Communication cost as a function of the cell size.

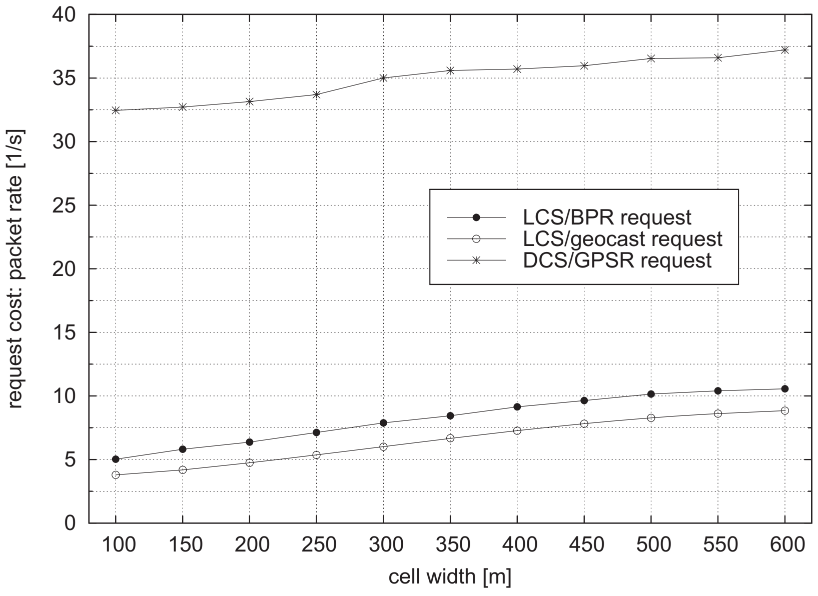


Figure 3.37: Request cost as a function of the cell size.

does not occur in LCS/BPR, because the applied scheme to distribute server advertisements is independent of the specific node population in a given region.

In Figure 3.37 we observe a strong correlation between communication cost and request latency from Figure 3.31. The significant larger overhead of DCS/GPSR compared to both LCS variants can be explained based on the previously stated fact that for each request, a full perimeter has to be traversed, independent of the cell size.

Figure 3.38 shows the communication cost of all three examined approaches in more detail. This figure and also Figure 3.39 through 3.41 are structured as follows. The graphs to the left show the total communication cost, which include cost of requests, maintenance and advertisement (solid lines), and the cost for requests only (dashed lines). The graphs to the right show maintenance and advertisement cost in separate curves and are included for reference to be able to understand the composition of the total cost. Observe the logarithmic scale on the ordinate in all of these figures. Further note that in the right part of the figures, the two curves for data migration + topology exploration for each of the LCS variants collapses into a single curve, because of their equivalent operation in both approaches.

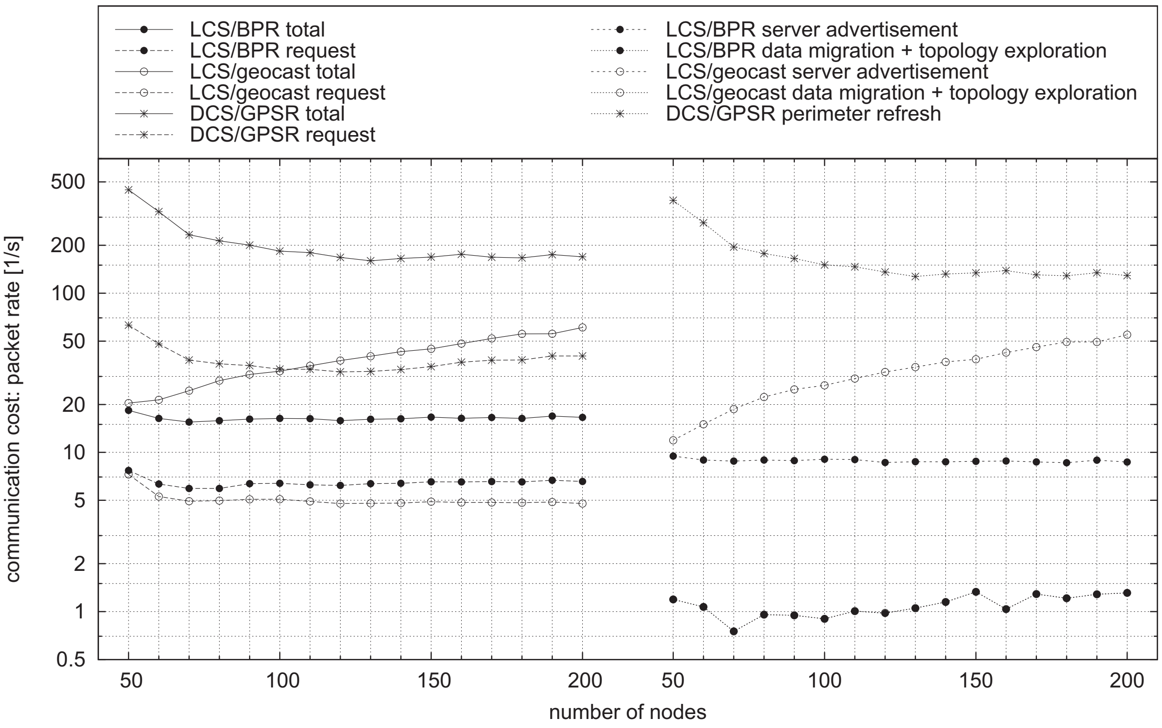


Figure 3.38: Communication cost as a function of the number of nodes.

We observe that LCS/BPR's total communication cost are smaller than that of LCS/geocast across the whole spectrum. Additionally, the performance gain becomes larger with increasing number of nodes. This is due to the increased flooding cost incurred by LCS/geocast for higher node densities, since every node has to send each advertisement once.

The total communication cost for DCS/GPSR are clearly above the cost for both LCS variants. This can be attributed to the routing overhead incurred by DCS/GPSR along perimeters. This overhead is required for both each request (left graph) and each refresh performed by the perimeter refresh protocol (right graph). This is also the reason why these two curves have similar characteristics across the spectrum of the number of nodes.

For small numbers of nodes, LCS/BPR shows one of its major strengths. Even down to 50 nodes, LCS/BPR's total communication cost only increase up to a very small amount. This increase is due to the fact that for small node densities, the chance to traverse a closed perimeter decreases, and more partial perimeters are fully traversed by bidirectional perimeter routing. However, the perimeter radius restricts the maximum number of hops traversed on a perimeter to a small value, keeping the total cost at a very low level.

This is different in the case of DCS/GPSR, where for low node densities, both longer and also more perimeters have to be traversed. Furthermore, the number of perimeters that cannot be completely traced also increases. The latter case leads to significant routing overhead until a packet finally experiences a timeout, which makes the packet cost increase significantly.

It is important to note that the small communication cost for LCS/BPR even at low node densities does not imply a severe degradation in the request accuracy, as was shown in Figure 3.26. Both results taken together substantiate the superior overall performance of LCS/BPR.

Figure 3.39 shows the communication cost as a function of the node speed. LCS/BPR incurs significantly lower overhead than the two other approaches across the whole considered spectrum. This is because LCS/geocast requires significant overhead in server advertisement compared to LCS/BPR, and DCS/GPSR has high request and perimeter refresh cost due to the traversal of full perimeters for each request and refresh, respectively.

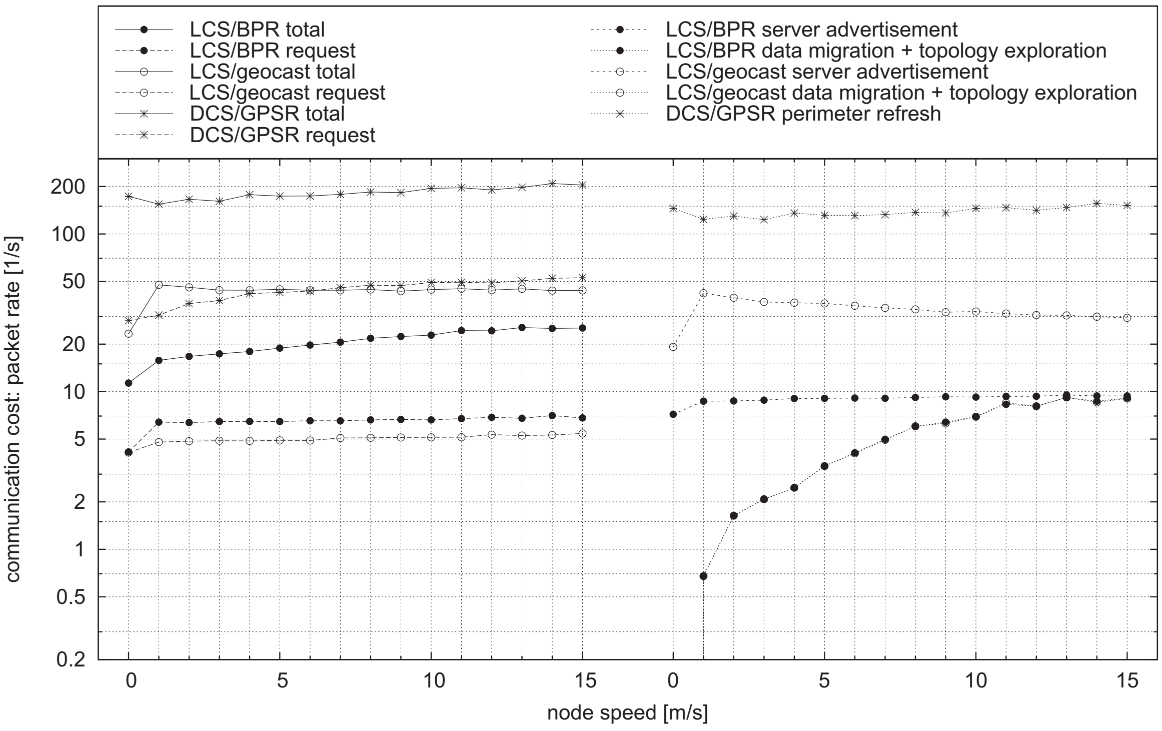


Figure 3.39: Communication cost as a function of the node speed.

However, with increasing node mobility, LCS/BPR's overhead also increases. This is due to the larger cost for data migration and topology exploration, shown in the right part of Figure 3.39. The observed increase is a natural occurrence because with larger node speed, more migrations are required to maintain spatial coherence. This also requires the execution of more topology explorations, which precede each migration attempt.

Figure 3.40 shows the communication cost as a function of the number of data items. LCS/BPR performs best compared to LCS/geocast and DCS/GPSR. This is because LCS/geocast, on one hand, needs a significant number of packets just for advertisement. On the other hand, the communication cost of DCS/GPSR increase rapidly in a linear way due to the proportionally increasing overhead in the PRP (note the logarithmic scale on both axes). This is because the PRP requires to send every data item during a single refresh cycle to all nodes on the perimeter. While this is a feasible approach in [RKY+02], where only a few data items (events) are considered, it clearly does not provide sufficient performance for larger amounts of data.

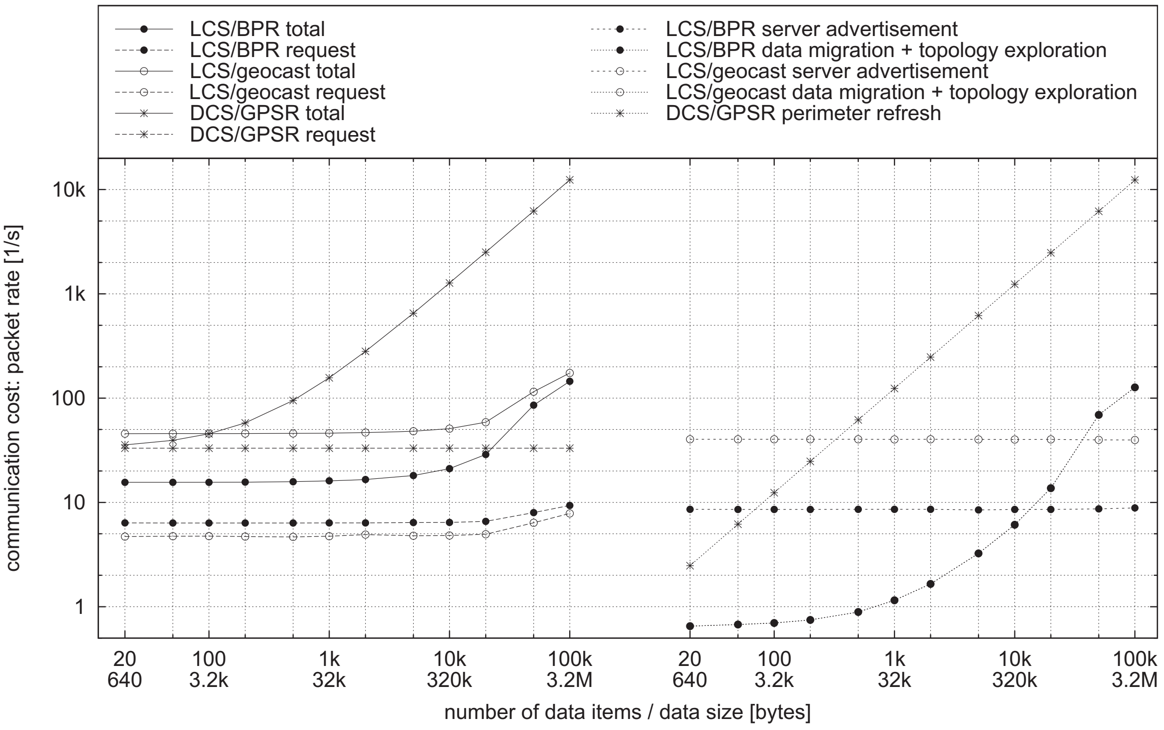


Figure 3.40: Communication cost as a function of the number of data items.

The second observation is that the communication cost of LCS/BPR increase noticeably in the upper range of the number of data items. This is due to migration, which incurs significant overhead, as can be seen in the right part of Figure 3.40, and which correlates immediately with the request latency discussed in Figure 3.32. However, since this also applies to LCS/geocast, the total communication cost of LCS/BPR remain below that of LCS/geocast.

An additional observation is that the request cost for both LCS variants increase at above 20000 data items. This is also due to the additional migration overhead, because packets are redirected by the migrating to the target server. The redirection effectively prolongs the distance a packet must travel until it reaches the target server, and adds to the overall request cost. Note the previous statement that this region of operation suggests to split the data subset across multiple servers, in order to reach a smaller number of data items on each resulting server.

Figure 3.41 shows the communication cost as a function of the number of requests. This figure emphasizes again the performance advantage of LCS/BPR over the other approaches, specifically, DCS/GPSR. The figure also reveals in detail that up to the maximum of 10000 requests, which corresponds to approximately 33 requests per second, LCS/BPR does not experience congestions. On the other hand, DCS/GPSR experiences heavy congestions starting at around 5000 data items, which occurs both for requests and refreshes. This behavior can be seen especially in the right graph of the perimeter refresh protocol overhead.

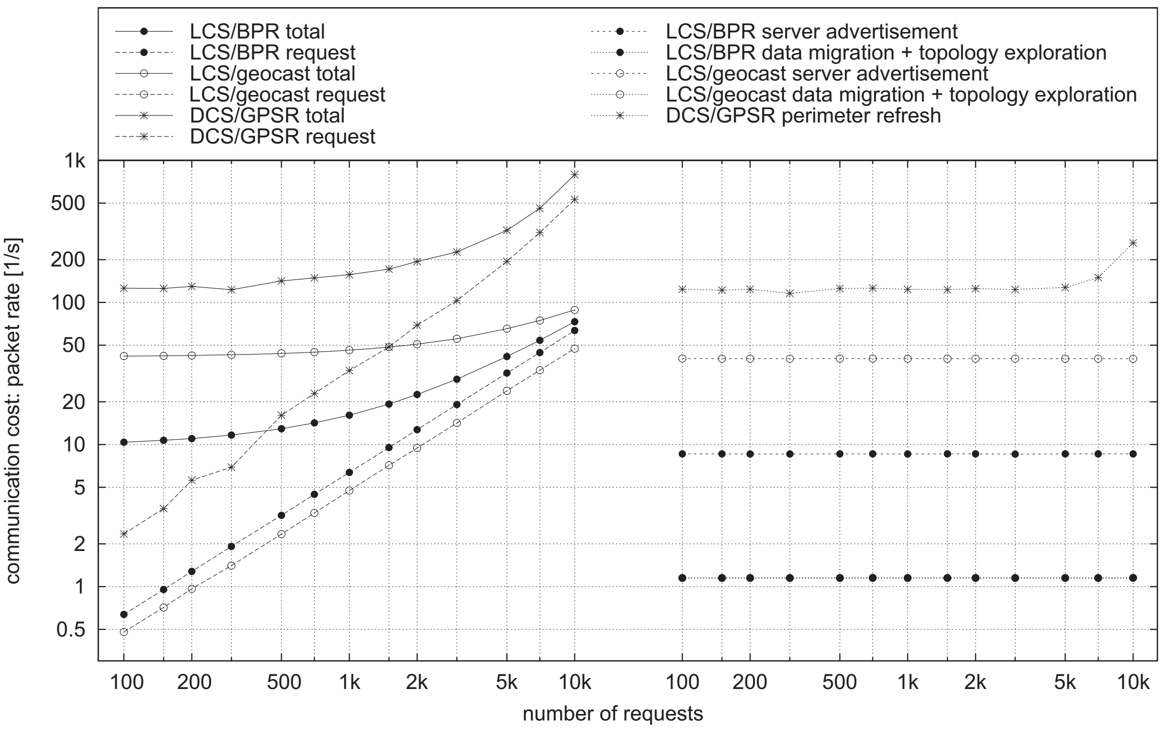


Figure 3.41: Communication cost as a function of the number of requests.

3.5.7 Evaluation Summary: Core Storage

The following key results can be summarized based on the evaluation of the previous sections. Firstly, the request accuracy of LCS/BPR is extremely high even under challenging conditions, such as low node densities and large numbers of data items. This fact shows well the general effectiveness of the core data storage approach, because the accuracy metric reflects well the overall conditions that algorithms must face in specific mobile ad-hoc network scenarios.

Second, LCS/BPR performs extremely well in the presence of low node densities in terms of request latency and request cost. At the same time, it is able to maintain high request accuracy. This makes it a strong candidate for scenarios where low node densities may occur quite frequently, and underpins its independency of any assumptions that are based on sufficient node population in any geographic regions.

Third, only for a very large number of data items, LCS/BPR's request latency and overall communication cost increase noticeably. This behavior suggests to split the data subset of a single server into multiple (e.g. two) subsets managed by individual servers in split cells, which would result in a significant improvement of both performance metrics. However, even in the case of large data subsets, migration does not severely impact accuracy.

Finally, the evaluation results show that small cells are to be preferred over large cells. However, across the spectrum of considered cell sizes, the performance of LCS/BPR can be maintained at a generally high level. This fact is particularly important for query processing, which shows the opposite performance characteristics and which we discuss in Section 5.5.

Chapter 4

Data Migration

In the previous chapter we have introduced core data storage, which provides the base mechanisms for data storage according to the location-centric storage paradigm (Definition 2.2). The main objective of core data storage was to address Requirement (1), (3), and (4) in Section 2.4, which capture storage efficiency and robustness. In the presence of node mobility, however, core data storage does not enforce spatial coherence according to Definition 2.2 between a data server's stored data and its associated reference coordinate.

To this end, we present the data migration component, which complements the core data storage component in the storage tier of the LCS framework of Figure 2.22. Data migration provides the strategies and mechanisms to maintain spatial coherence by migrating data subsets between data servers and thereby addresses specifically Requirement (2) in Section 2.4.

We begin with an overview of the migration framework in Section 4.1. In Section 4.2 through 4.4, we detail on migration policies, followed by a description of the migration mechanisms in Section 4.5. The evaluation of data migration will follow in Section 4.6.

4.1 Migration Framework Overview

Figure 4.1 presents the high-level structure of the framework for data migration. In the storage tier (Figure 2.22), the framework contains two components: migration policies and migration mechanisms. In the routing tier, these are supported by components that implement network topology exploration and resilient source routing.

The migration policies' task is to estimate the necessity, benefit, and success that a potential migration will bring in terms of achieving strong and continuous spatial coherence. Our approach works by splitting the migration policies in two parts: the migration recommendation and the migration decision. The *migration recommendation policy* (MRP, Section 4.2) uses the **local state** of the current data server to determine whether migration is needed and beneficial, that is, whether it leads to an increase in spatial coherence and thus storage efficiency. For that, it evaluates the *migration recommendation predicate* (P_{MRP}).

Only when P_{MRP} is true, that is, a migration is recommended, the *migration decision policy* (MDP, Section 4.4) is invoked to refine the recommendation in order to bring about a decision. The MDP examines the **remote state** of a set of possible candidate target nodes and attempts to find the most eligible one. In order to do so, the MDP is supported by the network topology exploration component (Section 4.3), which provides efficient distributed aggregation of the remote state. The MDP also considers the network stability of the current topology to determine whether a sufficiently stable path exists to perform a successful migration. A migration is only performed if the *migration decision predicate* (P_{MDP}) and the MRP agree to do so. In that case, a selected target server, an initial source route, and the priority value Π indicating the urgency of the migration, are handed to the data migration component (Figure 4.1).

The migration mechanisms are composed of two components: data migration and data consolidation. The *data migration* component is responsible for carrying out individual migrations based on the initial target server and source route provided by the migration policy. This component makes extensive use of resilient source routing (Section 4.5), which takes as input the provided source route and supports the transport of data between remote servers. The data migration component also relies on network topology exploration to update the source route during the migration of large quantities of data. Furthermore, the data migration provides as much transparency as possible for requests, both updates and queries.

Data consolidation is invoked asynchronously to the rest of the described processes in the event of migration failures. Such failures cannot be avoided in the system model we assume in Section 3.1 and require the consolidation of data stored at several servers. In close collaboration with core data storage, data consolidation is, however, able to detect server redundancies and initiate the merging of multiple servers, and thus, data subsets, into a single one.

4.2 Migration Recommendation Policy

The migration recommendation policy (MRP) is the initial step in the process of checking for whether or not a migration makes sense under the current network conditions. Each data server in the network monitors in regular time intervals Δt_{MRP} if the MRP ought to be invoked for a particular data subset. After each Δt_{MRP}, the MRP examines the server's local state and evaluates the *migration recommendation predicate*, denoted P_{MRP}.

If $P_{\mathrm{MRP}} = \text{true}$, local state indicates that migration is necessary from the point of view of the current data server, and that it will likely increase spatial coherence between the respective data subset and its associated reference coordinate (Section 3.1). If $P_{\mathrm{MRP}} = \text{false}$, the evaluation of the MRP is deferred to the next migration cycle after Δt_{MRP}. The length of the migration cycle is determined by the application and is a configurable parameter.

Besides this predicate, we introduce the notion of *migration priority* $\Pi \in [0, 1]$. This priority value dictates the strictness by which the subsequent MDP must determine a target server (Figure 4.1). If $\Pi = 0$ (minimum priority), the MDP has full freedom on choosing a candidate to receive the data to migrate. If $\Pi = 1$ (maximum priority), the MDP is forced to find a target node at any cost, even if the most eligible node may not be an optimal choice.

Let us first consider predicate P_{MRP}, which is composed of several partial predicates, each modelling a relevant aspect of a server's local state. The *spatial predicate*, denoted P_{spatial}, ensures that a data server will migrate upon moving too far away from its associated reference coordinate. Thus, it is the primary indicator for weak spatial coherence. Let t_0 denote the current time, $\mathbf{r}_0(t_0)$ the position of the server at t_0, and $d_0 = d_0(t_0) = |\mathbf{r}_0(t_0) - \mathbf{c}_r|$.

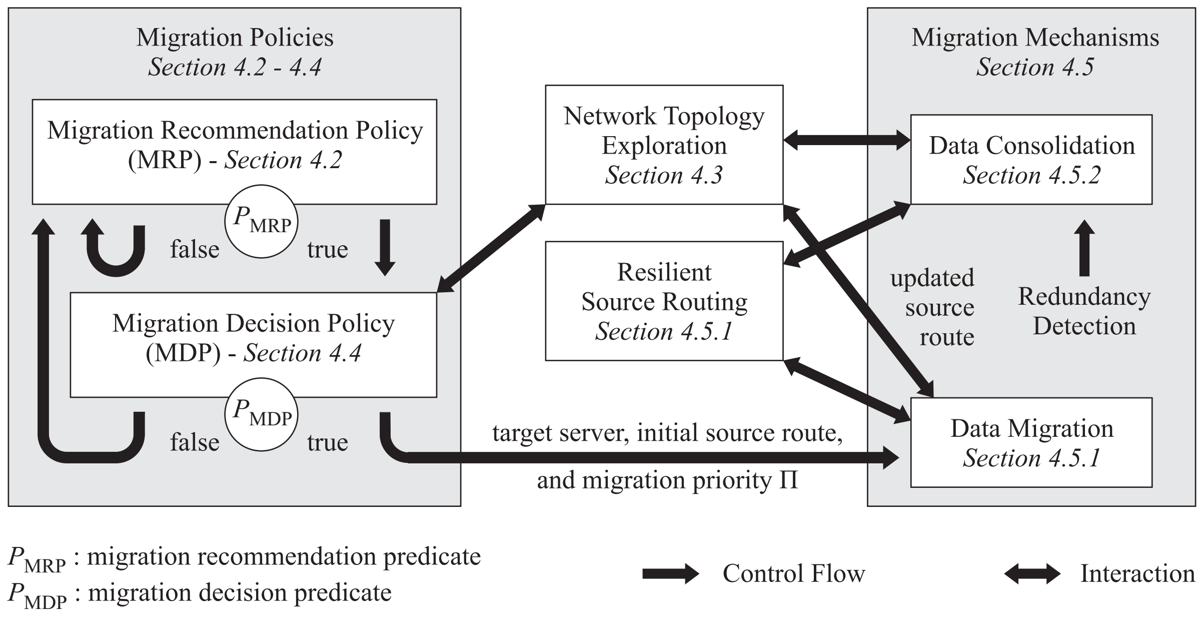


Figure 4.1: Overview of the data migration framework.

Figure 4.2 depicts several distances that are used to reason about how critical a migration is from the point of view of spatial considerations. By d_{opt}, we denote the optimal distance to the reference coordinate. Nodes located between d_{thresh} and d_{crit} are still suitable candidates. Beyond d_{crit}, nodes are considered no longer adequate to take the role of a data server. We normally choose $d_{\text{opt}}, d_{\text{thresh}}, d_{\text{crit}}$ to be multiples of r_{tx} but this is not strictly necessary.

The spatial predicate is defined based on the server's current position $d_0(t_0)$ and d_{thresh}:

$$P_{\text{spatial}} := \text{true} \Leftrightarrow d_0 > d_{\text{thresh}} \tag{4.1}$$

In other words, a data server first considers a migration if it is located more than d_{thresh} away from the reference coordinate $\mathbf{c}_r$. This parameter decides on the degree of spatial coherence that is desired in a particular scenario. Depending on request frequency and node density, d_{thresh} can be used to fine-tune overall network performance. We will show in Section 4.6 how different settings for d_{thresh} and node density influence spatial coherence.

When a data server moves beyond d_{crit}, it will indicate that a migration must occur from the spatial perspective, because sufficient spatial coherence is no longer given. In that case, the MDP may not have any freedom to decide whether or not to postpone migration. Since the loss of spatial coherence is a gradual process, it is possible to design the following linear function for the spatial priority Π_{spatial}:

$$\Pi_{\text{spatial}} := \begin{cases} 1 & \text{if} \quad d_0 > d_{\text{crit}} \\ \frac{d_0 - d_{\text{thresh}}}{d_{\text{crit}} - d_{\text{thresh}}} & \text{if} \quad d_{\text{crit}} \geq d_0 \geq d_{\text{thresh}} \\ 0 & \text{if} \quad d_0 < d_{\text{thresh}} \end{cases} \tag{4.2}$$

Intuitively, with increasing Π_{spatial}, a migration is recommended more urgently.

In the temporal domain, we define in analogy to the spatial domain the *temporal predicate*, denoted P_{temporal}, with associated *temporal priority* Π_{temporal}. The temporal priority realizes

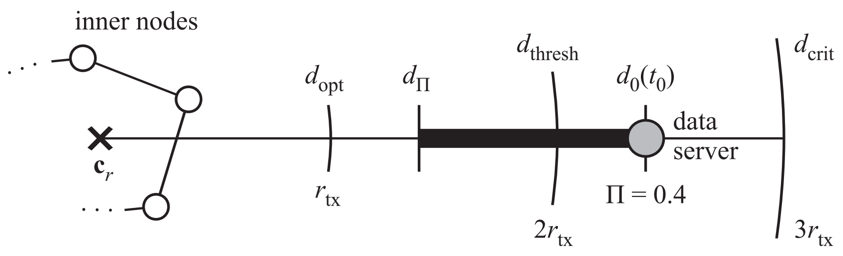


Figure 4.2: Migration recommendation predicate.

basic load balancing between nodes, even if nodes are low-mobile or stationary, in which case the spatial predicate might not trigger.

Let t_{act} denote the time a server became active for a particular data subset. Let Δt_{thresh} denote the temporal threshold, at which migration is to be first considered from the temporal viewpoint. Setting $t_{\text{thresh}} = t_{\text{act}} + \Delta t_{\text{thresh}}$, the temporal predicate is

$$P_{\text{temporal}} := \text{true} \Leftrightarrow t_0 > t_{\text{thresh}} \tag{4.3}$$

Let Δt_{crit} denote the critical time after which a migration must occur in any case. The general guideline is to choose a reasonable amount of time that allows nodes to take turns in the long term, typically several minutes. Setting $t_{\text{crit}} = t_{\text{act}} + \Delta t_{\text{crit}}$, the temporal priority is defined as

$$\Pi_{\text{temporal}} := \begin{cases} 1 & \text{if} \quad t_0 > t_{\text{crit}} \\ \frac{t_0 - t_{\text{thresh}}}{t_{\text{crit}} - t_{\text{thresh}}} & \text{if} \quad t_{\text{crit}} \geq t_0 \geq t_{\text{thresh}} \\ 0 & \text{if} \quad t_0 < t_{\text{thresh}} \end{cases} \tag{4.4}$$

It is possible to model additional predicates and priorities, for example, to take into account the computing capabilities, memory capacity, or energy resources of a device (Table 2.1). We omit this details but note that they may easily be accommodated based on the previously given notions of migration predicates and priorities.

Based on an assumption in the system model (Section 3.1), we further support the modelling of a node's explicit departure from the network. We define *exit predicate* P_{exit} to be true when a node explicitly indicates that it is about to leave the network. Because this is not a gradual process, the *exit priority* is defined as $\Pi_{\text{exit}} := (P_{\text{exit}} = \text{true}\ ?\ 1 : 0)$.

Considering the semantics of each of the stated partial predicates, a migration should be considered if any of these predicates is true. Therefore, we define the migration recommendation predicate P_{MRP} to be the disjunction of the partial predicates:

$$P_{\text{MRP}} := P_{\text{spatial}} \vee P_{\text{temporal}} \vee P_{\text{exit}} \tag{4.5}$$

In the case of the partial priorities, the largest priority dictates the overall importance of performing a migration. Thus, we define the migration priority Π as follows:

$$\Pi := \max\{\Pi_{\text{spatial}}, \Pi_{\text{temporal}}, \Pi_{\text{exit}}\} \tag{4.6}$$

If P_{MRP} is true, the migration decision policy will be triggered and delivered the value Π. Otherwise, the recommendation process repeats with the next migration cycle after Δt_{MRP}.

4.3 Network Topology Exploration

In order to obtain the necessary information about the remote state of the current data server prior to the evaluation of the MDP, we execute a two-phase distributed collection algorithm, which we call network topology exploration (NTE). In the first phase, we disseminate a topology request to a relevant subset of nodes and set up a reverse aggregation tree. In the second phase, each node reached sends a topology reply via the previously built aggregation tree, which contains information about its own local state relevant to the MDP.

Both dissemination and aggregation algorithms in this scope are well studied in the literature, see, for example, the survey on geocast routing protocols in [Mai04] and in-network aggregation in wireless sensor networks in [FRWZ07]. We therefore omit the details of these mechanism and focus on the shape of a suitable region $\mathbf{A}_{\mathrm{NTE}}$ that will likely contain highly eligible candidate nodes. Note that the information returned by NTE is not specified here, but described in detail in conjunction with the migration decision policy in Section 4.4.

Let $A_{\mathrm{NTE}} = |\mathbf{A}_{\mathrm{NTE}}|$ denote the area of the NTE region. Let $d_0 = |\mathbf{r}_0(t_0) - \mathbf{c}_r|$ denote the distance between data server u_0's current position and reference coordinate $\mathbf{c}_r$ at time t_0. $\mathbf{A}_{\mathrm{NTE}}$ must meet the following three requirements:

(1) To set up the aggregation tree in the vicinity of the server, it must be guaranteed that some initial nodes are reached, even when there are no nodes in the direction of $\mathbf{c}_r$. This is a known technical problem in the literature, and discussed e.g. in [Mai04]. We address this issue by defining a minimum distance R_0 around the current server u_0. We guarantee that requests will be flooded in the so-defined disk in any case. Therefore, u_0 must maintain at least the distance R_0 to any point on the boundary of $\mathbf{A}_{\mathrm{NTE}}$. An example where this restriction applies is given in Figure 4.3.a.

(2) Second, in order to limit the overall communication cost and total aggregation delay for the collection process, we limit A_{NTE} to a fixed maximum size always, independent of the distance d_0. We define $R = \min\{\max\{d_0, d_{\mathrm{thresh}}\}, d_{\mathrm{crit}}\}$. Then, $A_{\mathrm{NTE}} = \pi \cdot R_0^2$ if $R_0 \geq R$ (Figure 4.3.a), and $A_{\mathrm{NTE}} = \pi \cdot R^2$ otherwise (Figure 4.3.b-e). Note that while the area is fixed, the shape of region $\mathbf{A}_{\mathrm{NTE}}$ is not and depends largely on d_0.

(3) Third, in accordance with the migration recommendation predicate and priority in (4.1) and (4.6), respectively, eligible nodes are likely to be found near $\mathbf{c}_r$ and are to be preferred. Because a server may decide on a migration when t_{thresh} is reached or when it explicitly logs off from the network, such a server may be located near $\mathbf{c}_r$. In that case, nodes *further* from $\mathbf{c}_r$ than the current server (Figure 4.3.b) need also to be considered to provide a sufficiently large set of candidate nodes. On the other side, if the current server is located far from the reference coordinate, network stability may not allow to consider nodes all the way up to the reference coordinate (Figure 4.3.e). We therefore require that $\mathbf{A}_{\mathrm{NTE}}$ is always oriented towards $\mathbf{c}_r$, but limited by its own area A_{NTE}.

Depending on parameters d_0, R_0 and R, region $\mathbf{A}_{\mathrm{NTE}}$ takes various shapes, some of which are shown together with their formal specification in Figure 4.3. Observe that these parameters are sufficient to reconstruct $\mathbf{A}_{\mathrm{NTE}}$, so the more complex representations need not be transferred during the flooding phase of network topology exploration. The details on how to obtain the complete set of possible shapes is derived in Appendix D.

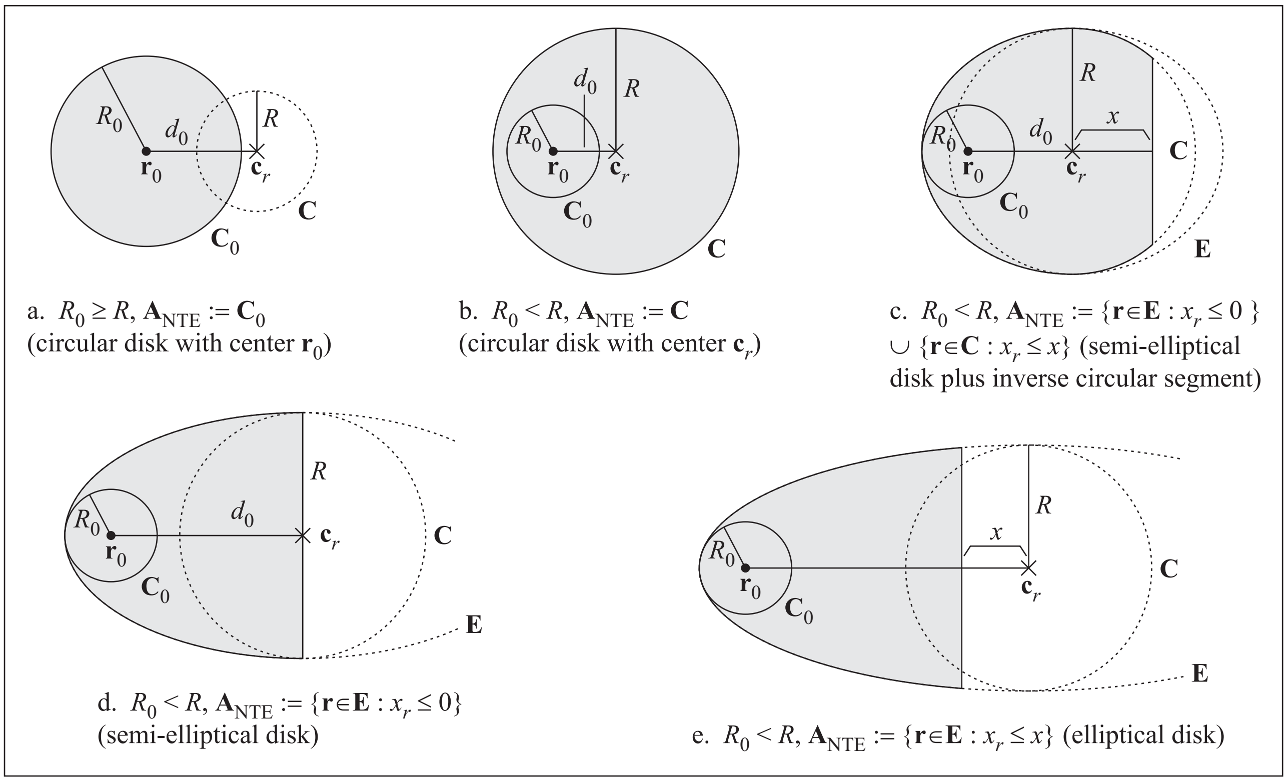

R_0
d_0
R
$\mathbf{r}_0$
$\mathbf{c}_r$
$\mathbf{C}$
$\mathbf{C}_0$
a. $R_0 \geq R$, $\mathbf{A}_{\text{NTE}} := \mathbf{C}_0$
(circular disk with center $\mathbf{r}_0$)
b. $R_0 < R$, $\mathbf{A}_{\text{NTE}} := \mathbf{C}$
(circular disk with center $\mathbf{c}_r$)
x
$\mathbf{E}$
c. $R_0 < R$, $\mathbf{A}_{\text{NTE}} := \{\mathbf{r} \in \mathbf{E} : x_r \leq 0\}$
$\cup \{\mathbf{r} \in \mathbf{C} : x_r \leq x\}$ (semi-elliptical
disk plus inverse circular segment)
d. $R_0 < R$, $\mathbf{A}_{\text{NTE}} := \{\mathbf{r} \in \mathbf{E} : x_r \leq 0\}$
(semi-elliptical disk)
e. $R_0 < R$, $\mathbf{A}_{\text{NTE}} := \{\mathbf{r} \in \mathbf{E} : x_r \leq x\}$ (elliptical disk)

Figure 4.3: Example shapes of network topology exploration region.

4.4 Migration Decision Policy

The task of the migration decision policy (MDP) is to issue the final decision on whether to perform a migration once the migration recommendation predicate (4.5) is true. Taking into consideration migration priority Π in (4.6), the MDP determines the values of three parameters (Figure 4.1): the migration decision predicate P_{MDP}, the selected target server u_{target}, and an initial source route U_{target} via which the migration will be carried out (Figure 4.1).

The MDP combines two aspects into a model for server selection. First, in order to rate the eligibility of a possible candidate node u_i, we introduce the notion of node eligibility $\epsilon_i \in [0, 1]$ in Section 4.4.1. This value describes how suitable a node is in terms of spatial and temporal characteristics to be used as migration target. At the same time, we will incorporate migration priority Π to relax node selection if a sufficiently eligible node cannot be determined.

Second, the topological properties of the network must be taken into account to avoid starting a migration in the case the topology is not sufficiently stable. For that purpose, we present a model for path stability in Section 4.4.2, which leads to the selection of only the most stable network paths via which migration is eventually performed.

In the following derivations of the MDP output values, we assume that an instance of network topology exploration was executed, and nodes inside of $\mathbf{A}_{\text{NTE}}$ have reported i) the geometric position $\mathbf{r}_i(t_i)$ of each node u_i at fixing time t_i, ii) the estimated speed $\bar{v}_i(t_i)$ of each reporting node, and iii) the fixing time t_i itself. By u_0, $\mathbf{r}_0(t_0)$, and $\bar{v}_0$ we denote the current server, its position at t_0, and its estimated speed, respectively. The final values for P_{MDP}, u_{target}, and U_{target} are defined in Section 4.4.3.

4.4.1 Node Eligibility

Based on the information reported by network topology exploration, the node eligibility $\epsilon_i \in [0, 1]$ combines three properties of a node u_i at fixing t_i into a single value.

Following from the definition of spatial coherence (Definition 2.3), the eligibility of a node is generally larger when it is located closer to the reference coordinate. This property was also modelled in the spatial domain of the MRP and is implemented by the distance-based node eligibility, denoted ϵ_i^d, in the MDP (Section 4.4.1.1).

A second magnitude not present in the MRP is the property that a node is more eligible when it is likely to move at low speed, therefore remaining in proximity to the reference coordinate for a longer period of time. We will characterize this magnitude by the notion of sojourn-based eligibility, denoted $\epsilon_i^{\delta t}$ and discussed in Section 4.4.1.2.

Third, the time-based eligibility, denoted $\epsilon_i^{\Delta t}$ and defined in Section 4.4.1.3, captures the duration for which a node is assigned the server role. Hence, this property corresponds to the temporal domain in the MRP. Finally, the node eligibility ϵ_i combines these three values into a single one, which will be accomplished in Section 4.4.1.4.

4.4.1.1 Distance-based Node Eligibility

Let us first characterize distance-based eligibility, and estimate the benefit that some node u_i relative to the current server u_0 will have in terms of its distance to reference coordinate $\mathbf{c}_r$.

We distinguish two cases, according to whether $\Pi < 1$ or $\Pi = 1$. The case where $\Pi < 1$ implies that $\mathbf{c}_r$ is contained inside of $\mathbf{A}_{\text{NTE}}$ (Figure 4.4.a). This follows from the definition of the migration recommendation predicate in (4.1) and the definition of $\mathbf{A}_{\text{NTE}}$.

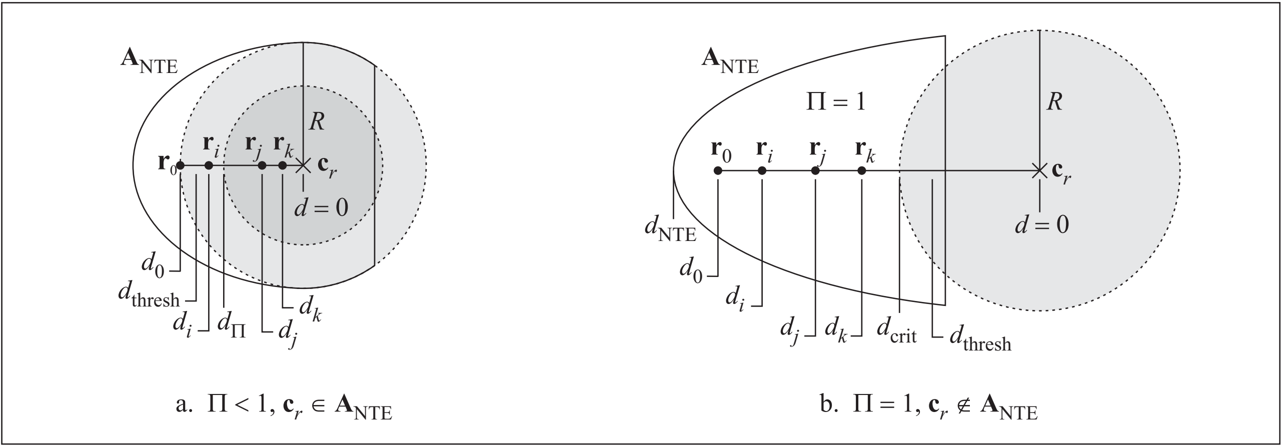


Figure 4.4: Migration decision policy: distance-based eligibility

Generally, nodes are more eligible when located closer to $\mathbf{c}_r$. However, when the network is only sparsely populated, there might not be any nodes located close to $\mathbf{c}_r$. In that case, the benefit of a migration will be small in comparison to what can be achieved for higher node densities. In such situations, it is more desirable to postpone migration to the next migration cycle until more suitable candidates become available. This is exactly the purpose of priority Π, which we apply to implement a deferral of a migration in such situations as follows.

Using the migration priority Π from (4.6), we are able to define a distance limit d_Π (Figure 4.4.a) that any candidate node must not exceed. Nodes located further from $\mathbf{c}_r$ than d_Π will not be considered as valid candidates. For $\Pi < 1$, we define:

$$d_\Pi = 0.5 \cdot (1 + \Pi) \cdot d_{\text{thresh}} \tag{4.7}$$

The interpretation of (4.7) is as follows: Consider $\Pi = 0$, which by definition leaves maximum freedom to the MDP to decide on a target server. In spatial terms, if $\mathbf{c}_r \in \mathbf{A}_{\text{NTE}}$, a node must gain at least 0.5 times the distance d_{thresh} in order to be considered eligible. Choosing d_{thresh} instead of d_0 in (4.7) makes sure that progress is always relative to the threshold. This is important if u_0 is almost co-located with the reference coordinate, in which case no candidate nodes may exist in a very small region. For larger values of Π, distance d_Π increases, which effectively increases the set of nodes that will be considered in the selection process.

The choice of the factor 0.5 can be understood from Figure 4.2. In the example, the distance of the current server to the reference coordinate is $d_0(t_0)$. According to (4.6), $\Pi = 0.4$. Then, (4.7) yields $d_\Pi = 0.7 \cdot d_{\text{thresh}}$, indicated in Figure 4.2. We can see that now not only nodes located up to d_{opt} are considered in the selection process, but also nodes located up to d_Π. That is, once the current server moves beyond d_{thresh}, priority Π together with the factor 0.5 effectively assures that migration will only occur over a distance of at least r_{tx}.

In the second case where $\Pi = 1$, migration must occur according to the MRP's recommendation, which effectively turns into a decision in this case. Therefore, we set $d_\Pi = d_{\text{NTE}}$, where d_{NTE} is the maximum distance between the NTE region, $\mathbf{A}_{\text{NTE}}$, and the reference coordinate $\mathbf{c}_r$ (Figure 4.4.b). That is, *any* node located inside of $\mathbf{A}_{\text{NTE}}$ is considered, even if it is located further away from the reference coordinate than u_0 itself.

We are ready now to define the distance-based eligibility ϵ_i^d of every node u_i. It is always defined relative to d_{thresh} to be compatible with the MRP. Let $d_i(t_0) = d_i(t_i) + \bar{v}_i \cdot (t_0 - t_i)$, then

$$\epsilon_i^d = \begin{cases} 1 - \frac{d_i(t_0)}{d_{\text{thresh}}} & \text{if} \quad \Pi < 1 \\ 1 - \frac{d_i(t_0)}{d_{\text{NTE}}} & \text{if} \quad \Pi = 1 \end{cases} \tag{4.8}$$

In the previous equation, we set $\epsilon_i^d = 0$ if $d_i(t_0) > d_{\text{thresh}}$.

We further require the notion of the minimum distance-based eligibility, denoted ϵ_d^Π, which denotes the eligibility that a node must satisfy in order to be a valid candidate:

$$\epsilon_d^\Pi = \begin{cases} 1 - \frac{d_\Pi}{d_{\text{thresh}}} & \text{if} \quad \Pi < 1 \\ 0 & \text{if} \quad \Pi = 1 \end{cases} \tag{4.9}$$

4.4.1.2 Sojourn-based Node Eligibility

The sojourn-based node eligibility, denoted $\epsilon_i^{\delta t}$, is related to the expected speed that a node u_i takes relative to the considered reference coordinate $\mathbf{c}_r$. We define the notion of estimated residual sojourn time, based on notions in [TDC01, MMM05], in such a way that it can be adopted for the definition of the sojourn-based eligibility.

Using the assumption that a data migration will occur after Δt_{thresh} if the spatial migration predicate in (4.1) does not trigger, we can define Δt_{thresh} as the maximum residual sojourn time. We first consider the case where $\Pi < 1 \Rightarrow \mathbf{c}_r \in \mathbf{A}_{\text{NTE}}$. We define the sojourn time δt_i of node u_i relative to the current time t_0 as follows:

$$\delta t_i = \min\left\{\frac{d_{\text{thresh}} - d_i(t_0)}{\bar{v}_i}, \Delta t_{\text{thresh}}\right\} \tag{4.10}$$

If $d_i(t_0) > d_{\text{thresh}}$, we set $\delta t_i = 0$. If $\bar{v}_i = 0$ for some i, we set $\delta t_i = \Delta t_{\text{thresh}}$. For $\Pi = 1$, we take the distance d_{NTE} as the reference:

$$\delta t_i = \min\left\{\frac{d_{\text{NTE}} - d_i(t_0)}{\bar{v}_i}, \Delta t_{\text{thresh}}\right\} \tag{4.11}$$

If $d_i(t_0) > d_{\text{NTE}}$, we set $\delta t_i = 0$.

In contrast to the distance-based node eligibility, an absolute reference for the sojourn-based eligibility does not make sense. This is because the sojourn time distribution of nodes can potentially be arbitrary, depending on the mobility of network nodes. We thus divide the set of nodes into two partitions, each having a size that is a function of migration priority Π.

Let us first consider $\Pi < 1$. Let $\delta t'_0, \ldots, \delta t'_n$ denote the estimated residual sojourn time values in increasing order, with equal values in arbitrary order. If $\delta t'_n > 0$, we define $k' \in \{0, \ldots, n/2\}$ to be $k' = \lfloor (1 - \Pi) \cdot (n/2) \rfloor$. Then we define $k = k'$ if $\delta t'_{k'} > 0$, and if $\delta t'_{k'} = 0$, we set $k = \min\{k'' : \Delta t'_{k''} > 0\}$. Then we set the minimum residual sojourn time to $\delta t_\Pi = \delta t_k$. If $\delta t'_n = 0$, we set $\delta t_\Pi = \delta t_{\text{thresh}}$. In the case where $\Pi = 1$ we immediately set $\delta t_\Pi = \delta t'_0$.

The set of nodes to be considered is defined as all nodes whose sojourn time is equal to or greater than the sojourn time $\delta t'_k$. In particular, value $\Pi = 0$ corresponds to the case where only nodes whose sojourn time is above or equal to $\delta t_{n/2}$ are considered (in the case where $\delta t_{n/2} > 0$). If $\Pi \lessapprox 1$, then $k = 0$ and all nodes (for which $\delta t_i > 0$) are considered as valid candidates. Thus, for smaller Π, the requirements regarding the residual sojourn time are relaxed in the sense that also nodes with larger sojourn time are considered in the selection process.

We can now define the sojourn-based eligibility of a node u_i as follows. For that, we define $\delta t_{\text{lim}} = \delta t'_n$ if $\delta t'_n > 0$ and $\delta t_{\text{lim}} = \Delta t_{\text{thresh}}$ otherwise. We obtain:

$$\epsilon_i^{\delta t} = \frac{\delta t_i}{\delta t_{\text{lim}}} \tag{4.12}$$

The minimum sojourn-based node eligibility is defined as

$$\epsilon_\Pi^{\delta t} = \frac{\delta t_\Pi}{\delta t_{\text{lim}}} \tag{4.13}$$

Observe that if $\delta t'_n = 0$, which implies that all $\delta t_i = 0$, then $\epsilon_i^{\delta t} = 0$ for all i, and $\epsilon_\Pi^{\delta t} = 1$. Note that this special case implies that there are no nodes within threshold distance d_{thresh} to $\mathbf{c}_r$ at t_0, and thus $\epsilon_i^d = 0$ for all i also holds.

4.4.1.3 Time-based Node Eligibility

While the MRP uses the temporal predicate to share load from the viewpoint of the current server, the MDP generalizes the notion to a set of nodes such that the management of data subsets is more evenly balanced and migration oscillations between a few nodes are avoided.

Let us first define the load factor $L_i \in [0, 1]$ of a node u_i. Let Δt_i^{busy} denote the total time node u_i was server for any number of data subsets. Let Δt_i^{up} denote the up time of a node since its joining the network (e.g. by being turned on). The load factor is

$$L_i = \frac{\Delta t_i^{\text{busy}}}{\Delta t_i^{\text{up}}} \tag{4.14}$$

L_i assumes its minimum if a node has never served, and the upper bound 1 if a node was always server for some data subset. Note that the load factor is already computed at remote nodes based on busy and up times, and the single value L_i is returned during topology exploration.

Relaxation by priority Π is implemented similar to the sojourn-based node eligibility. We use the reported subset of nodes from network topology exploration as the reference, because the goal is to distribute load evenly among these nodes. This time, we assume that $L'_0, \ldots, L'_n$ denote load factors in decreasing order. The load limit is $L_\Pi = L'_k$ with value k being defined based on the procedure in the sojourn-based node eligibility. The set of considered nodes is again split into two partitions according to Π. This time, only nodes with a sufficiently small load factor are considered. A node's time-based eligibility is

$$\epsilon_i^{\Delta t} = 1 - L_i \tag{4.15}$$

The maximum time-based node eligibility is defined as follows:

$$\epsilon_\Pi^{\Delta t} = 1 - L_\Pi \tag{4.16}$$

4.4.1.4 Combined Node Eligibility

We will now combine the individual notions of eligibility into a single eligibility value ϵ_i for each node u_i. First, a sensible combination requires that each of the individual eligibility values obey their own limit. Second, compensation for one dimension may be allowed if one or both of the other dimensions dominate. For example, this may occur when a fast-moving node is almost co-located with the reference coordinate, in which case the distance-based node eligibility dominates over the sojourn-based eligibility.

We therefore choose a multiplicative approach that we illustrate in Figure 4.5. We have put the distance-based and sojourn-based eligibility magnitudes on the vertical and horizontal axis, respectively. The time-based node eligibility extends into the third dimension. For ease of exposition, we have not detailed it in the figure.

The combined eligibility ϵ_i of a node u_i is defined as follows:

$$\epsilon_i := \epsilon_i^d \cdot \epsilon_i^{\delta t} \cdot \epsilon_i^{\Delta t} \tag{4.17}$$

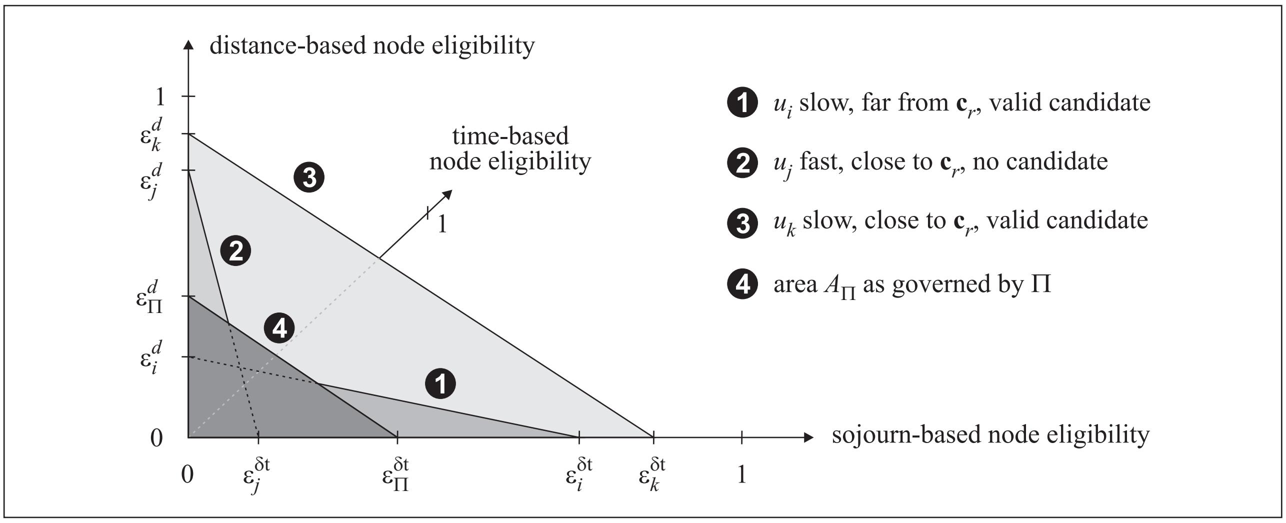

distance-based node eligibility
1
ε_k^d
ε_j^d
ε_Π^d
ε_i^d
0
time-based node eligibility
1
❶ u_i slow, far from $\mathbf{c}_r$, valid candidate
❷ u_j fast, close to $\mathbf{c}_r$, no candidate
❸ u_k slow, close to $\mathbf{c}_r$, valid candidate
❹ area A_Π as governed by Π
sojourn-based node eligibility
0
$\varepsilon_j^{\delta t}$
$\varepsilon_\Pi^{\delta t}$
$\varepsilon_i^{\delta t}$
$\varepsilon_k^{\delta t}$
1

Figure 4.5: Migration decision policy: combined node eligibility.

Likewise, the combined minimum eligibility ϵ_Π that a node must obey is

$$\epsilon_\Pi := \epsilon_\Pi^d \cdot \epsilon_\Pi^{\delta t} \cdot \epsilon_\Pi^{\Delta t} \tag{4.18}$$

Value ϵ_Π combines the requirements in each dimension into a one-dimensional value. In the case where a node u_i satisfies each individual eligibility, the overall limit is also satisfied, thus a node is considered eligible according to priority Π. For $\Pi = 1$, area A_Π, labelled ❹ in Figure 4.5, vanishes, thus any node is considered in the election process. In the figure, u_k clearly satisfies both the distance-based and sojourn-based eligibility requirements, thus it is eligible. Nodes u_i and u_j do not satisfy both individual eligibility values. However, u_i is able to compensate, since the value defined by $\epsilon_i^d \cdot \epsilon_i^{\delta t}$ is larger than $\epsilon_\Pi^d \cdot \epsilon_\Pi^{\delta t}$, while u_j is not. Note again that we have excluded the time-based node eligibility in this example, which works analogously.

4.4.2 Path Stability

In this section we introduce the concepts that incorporate the stability of network paths between the current server u_0 and target nodes u_i into the migration decision policy. Together with node eligibility ϵ_i in (4.17), the stability considerations will form a part of the migration decision predicate in Section 4.4.3 and influence the selection of the designated target server.

Several previous work considers models of link and path stability for different purposes, for example, [TDC01, GdWFM02, HSC03, YLG03, MMM05]. While we share some basic notions with this work, our proposed model and algorithm involve additional constraints that are not present in this form in the mentioned work.

We begin by defining our model for the estimated residual link lifetime in Section 4.4.2.1. In Section 4.4.2.2, we present the algorithm that determines the most suitable paths from the current server u_0 to potential candidate target servers u_i.

4.4.2.1 Estimated Residual Link Lifetime

Let us first consider the following model of link quality, which is based on geometric information that includes node positions $\mathbf{r}_i(t_i)$ and node speeds $\bar{v}_i(t_i)$, as returned from network topology exploration. We assume that the quality of a link connecting nodes u_i, u_j is modelled by a variable $q_{i,j}(t) \in [0,1]$ that describes the packet loss probability on that link. Let $d_{i,j}(t_0)$ denote the Euclidean distance between nodes u_i and u_j at time t_0. We define:

$$q_{i,j}(t) = \begin{cases} 1 & \text{if} \quad d_{i,j}(t_0) \leq r_{\text{tx}} \\ 0 & \text{otherwise} \end{cases} \tag{4.19}$$

The given model is an idealization in that it considers a link to be perfect up to the nominal transmission range r_{tx}. Instead of modelling the imperfection of a wireless link at all times, we address this issue implicitly by assuming a decreased available bandwidth for migration. We will come back to this aspect in Section 4.4.2.2.

Let t_0 denote the current time and observe that $\forall i : t_0 \geq t_i$. Assume further, w.l.g., that $t_i \geq t_j$. The distance $d_{i,j}(t_0)$ between any two nodes u_i, u_j at t_0 is defined as follows:

$$d_{i,j}(t_0) := |\mathbf{r}_i(t_i) - \mathbf{r}_j(t_j)| + \bar{v}_j \cdot (t_i - t_j) + (\bar{v}_i + \bar{v}_j) \cdot (t_0 - t_i) \tag{4.20}$$

The distance $d_{i,j}(t)$ between u_i, u_j at any future time $t \geq t_0$ is

$$d_{i,j}(t) = d_{i,j}(t_0) + (\bar{v}_i + \bar{v}_j)(t - t_0) \tag{4.21}$$

Excluding special cases, the residual link lifetime is described by the difference $t - t_0$ in (4.21). Reorganizing the equation yields

$$t - t_0 = \frac{d_{i,j}(t) - d_{i,j}(t_0)}{\bar{v}_i + \bar{v}_j} \tag{4.22}$$

Substituting r_{tx} for $d_{i,j}(t)$ in (4.22) yields the residual link lifetime $\tau_{i,j}$ between nodes u_i, u_j:

$$\tau_{i,j} = \begin{cases} \min\left\{\Delta\tau_{\max}, \dfrac{r_{\text{tx}} - d_{i,j}(t_0)}{\bar{v}_i + \bar{v}_j}\right\} & \text{if} \quad d_{i,j}(t_0) \leq r_{\text{tx}} \ \wedge \ \bar{v}_i + \bar{v}_j > 0 \\ \Delta\tau_{\max} & \text{if} \quad d_{i,j}(t_0) \leq r_{\text{tx}} \ \wedge \ \bar{v}_i + \bar{v}_j = 0 \\ 0 & \text{if} \quad d_{i,j}(t_0) > r_{\text{tx}} \end{cases} \tag{4.23}$$

In (4.23), we assume $\tau_{\max}$ to be a sensible upper bound, which is for numerical reasons only. If both $\bar{v}_i$ and $\bar{v}_j$ are zero, there is no relative movement between u_i and u_j and we also choose this value. If the distance between two nodes is larger than the nominal transmission range r_{tx} at the current time t_0, the residual link lifetime is, naturally, zero.

4.4.2.2 Algorithm for Path Computation

We now present the algorithm for determining appropriate paths from the current server u_0 to candidate nodes u_i via a sequence of intermediary nodes. The algorithm, shown in Listing 4.1, is executed by u_0 once the residual link lifetimes $\tau_{i,j}$ have been calculated according to Section 4.4.2.1. Procedure ComputePaths takes five arguments: $G = (U, E, \tau)$ denotes the graph that is constructed from the information returned by NTE (Section 4.3) and values $\tau_{i,j}$. U denotes the set of nodes u_i and E contains all edges $e_{i,j}$ for which $\tau_{i,j} > 0$ according to (4.23). Function $\tau : E \mapsto \mathbb{R}_0^+$ returns $\tau_{i,j}$ for each edge $e_{i,j}$, that is, $\tau(e_{i,j}) = \tau_{i,j}$.

Let D denote the total size in kB of the data subset to be migrated, and D_{res} the residual size of the data yet to be migrated. By B we denote the available bandwidth for migration in kB/s. This parameter models the fact that the full bandwidth of the communication channel may not be available to migration due to cross traffic or fluctuating link quality, as noted in Section 4.4.2.1. While a reasonable value can be statically assigned (e.g. 1 kB/s, as assumed in the evaluation of data migration in Section 4.6), it is also possible to estimate B on the fly by an additional algorithm. This detail, however, is out of the scope of this dissertation.

An important fact to consider in the path selection process is that it is unknown beforehand whether a sufficiently stable path exists to migrate the complete data subset from u_0 to some other node u_i. In this case, an initial path has to be determined, which may subsequently be updated while a migration is in progress. To distinguish between the two cases, procedure ComputePaths takes the parameter P_{target}, which is false if a path is determined for the first time, and true otherwise. The fifth argument, $\mathbf{U}_{\text{target}}$, which is passed by reference, contains a feasible suitable path to each node u_i once the procedure returns.

We further assume two global parameters. Firstly, Δt_{path} denotes the time required to redetermine a path while a migration is in progress. This value includes the time incurred by topology exploration, and the CPU time for path computations. Note that both values can be estimated by parameter settings of network topology exploration, such as aggregation delays, and, e.g. trial runs of path computations. Second, Δt_{MDP} denotes the blocking interval of the migration decision policy. Similar to the migration recommendation policy's monitoring interval Δt_{MRP} (Section 4.2), this parameter becomes effective when the migration decision policy is not able to determine an eligible node that can be reached via a sufficiently stable path. In that case, migration is deferred for the duration of Δt_{MDP}. While Δt_{MRP} can be set to a small value because the MRP involves only local computations, Δt_{MDP} is typically set to a larger value such that the network topology will likely have changed. Note that independent of the setting of Δt_{MDP}, migration will eventually occur when the migration priority Π reaches its maximum value of 1. This is guaranteed by the temporal priority according to (4.4).

Listing 4.1: Computation of network paths.

```
procedure ComputePaths(G: Graph; $D_{\text{res}}$: integer; B: real;
                       $P_{\text{target}}$: boolean; var $\mathbf{U}_{\text{target}}$: Path[n])
begin
  var $\tau_{\text{reqmax}}$, $\tau_{\text{reqmin}}$, $\tau_{\text{req}}$: real
  var $\text{pred}_{\text{BFS}}$: integer[|G.V|]
  var $l_{\min}$, $l_{\max}$, $l_{\text{stretch}}$: integer[|G.V|]
  var H: Graph
  var $l_H$: integer[|G.V|]
  var $\text{pred}_{H2}$: integer[|G.V|][|G.V|]
  var $d_{\text{final}}$: integer[|V|]  // final path distances
  var $\tau_{\text{final}}$: real[|V|]  // final residual path lifetimes
  var $E_G$: PriorityQueue
  var e: Edge
  var j: integer
```

```
15
16    $\tau_{\text{reqmax}}$ := $D_{\text{res}}/B$
17    if $P_{\text{target}}$ then
18       $\tau_{\text{reqmin}}$ := 0.001
19    else
20       $\tau_{\text{reqmin}}$ := $\max\{(1-\Pi)\cdot\min\{\tau_{\text{reqmax}}, \Delta t_{\text{path}}\}, 0.001\}$
21    end if
22
23    BreadthFirstSearch($G$, $u_0$, $\text{pred}_{\text{BFS}}$, $l_{\min}$)
24    for each $i \in \{1,\dots,|G.V|\}$ do $l_{\max}[i]$ := $\min\{\lceil l_{\min}[i]\cdot l_{\text{stretch}}\rceil, l_{\max}\}$
25
26    $\tau_{\text{req}}$ := $\tau_{\text{reqmax}}$
27    while $\tau_{\text{req}} \geq \tau_{\text{reqmin}}$ do
28       $E_G$.Init($G.E$, $G.\tau$) // largest residual link lifetime first
29       while $|E_G| > 0$ do
30             $e$ := GetAndRemoveNext($E_G$)
31             if $G.\tau(e) < \tau_{\text{req}}$ then break
32             $H.E$ := $H.E \cup e$
33             $H.V$ := $V \cup G$.GetAdjacentVertices($e$)
34             if $u_0 \notin H.V$ then continue
35             ClearArray($\text{pred}_{\text{BFS}}$, $-1$)
36             ClearArray($l_H$, 0)
37             BreadthFirstSearch($H$, $u_0$, $\text{pred}_{\text{BFS}}$, $l_H$)
38             for each $i \in \{1,\dots,|H.V|\}$ do
39                if $\text{pred}_{\text{BFS}}[i] = -1$ then continue // no predecessor
40                if $\text{pred}_{H2}[i][i] > -1$ then continue // previous predecessor
41                if $l_H[i] > l_{\max}[i]$ then continue // path length exceeded
42                j := i
43                while $\text{predBFS}[i] \neq -1$ do
44                   $\text{pred}_{H2}[i][j] := \text{pred}_{\text{BFS}}[j]$
45                   $j$ := $\text{pred}_{\text{BFS}}[j]$
46                end while
47                $d_{\text{final}}[i]$ := $l_H[i]$
48                $\tau_{\text{final}}[i]$ := $G.\tau(e)$
49             end for
50       end while
51
52       ConvertPaths($\text{pred}_{H2}$, $\mathbf{U}_{\text{target}}$)
53       if EvaluateMigrationDecisionPredicate($\mathbf{U}_{\text{target}}$) = true then return
54       $\tau_{\text{req}}$ := $\tau_{\text{req}}/2$
55    end while
56
57    if $P_{\text{target}}$
58       DeferMigration($\Delta t_{\text{MDP}}$)
59    else
60       GreedyMigration($\Delta t_{\text{greedy}}$)
61    end if
62 end
```

After the initialization of variables in lines 4-14 in Listing 4.1, the first objective is to set sensible upper and lower bounds for the required path lifetime. The upper bound, τ_{reqmax}, is the quotient of D_{res} and B (line 16) and denotes the estimated time required to migrate the complete data subset. This is the optimum value in the sense that if a path satisfies τ_{reqmax}, migration is likely to succeed without the necessity of subsequent path recomputations.

The lower bound, denoted τ_{reqmin}, depends on whether a migration is already in progress or the designated target server is yet to be determined. In the former case, no other option than finding the most stable alternative path to the already selected designated target server exists. In this case, we set the minimum required lifetime to a value that is close to but larger than zero in order for the following while loop to terminate eventually. In the procedure, we have arbitrarily set this value to 0.001, that is, one millisecond (line 18).

In the latter case, all options to determine a designated target server are still open and it is not mandatory to tolerate situations in which only unstable paths exist. In this case, τ_{reqmin} designates the minimum lifetime that a path must satisfy, otherwise, migration is deferred to the next blocking interval Δt_{MDP}. At this point, we are able to incorporate migration priority Π from (4.6) one more time, which quantifies the urgency of a migration and which we have used previously to describe sufficiently eligible nodes ((4.18)).

Let us first consider the expression $\min\{\tau_{\text{reqmax}}, \Delta t_{\text{path}}\}$ in the assignment of τ_{reqmin} as shown in line 20 of Listing 4.1. The rationale behind this term is that it is desirable to find a network path that lasts for at least the time it takes to redetermine a subsequent path. This means that subsequent path recomputations can be initiated sufficiently in advance so that a new path is available in time for a transparent switch of paths during migration (Section 4.5). When $\tau_{\text{reqmax}} < \Delta t_{\text{path}}$, however, only a small number of data items are to be transferred, and a path with smaller lifetime is already sufficient.

Let us now assume a small value for the migration priority Π, in which case τ_{reqmin} is set to a value close to the expression $\min\{\tau_{\text{reqmax}}, \Delta t_{\text{path}}\}$. This reflects the semantics of Π in that $\Pi \approx 0$ corresponds to the strongest restrictions on τ_{reqmin}. However, for larger Π, indicating a more urgent migration, the value of the expression is relaxed. Eventually, Δt_{reqmin} assumes its minimum value, which represents virtually no requirements on the path lifetime.

The purpose of lines 23-24 is to determine the maximum length that shall be allowed for a specific path. The motivation for this restriction is due to the following property of path lengths. Because the link lifetime model in Section 4.4.2.1 is based on the geometric distance between nodes, more stable paths imply a larger number of hops. However, with every additional hop communication cost increase.

For that reason, we restrict the maximum length of paths as follows. In line 23, the breadth first search (BFS) initially determines the shortest path from u_0 to all other target nodes u_i without considering the stability of links. Procedure BreadthFirstSearch returns, apart from the predecessor list that is not required here, the array $l_{\min}$, which contains the length of the shortest path to each node u_i. In line 24, the minimum path length between u_0 and each u_i is stretched by the factor l_{stretch}, which gives the maximum allowed path length $l_{\max}$ between u_0 and u_i. This allows to specify a sensible degree of flexibility in the subsequent path selection process. To avoid arbitrarily long paths, we limit the length of any path to the maximum value $l_{\max}$. Considering the dimensions of typical network topology exploration regions (Section 4.3), this value can typically be set in the range of a few hops.

Lines 26-50 perform the actual computation of the most stable paths from u_0 to each node u_i based on the restrictions on path lifetime and path length in the intervals $[\tau_{\text{reqmin}}, \tau_{\text{reqmax}}]$ and each $[l_{\min}[i], l_{\max}[i]]$, respectively. The outer while loop (lines 27-50) is initialized in line 26 with the required path lifetime τ_{req}. This value initially takes the most optimistic value τ_{reqmax}

to seek out highly stable paths first, before less stable paths are explored. Next, the priority queue E_G is initialized in line 28 with the set of edges E of graph G in descending order of the residual link lifetimes $\tau_{i,j}$. Right after entering the inner while loop (lines 29-49), edge e with the next largest residual link lifetime is retrieved from E_G (line 30). While the lifetimes of retrieved edges obey the required lifetime τ_{req} (line 31), edge e and its adjacent vertices are added to graph H (lines 32 and 33). If the start vertex u_0 has not yet been added to $H.V$ (line 34), no path exists that originates at u_0 and the while loop is continued. Otherwise, the predecessors and path lengths of the following BFS are reset (lines 35 and 36).

At this point, graph H contains only edges whose lifetime is at least τ_{req}, and a breadth first search is run on H in line 37. The task of lines 38-49 is to determine the paths between u_0 and each u_i such that each path's length falls within the valid interval $[l_{\min}[i], l_{\max}[i]]$. Lines 39-41 test various conditions under which a path cannot be updated. Firstly, in line 39, BFS could not determine a predecessor, in which case no path from u_0 to the considered node u_i exists. Second, in line 40, a previous predecessor was already determined, hence, a valid path with a larger residual path lifetime was already found in a previous iteration. Third, line 41 tests the length restriction, and no path larger than the restriction is allowed. If all restrictions are satisfied, the newly found path from u_0 to some u_i is updated in lines 42-46. Finally, the length and residual path lifetime of this path are set in lines 47 and 48, respectively.

The inner while loop eventually terminates due to the inability to find additional edges that satisfy the required lifetime (line 31). After leaving the while loop, the helper procedure in line 52 converts the predecessor list of each path from u_0 to u_i to the array of paths $\mathbf{U}_{\text{target}}$. Note that only paths that satisfy the stability requirements are present, hence, a path to every u_i does not necessarily have to exist. Observe also that $l_{\min}[i] \leq d_{\text{final}}[i] \leq l_{\max}[i]$. Next, the migration decision predicate is evaluated via a call to EvaluateMigrationDecisionPredicate in line 53, taking $\mathbf{U}_{\text{target}}$ as argument. Note that we have omitted node eligibility ϵ_i for ease of exposition (Section 4.4.3). If the MDP returns true, it was able to determine an eligible node that is also reachable via a sufficiently stable path. In this case, the procedure returns and the migration mechanism is subsequently fed with the designated target server and the path to it. If the migration decision is negative, then τ_{req} is relaxed according to an exponential decay function (line 54), and the process of finding sufficiently stable paths is repeated.

Eventually, τ_{req} drops below $\tau_{\min}$, in which case the outer while loop is exited. If the migration was just initiated and the target server was not yet selected, the migration is deferred to the next migration cycle, which occurs after Δt_{MDP} (line 58). In the case where a migration is already in progress and a source route is not available, geometric routing is used instead. This decision is motivated by the fact that migration should continue in the hope that it will succeed without specific robustness measures. We will investigate both source routing and geometric routing with respect to the impact on migration performance in Section 4.6.

4.4.3 Output of the Migration Decision Policy

We now define the migration decision predicate P_{MDP}, which decides on whether or not a migration is to be carried out. The input to the MDP comprises the node eligibility values ϵ_i and ϵ_Π from (4.17) and (4.18), respectively, and the set of paths $\mathbf{U}_{\text{target}}$ as returned from procedure EvaluateMigrationDecisionPredicate($\mathbf{U}_{\text{target}}$) in Listing 4.1.

Let us first construct the auxiliary set I containing only indexes for target nodes that are sufficiently eligible and to which a sufficiently stable path from u_0 exists:

$$I = \{i \mid \epsilon_i \geq \epsilon_\Pi \ \wedge \ \exists \text{ path } \mathbf{U}_{\text{target}}[i] \text{ to } u_i\} \tag{4.24}$$

The migration decision predicate can be directly stated as follows:

$$P_{\text{MDP}} := |I| > 0 \quad \text{or, alternatively,} \quad P_{\text{MPD}} := \exists i \in I \tag{4.25}$$

Hence, migration will occur if a node exists that has at least the eligibility that priority Π dictates, and a sufficiently stable target path exists to that node. Observe that an eligible node may not be reachable because no sufficiently stable path exists and vice versa.

The target node u_{target} and the target path U_{target} to that node are defined as follows:

$$(u_{\text{target}}, U_{\text{target}}) := (u_i, \mathbf{U}_{\text{target}}[i]) : i \in I \wedge \forall j \neq i : j \in I \Rightarrow (\epsilon_i, i) > (\epsilon_j, j) \tag{4.26}$$

In the previous equation, we have established a total order of tuples (a, b) with component-wise comparison, where the first element beats the second element in a comparison.

If the migration decision predicate in (4.25) is true, the most eligible target node reachable via a sufficiently stable path is input to the migration mechanism. Otherwise, the policy is deferred to the next migration cycle after the blocking interval Δt_{MDP} (Section 4.4.2.2).

4.5 Migration Mechanism

This section describes the mechanism of the data migration component, shown in Figure 4.1 and 2.22. The task of the mechanisms is to implement the robust and efficient transfer of a data subset from its current (source) server u_0 to the designated target server u_{target}.

The objective of the mechanisms is twofold. Firstly, data migration is carried out upon the triggering of the migration decision policy. We describe this process in Section 4.5.1. Second, data consolidation is required when two servers are handling the same data subset concurrently due to a previous migration failure. This process is described in Section 4.5.2.

4.5.1 Data Migration

The migration process is initiated by the source server u_0 after the positive evaluation of the migration decision predicate in (4.25). The algorithm shown in Listing 4.2 specifies the interactions between u_0 and u_{target} that take place during a migration. The interactions are governed by the following states. A data server operating in *active* state receives and processes updates and queries on the data subset, which it receives based on the core data storage mechanism in Section 3.3.2. The *migrating* and *receiving* states denote the condition where source and target server, respectively, are involved in a running migration process. During these states, a data subset is transferred from the source server u_0 to the target server u_{target}. By *retired* we denote the state where the source server has completed its migration and the target server has fully taken over, that is, assumed the *active* state.

The entry point of the data migration process is procedure InitMigration in Listing 4.2 (line 11), which takes target server u_{target} and path U_{target} from (4.26) as input. Every migration is set up by an explicit migration request that u_0 sends to u_{target} (line 15). The request contains U_{target} so that both nodes will communicate via that path. When u_{target} receives the migration request (line 18) and if it is able to accept a migration, it assumes the *receiving* state (line 22) and replies with a migration acknowledgement (line 23). However, if, for example, u_{target} is already involved in the processing of another migration, it may decline the request to avoid becoming a communication bottleneck (not shown in Listing 4.2).

When the migration acknowledgement is received from the target server (line 26), u_0 assumes the *migrating* state (line 28). The subsequent transfer of the serialized data subset from u_0 to u_{target} is initiated by an asynchronous call to InitSendDataSubset (line 29). During the complete transfer, a customized transport/routing layer protocol is used, optimized for the source route U_{target} that is used between both nodes. On the transport layer, any transport protocol suitable for mobile environments may be used. However, we optimize flow control by using a sliding window-based mechanism that is tuned to wireless ad-hoc networks. In this approach, window sizes are determined as a function of the number of hops between u_0 and u_{target}. In order to achieve maximum throughput, the window size is selected based on the results obtained in [CXSN04]. The rationale is to consider the special nature of the wireless medium. In contrast to a switched network, where the full bandwidth on both the inbound and outbound line of a switch/router is concurrently available, this does not hold in MANETs. Instead, a router, i.e. any network node, must share the medium for inbound and outbound traffic.

On the routing tier, resilient source routing is used, which takes the initially determined path U_{target} as input. The path's validity is determined by the residual path lifetime $\tau(U_{\text{target}})$. The timer in line 30 controls when path recomputation is due, such that a new path is available for a transparent switch before the previous path becomes stale. For that, the timer is initialized with the difference between $\tau(U_{\text{target}})$ and Δt_{path}, where the latter denotes the time to redetermine a path (Section 4.4.2.2). Observe that if the path's residual lifetime was already below Δt_{path}, the timer will immediately expire. Upon expiration, procedure TimerExpired (line 33) initiates the recomputation of a new path (line 35). While data migration continuous to use the current path U_{target}, network topology exploration according to Section 4.3 and procedure ComputePaths in Listing 4.1 are run in the background to redetermine a new path. Once determined, procedure PathChangeNotification is invoked (line 38) and the migration mechanism transparently switches from the current path to this new path (line 40). The timer is initiated again in line 41 for subsequent path computations.

Every data item received at the target server (line 44) is stored locally (line 46) and acknowledged (line 47). When the last data item is received by the target server (line 48), it immediately moves into *active* state (line 49). Then it begins sending server advertisements (line 50, Section 3.3.1), becoming known to other nodes in the vicinity of its associated reference coordinate and accepting incoming requests. Upon the reception of a data acknowledgement at the source server (line 54), u_0 checks whether all data items of the subset to be migrated were acknowledged, that is, whether the data transfer is complete (line 56). In that case, the source server becomes aware that u_{target} has taken over. It stops sending server advertisements (line 57), goes to *retired* state (line 58), and migration terminates. At this point, the target server has fully taken over the data subset and is the only one that is now in *active* state.

Listing 4.2: Data migration.

```
module DataMigration
begin
  var $u_{\text{target}}$: Node // used by $u_0$
  var $u_0$: Node // used by $u_{\text{target}}$
  var $U_{\text{target}}$: Path // used by $u_0$ and $u_{\text{target}}$
  var pathRecomputationTimer: Timer // used by $u_0$
  var state: MigrationState := active // used by $u_0$ and $u_{\text{target}}$
  var storage: Set of Object // used by $u_0$ and $u_{\text{target}}$
  var buffer: Set of Object // used by $u_0$

  procedure InitMigration($u_{\text{target}}$: Node; $U_{\text{target}}$: Path)
  begin // executed by $u_0$
    this.$u_{\text{target}}$ := $u_{\text{target}}$
    this.$U_{\text{target}}$ := $U_{\text{target}}$
    SendMigrationRequest($u_{\text{target}}$, $U_{\text{target}}$, $u_0$)
  end

  procedure ReceiveMigrationRequest($u_0$: Node; $U_{\text{target}}$: Path)
  begin // executed by $u_{\text{target}}$
    this.$u_0$ = $u_0$
    this.$U_{\text{target}}$ := $U_{\text{target}}$
    state = receiving
    SendMigrationAck($u_0$)
  end

  procedure ReceiveMigrationAck()
  begin // executed by $u_0$
    state := migrating
    InitSendDataSubset($u_{\text{target}}$, $U_{\text{target}}$)
    SetTimer(pathRecomputationTimer, $\max\{0, \tau(U_{\text{target}}) - \Delta t_{\text{path}}\}$)
  end

  procedure TimerExpired()
  begin // executed by $u_0$
    RecomputePaths()
  end

  procedure PathChangeNotification($U_{\text{target}}$: Path)
  begin // executed by $u_0$
    this.$U_{\text{target}}$ := $U_{\text{target}}$
    SetTimer(pathRecomputationTimer, $\max\{0, \tau(U_{\text{target}}) - \Delta t_{\text{path}}\}$)
  end

  procedure ReceiveData($o_i$: Object)
  begin // executed by $u_{\text{target}}$
    ExecuteUpdate(storage, $o_i$) // executed while $u_{\text{target}}$ is in receiving state
    SendDataAck($u_0$, $o_i$)
    if DataComplete() then
      state := active
      InitServerAdvertisement()
    end if
  end
```

```
53
54   procedure ReceiveDataAck($o_i$: Object)
55   begin // executed by $u_0$
56     if DataTransferComplete() then
57       UninitServerAdvertisement()
58       state := retired
59     end if
60   end
61
62   procedure ReceiveUpdate($o_i$: Object)
63   begin // executed by $u_0$ and $u_{\text{target}}$
64     if state = active then
65       ExecuteUpdate(storage, $o_i$) // executed by $u_0$
66     else if state = migrating then
67       ExecuteUpdate(buffer, $o_i$) // executed by $u_0$
68       ForwardUpdate($u_{\text{target}}$, $o_i$)
69     else if state = retired then
70       ForwardUpdate($u_{\text{target}}$, $o_i$) // executed by $u_0$
71     else if state = receiving then
72       // this case cannot occur
73     end if
74   end
75
76   procedure ReceiveForwardedUpdate($o_i$: Object)
77   begin // executed by $u_{\text{target}}$
78     ExecuteUpdate(storage, $o_i$)
79     SendUpdateAck($u_0$, $o_i$)
80   end
81
82   procedure ReceiveUpdateAck($o_i$: Object)
83   begin // executed by $u_0$
84     ExecuteUpdate(storage, GetObject(buffer, $o_i$))
85   end
86
87   procedure ReceiveQuery($Q$: Query)
88   begin
89     if state = active or state = migrating then
90       ExecuteQuery(storage, $Q$) // executed by $u_0$
91     else if state = retired then
92       SendQuery($u_{\text{target}}$, $Q$) // executed by $u_0$
93     else if state = receiving then
94       // this case cannot occur
95     end if
96   end
97 end
```

Lines 62-96 of Listing 4.2 contain the relevant pseudocode that deals with the handling of requests, that is, updates and queries, while a migration is in progress. This part is responsible for guaranteeing that eventual consistency according to Definition 3.1 is maintained. Recall that for eventual consistency to hold, the most recently updated value on a data item will eventually be returned if for a long time no updates occur in the system.

The way requests are handled essentially depends on the current state that the source and target server are in while receiving updates (line 62) and queries (line 87). Let us first consider the handling of updates. While u_0 is in *active* state (line 64), it accepts any incoming updates from remote nodes and processes them locally (line 65). When an update occurs on a data item that is received while u_0 is in *migrating* state (line 66), then that update is put in a local buffer (line 67) and forwarded to the target server as well (line 68). Note that the data item is not updated on the local storage of the source server. Any data update received by u_{target} is immediately written to the local data storage (line 78) of that server. After that, the target server sends an acknowledgement for that data item back to the source server (line 79). Once an acknowledgement of a data item is received at the source server (line 82), the update can be moved from the buffer to the persistent storage (line 84).

From the point where the source server is in *retired* state (line 69), it forwards updates immediately to the target server without further processing (line 70). Finally, note that while u_{target} is in *receiving* state (line 72), it does not receive any requests other than those forwarded from u_0. This is because it does not yet disseminate advertisements (Section 3.3.1), which are required for client requests to locate u_{target} in the first place.

Queries, on the other hand, are received through the ReceiveQuery procedure in line 87. In both *active* and *migrating* state (line 89) a query received at the source server u_0 is immediately evaluated based on the local storage (line 90). In *retired* state (line 91), queries are forwarded by the retired server to the now current server (line 92). Note that as in the case of updates, no queries are received during *receiving* state (line 93).

With respect to eventual consistency, the handling of updates in lines 62-85 guarantees that no update is lost once made persistent in the local storage of any data server. While it is trivial to achieve eventual consistency when a server is in *active* or *retired* state, the situation is more complicated while a migration is in progress. The following problem occurs in the event where a data update that is forwarded from the source to the target server is lost.

Assume that a data update is received at u_0 while a migration is in progress (line 66). Assume further that instead of executing the update on the buffer of u_0, as shown in line 67, it is immediately written to the local storage instead. Next, let us assume that the data item, which is forwarded in line 68, is lost on the way to the target server u_{target} while migration is in progress. Note that this is allowed according to the system model we stated in Section 3.1. Let us finally assume that the migration of the data subset eventually succeeds, that is, the target server transits to *active* state, while the source server drops the data subset and goes to *retired* state. In this event, the original update on the data item is lost. If no further updates on this object occur, this update was the most recent one, hence, eventual consistency is violated.

To avoid this situation, data migration first updates a request in an intermediary buffer in line 67. Only when the source server has received the update's corresponding acknowledgement in line 82 from the target server, it makes the update persistent in its local storage (line 84). In case a forwarded data update is lost between the source and the target server, that update is never made persistent due to the absence of a corresponding acknowledgement.

Observe that for the above arguments to hold, node failures must not occur, which is what we have assumed in the system model in Section 3.1. The handling of node failures in conjunction with data replication will be discussed briefly in the outlook in Section 6.2.1.

4.5.2 Data Consolidation

In the situation where a network partition occurs during migration, it is possible under certain circumstances that the two servers involved in the migration process cannot definitely decide on which of the two is to remain the only one responsible. The critical situation occurs when u_{target} has just moved into *active* state, but u_0 will not receive all of u_{target}'s data acknowledgements due to both now being located in different network partitions. Because u_0 has no way to find out about u_{target}'s decision, it must eventually return to *active* state in order for the data subset to not become lost and to retain eventual consistency. As a result, two servers remain that are both in *active* state and which handle the same data subset. Thus, both servers disseminate server advertisements independently, which are used by the core storage's request forwarding algorithm (Section 3.3.2). This situation is shown in Figure 4.6.a for two servers u_1, u_2.

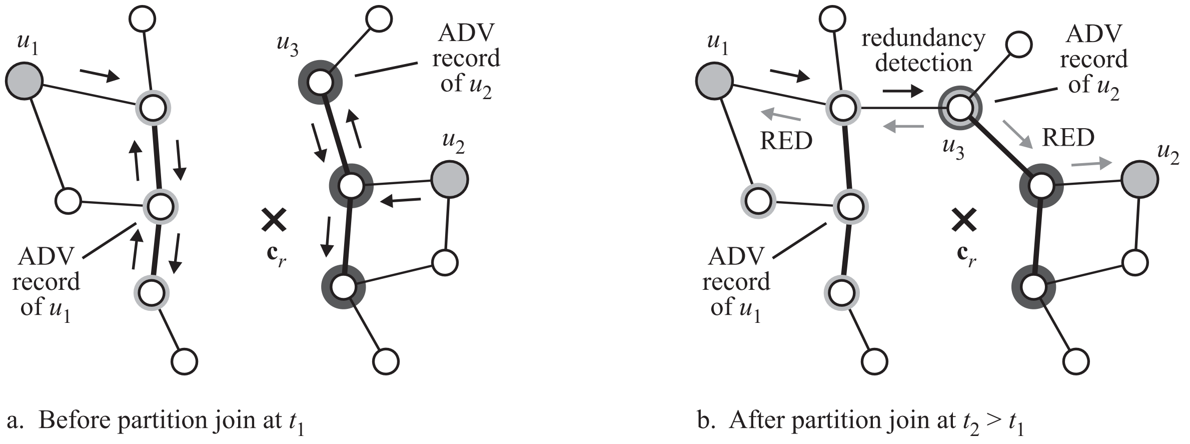


Figure 4.6: Occurrence and detection of a server redundancy.

Due to the increased management cost incurred by multiple servers and unwanted data inconsistencies that may result from multiple versions of the same data subset, it is highly desirable to resolve such redundancies as early as possible. For this purpose, the migration mechanism implements a complementary data consolidation algorithm that merges multiple data subsets back into a single one once a redundancy has been detected.

In the situation of a partitioned network, the earliest possible time to become aware of a server redundancy is upon the rejoining of two network partitions. Note that we assume in the system model in Section 3.1 that partitions always join after a finite amount of time. The following redundancy detection scheme, which is an extension to the server advertisement mechanism described in Section 3.3.1, is able to efficiently detect multiple active servers.

Let us assume that initially, two servers u_1, u_2 exist in different network partitions (Figure 4.6.a). Advertisements from both servers are confined to the respective network partition. Once these partitions join (Figure 4.6.b), advertisements of both data servers will eventually coincide for the first time at some network node. Note that this is guaranteed by the way server advertisements are distributed (Section 3.3.1). In Figure 4.6.b, this is node u_3, which already stores an ADV record of u_2 and which has just received u_1's first advertisement information.

As an extension to the server advertisement mechanism introduced in Section 3.3.1, we assume that advertisements are always disseminated in *redundancy detection mode*. When a server advertisement is received by a node in this mode, that node queries for any other existing advertisement information it might store. Each existing record is checked for matching identifiers of the data subsets and reference coordinates, but for different server identifiers. This condition uniquely defines the existence of a server redundancy. In that event, the detecting node sends a *redundancy notification* message to both servers. This is shown in Figure 4.6.b, where node u_3 sends a notification (RED) to both u_1 and u_2. The decision to send the notification to both servers is to increase the chance to quickly detect a redundancy. Once the redundancy notification is sent, the advertisement continues to propagate. However, the advertisement does not propagate in redundancy detection mode anymore, in order to have only a single redundancy detection per advertisement cycle and give servers sufficient time to handle it.

We now describe the algorithm by which two data servers initiate and handle the consolidation of multiple versions of the same data subset. The algorithm's pseudocode is shown in Listing 4.3. We assume that, initially, all servers are in *active* state (line 4), and that u_i, u_j denote two servers whose data subsets need to be merged. The consolidation process is triggered by the reception of any redundancy notification message at a data server u_i (line 9). The notification contains the ID of the other server u_j. The redundancy notification is only considered by the receiving server u_i if this server is in *active* state (line 11). In any other state, either a redundancy is already being handled or a migration is taking place. For reasons that we discuss later on, a redundancy is always handled by the server with the larger ID, which we refer to as the higher-ID server, in contrast to the lower-ID server. If the receiving server has the smaller ID (line 12), therefore, it sends the redundancy notification to its higher-ID partner (line 13). Otherwise, server u_i requests consolidation from the lower-ID server (line 15).

Listing 4.3: Data consolidation.

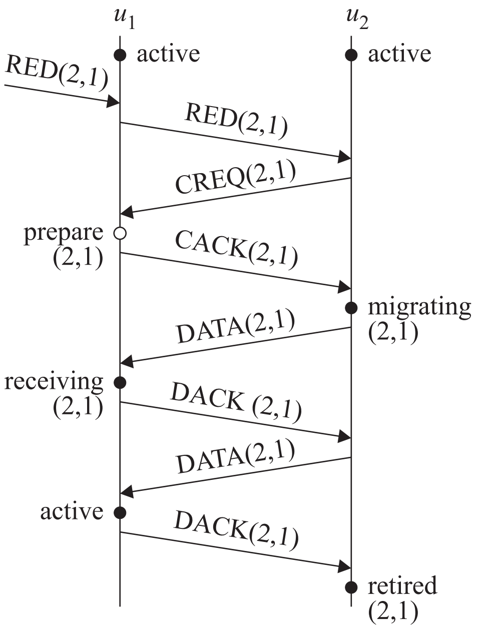
```
1  module DataConsolidation
2  begin
3    var $u_i$: Node // own ID
4    var state: MigrationState := active
5    var states: Set of State
6    var storage: Set of Object
7    var buffer: Set of Object
8
9    procedure ReceiveRedundancyNotification($u_j$: Node)
10   begin // the detected redundancy is between $u_i$ and $u_j$
11     if state = active then
12     if i < j then // the server with the largest ID must handle the redundancy
13       SendRedundancyNotification($u_j$, $u_i$)
14     else // $i > j$
15       SendConsolidationRequest($u_j$, $u_i$)
16     end if
17   end
18
19   procedure ReceiveConsolidationRequest($u_j$)
20   begin
21     if state = active then // accept any consolidation request while in active state
22       Add(states, {prepare, $u_i$, $u_j$})
23       SendConsolidationAck($u_j$, $u_i$)
24     end if
25   end
26
```

```
27    procedure ReceiveConsolidationAck($u_j$)
28    begin
29      if state = active then
30        Clear(states) // remove possibly existing prepare states
31        state := {consolidating, $u_i$, $u_j$}
32        InitSendDataSubset($u_j$)
33      end if
34    end
35
36    procedure ReceiveData($u_j$, $o_i$)
37    begin
38      if Exists(states, {prepare, $u_j$, $u_i$}) then // the first server gets its turn
39        Clear(states)
40        state := {consolidating, $u_i$, $u_j$}
41        ReceiveData($u_j$, $o_i$) // call recursively
42      else if state = {receiving, $u_i$, $u_j$} then
43        ExecuteUpdate(buffer, $o_i$)
44        SendDataAck($u_j$, $o_i$)
45        if DataComplete() then
46          for each $o_i$ in buffer do
47            ExecuteUpdate(storage, GetObject(buffer, $o_i$))
48          end for
49          state := active
50        end if
51      end if
52    end
53
54    procedure ReceiveDataAck($u_j$: Node; $o_i$: Object)
55    begin
56      if DataTransferComplete() then
57        state := retired
58      end if
59    end
60
61    procedure DataTransferTimeout()
62    begin
63      TerminateDataTransfer()
64      state := active
65    end
66  end
```

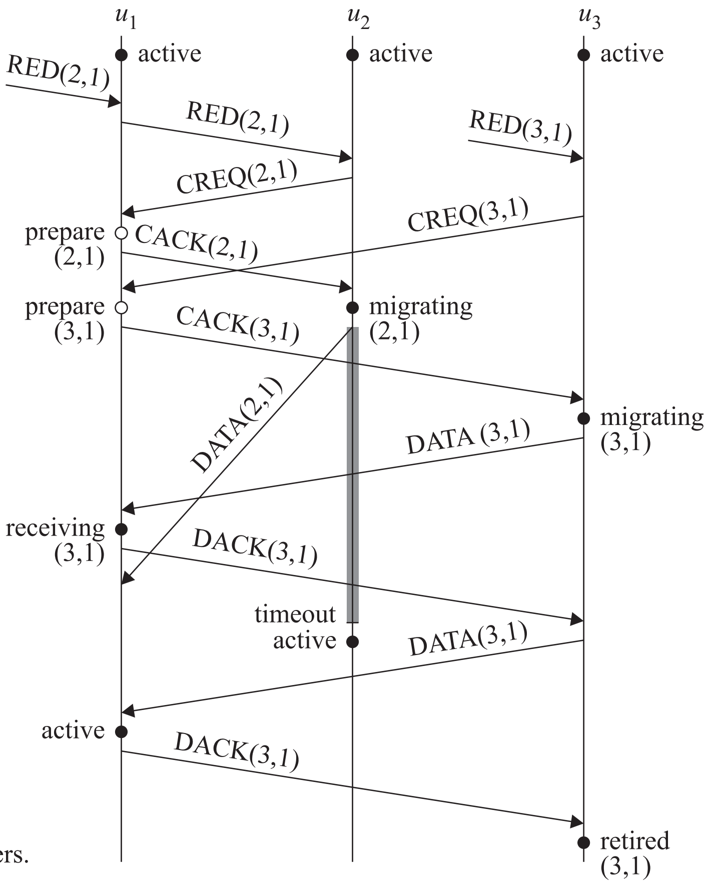
The next step is to negotiate the initiation of the consolidation process (lines 19-34), during which the involved servers must be in *active* state (line 21 and 27). While negotiation works similar to the migration process in Listing 4.2, lines 18-31, data consolidation additionally utilizes an auxiliary set of *prepare* states (line 5). Note that for ease of exposition, we have not considered path recomputation, which works in analogy to the migration process. When a consolidation request from the higher-ID server is received at the lower-ID server (line 19), the latter stores a *prepare* state for the corresponding server pair (line 22). This state essentially makes the lower-ID server aware of a potentially following data transfer from a higher-ID server. The lower-ID server immediately acknowledges the request (line 23). Upon the reception of the acknowledgement at the higher-ID server (line 27), that server clears possibly existing *prepare* states (line 30), since it now gets involved in a data transfer (line 31) and cannot accept any other concurrent requests. The higher-ID server then initiates the sending of the data subset asynchronously (line 32), just like in the case of a migration in Listing 4.2, line 29.

The actual data transfer takes place between lines 36 and 59. When a data item is received at the lower-ID server (line 36), it is first checked whether a *prepare* state exists for the higher-ID server that sent the item (line 38). If this is the case, the transfer is initiated by clearing all auxiliary *prepare* states (line 39), switching to *receiving* state (line 40), and calling procedure ReceiveData recursively (line 41). Since the lower-ID server is now in *receiving* state (line 42), it executes (line 43) and acknowledges the update (line 44). In contrast to migration, the data item is updated on an intermediary buffer, because the receiving server also has a data subset in the storage. For simplicity, we omit the details of how a server is able to handle requests on the storage while data is received during a consolidation, which can easily be distinguished. When the data subset is complete (line 45), the buffer will be merged with the local storage (lines 46-48) and the server moves back into *active* state (line 49). For each received acknowledgement at the higher-ID server (line 54), that server checks whether all data items have been acknowledged (line 56). If yes, the server retires (line 57), which completes the transfer. If an outstanding acknowledgement is not received by the higher-ID server after some limited amount of time, a data transfer timeout will occur (line 61). In this case, the currently processed migration will be terminated (line 63) and the server assumes *active* state again (line 64).

While the consolidation of a data subset always takes place between two servers at a time, redundancies between three and more servers may also occur. The algorithm in Listing 4.3 supports such situations by breaking ties so that two servers will eventually enter a consolidation



a. Consolidation of two servers.



b. Breaking ties between three servers.

Figure 4.7: Data subset consolidation.

process. This is made possible by the previously mentioned decision to always choose the higher-ID server to take the lead in a consolidation. This way, no circular consolidation requests can occur, which is a sufficient condition to break any ties between three or more servers.

We illustrate some typical cases in which ties are broken in the communication diagrams shown in Figure 4.7 and 4.8. Figure 4.7.a depicts the normal case where two servers perform the consolidation process according to the algorithm in Listing 4.3. By RED(i, j), CREQ(i, j), CACK(i, j), DATA(i, j), and DACK(i, j) we denote a redundancy notification, consolidation request, consolidation acknowledgement, data, and data acknowledgement message, respectively. In all figures, only two data messages are required to transfer the full data subset.

In Figure 4.7.b we show a situation where a tie is broken based on a first come, first served basis at the lower-ID server. Observe that two redundancy notifications are received at u_2 and u_3, indicating a redundancy between u_2 and u_1 on one side, and u_3 and u_1 on the other side. Two consolidation requests are subsequently sent from u_2 and u_3 to u_1. As a consequence, u_1 buffers two *prepare* states, one for each server. After u_2 has received the consolidation acknowledgement, its first data message is delayed by the network. In this situation, the first data message sent by u_3 overtakes the other data message and u_3 gets its chance, which is reflected by u_1 entering *receiving* state. Eventually, a timeout occurs at u_2, which simply moves back into *active* state. In the meantime, u_1 and u_2 continue the consolidation process until the end.

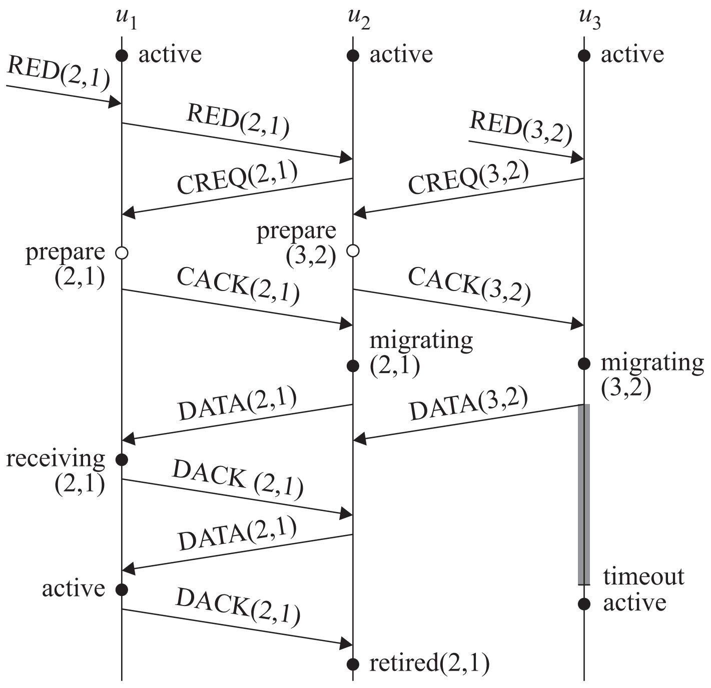


Figure 4.8: Concurrent data subset consolidation.

Figure 4.8 illustrates the case where a tie is broken based on an override of a *prepare* state by a *migrating* state. The diagram shows how two redundancies between servers u_1, u_2 on one side and u_2, u_3 on the other side are concurrently detected by u_2 and u_3, respectively. After the corresponding consolidation requests have been received at u_1 and u_2, both servers temporarily buffer a *prepare* state. Once the consolidation acknowledgements have been returned, u_2 and u_3 both move into *migrating* state. At this time, u_2 has deleted the existing *prepare* states according to line 30 in Listing 4.3. Hence, the *migrating* state overrides any *prepare* state. From this point forward, u_1 and u_2 carry out the consolidation process until its completion. However, server u_3 still sends its first data message to u_2, but which will be ignored by u_2 due to another previously confirmed migration. Because u_2 does not send back a corresponding data acknowledgement, a timeout will eventually occur at u_3, which will move back into *active* state. Note that for ease of exposition, we have omitted the possibility of sending negative acknowledgements in both Figure 4.7.b and 4.8. Such messages can be used to provide explicit feedback for those servers whose migration request is denied.

4.6 Performance Analysis

In this section we examine in detail the performance of the data migration approach by means of a comparative experimental analysis. We have implemented two variants of data migration, which employ source routing and geometric routing, and which we denote by DataMiP (source routing) and DataMiP (BPR), respectively.[1] In addition, we have implemented two further approaches, termed greedy migration and progressive migration.

DataMiP (source routing) designates the migration approach that relies on network paths that are computed by the algorithm in Listing 4.1. According to this approach, a source route is updated during a migration process whenever the previous one is about to weaken. This migration approach is shown in Listing 4.2. In the *DataMiP (BPR)* variant, the stability analysis in Section 4.4.2 is also performed during the determination of a target node. The difference to DataMiP (source routing) is that the output path of the algorithm in Listing 4.1 is omitted during migration. Instead, geometric routing based on bidirectional perimeter routing (BPR, Section 3.2) is used for the migration of the data subset.

In *greedy migration*, once the current server reaches the threshold distance d_{thresh}, greedy routing is used as the means for network topology exploration to find a suitable target node. A request is sent by the current server into the direction of the reference coordinate until a node is reached at which greedy routing fails. BPR is then used as the underlying routing protocol to migrate data to the selected target node. *Progressive migration* is also initiated upon reaching d_{thresh}. In contrast to greedy migration, the selection of a target server is based solely on the set of known neighbors of the current server. From this set, the neighbor is chosen that is located closest to the reference coordinate, but only if a node exists that is nearer to the reference coordinate than the current server. Upon completion of migration to the selected node, subsequent iterations of progressive migration follow where data is migrated to nodes located even closer to the reference coordinate. This procedure continues until no further node can be determined. Both greedy and progressive migration are re-initiated once d_{thresh} is reached again.

The following system parameters, which are also shown in Table 4.1, are applicable in the evaluation of the migration approaches. The general system parameters that are assumed in all experiments presented in this dissertation are listed in Table 3.3.

A total simulation time of 3600 seconds was chosen, corresponding to an increase by a factor of 10 compared to the evaluation of core data storage (Section 3.5.2). This is necessary to achieve good statistical values, because migrations occur at a much lower frequency than requests in most of the simulations. As in the case of core data storage, only a single reference coordinate is placed in the center of the simulation area, with a default size of $200 \cdot 200$ m^2. No requests are processed in order to avoid possible side effects on the migration performance. Furthermore, server advertisements are sent with a rate of 1/second.

The default number of data items is increased also by a factor of 10 compared to core data storage and set to 10000. This value corresponds to 320 kB of data for each migration, assuming 32 bytes per data item. We assume a data packet capacity of 1500 bytes, thus approximately 47 data items can be carried in a single data packet. The threshold distance d_{thresh}, which is applicable in all four discussed approaches, is set to 100 m. The critical distance d_{crit}, which is only applicable in the two DataMiP variants, is set to 150 m.

[1]DataMiP standing for Data Migration Protocol.

System Parameter	Default Value
Simulation time	3600 s
Number of cells	1, centered at (x,y) = (300,300)
Cell size	$200 \cdot 200$ m^2
Number of requests	0 (no requests)
LCS server advertisement interval	1 s
Number of data items	10000
Total data size	320 kB (32 bytes / data item)
Data packet capacity	1500 bytes $\cong$ 47 data items
Threshold distance d_{thresh}	100 m
Critical distance d_{crit}	150 m $= 1.5 \cdot d_{\text{thresh}}$
Speed ratio	1:5
Speed of low-mobile nodes	1.5 m/s (20% of nodes)
Speed of high-mobile nodes	15 m/s (80% of nodes)

Table 4.1: Data migration: system parameters.

In order to assess the susceptibility of the migration decision policy in Section 4.4 to different node speeds, we partition the overall set of nodes into two sets by a speed ratio of 1:5. According to this ratio, 20% of the nodes are parameterized with a speed of 1.5 m/s, and 80% of the nodes are parameterized with 15 m/s. All nodes maintain motion according to the random waypoint mobility model [BMJ$^+$98] with a pause time of 30 s.

4.6.1 Performance Metrics

We consider four performance metrics, which are averaged over the total simulation time: spatial coherence, migration efficiency, migration duration, and migration robustness.

Spatial Coherence

Spatial coherence follows directly from Definition 2.3 and quantifies the ability of each migration approach to maintain proximity between a data subset (that is, the active data server) and the associated reference coordinate. Hence, the mean spatial coherence is defined as the mean geometric distance of the current server from the reference coordinate over the simulation time. While a migration is in progress, spatial coherence is calculated with respect to the migrating server, because this server continuous to receive requests.

Migration Efficiency

Migration efficiency quantifies the communication cost that are required by migration processes over time. It is defined in units of mean aggregated packet size per unit time. It includes all traffic that is incurred by migration processes, including network topology exploration.

Migration Duration

The duration of a single migration is defined as the time from the initiation of a migration to the switching to *retired* state at the migrating server. The aggregated migration duration is defined as the sum over all durations of individual migrations over the simulation time. We define the *migration duration fraction* as the aggregated migration duration divided by the total simulation time. For example, a migration duration of 0.1 corresponds to the situation where migrations occur during 10% of the total simulation time.

Migration Robustness

Migration robustness describes how many of the total number of initiated migrations can be successfully completed. We quantify the robustness of migrations by distinguishing two metrics for the assessment of migration failures. The number of *recoverable* migration failures counts those migrations that fail, but where the involved servers can deterministically agree on which of the servers shall remain responsible for the corresponding data subset. The number of *fatal* migration failures counts only those failures where both servers will move into *active* state, because they are unaware of the other server's decision. This kind of failure is especially critical because it always leads to redundant data subsets that need consolidation (Section 4.5.2). In contrast to both DataMiP variants, greedy and progressive migration do not possess the ability to consolidate multiple data subsets.

4.6.2 Spatial Coherence

Before we discuss the results of spatial coherence in Figure 4.9 through 4.12, it is important to note that the magnitude of the achieved coherence is tunable by adjusting parameters d_{thresh} and d_{crit} (Section 4.2). Therefore, comparing different approaches in quantitative terms is limited. Rather, we will focus on qualitative interpretations and on whether a sufficient level of spatial coherence can be maintained. The details on how to set algorithm parameters to achieve *specific* magnitudes for spatial coherence is out of the scope of this dissertation.

Figure 4.9 shows spatial coherence as a function of the threshold distance. We observe that DataMiP's coherence is noticeably larger than that of progressive migration below a threshold distance of approximately 50 m. The best values for progressive migration and DataMiP (source routing) are 25 m (at $d_{\text{thresh}} = 20$ m) and 45 m (at $d_{\text{thresh}} = 60$ m), respectively. The key observation is that both values are well within a single communication range, which is 100 m in the simulations. This implies that in both cases the degrees of spatial coherence are virtually equivalent in terms of routing performance, because a packet reaching the vicinity of the reference coordinate will reach the data server in only one hop in both cases.

The second observation in the lower range of threshold distance is that the value of the spatial coherence achieved by DataMiP increases. At first glance, this might look like a problem with DataMiP. However, based on the argumentation of virtually equivalent routing performance in the previous paragraph, DataMiP exploits this range by taking a deliberate decision in the migration recommendation policy (MRP) that can be explained as follows. In the figure, we have used $d_{\text{crit}} = 1.5 \cdot d_{\text{thresh}}$ for all settings of the threshold distance. Thus, for small values of d_{thresh}, when the MRP first recommends a migration according to Equation (4.1), no migration will occur, because a node that is more eligible than the current data server will not be found. However, after several more cycles of policy evaluation, the server will eventually reach d_{crit}, in which case migration priority $\Pi = 1$ according to Equation (4.6). From the point of view of the current server, migration is thus necessary, and will be performed to another node that may be even less eligible in terms of the geometric distance. Therefore, the graph is an important indicator in determining the operational domain in terms of d_{thresh}. As a simple rule, for $d_{\text{thresh}} \approx r_{\text{tx}}$, there is a always a good chance to find a fair amount of candidate nodes.

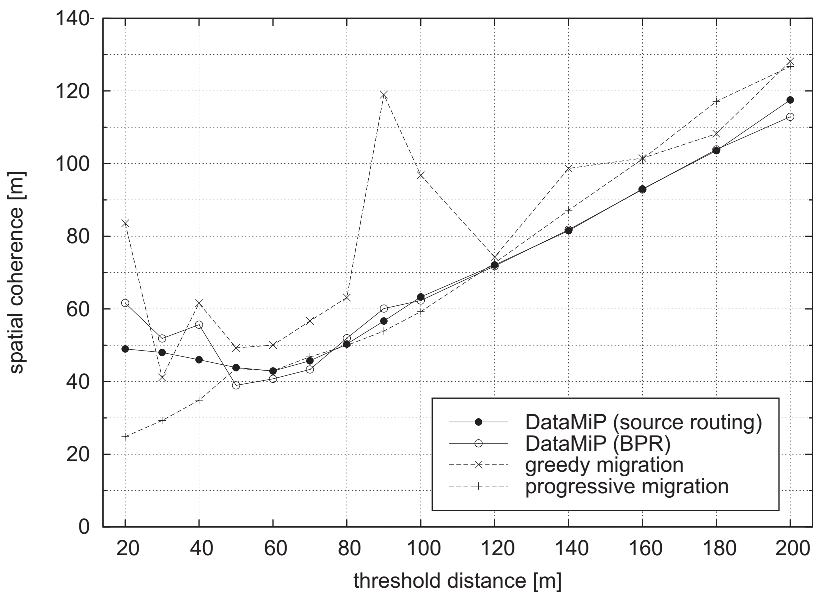


Figure 4.9: Spatial coherence as a function of the threshold distance d_{thresh}.

Finally, we can observe a rather erratic behavior of greedy migration both in the lower range of threshold distances, and between threshold distances of 80 m and 120 m. In contrast, both DataMiP variants, specifically DataMiP (source routing), show a very stable trend. The irregular behavior is characteristical for approaches that employ geometric routing. From progressive migration we could see previously that spatial coherence can be maintained at around 25 m. This indicates that nodes exist even closer to the reference coordinate, since 25 m is the mean value. Between 80 m and 120 m, therefore, a transition from one-hop to two-hop greedy migration occurs. Until the occurrence of this transition, greedy routing will attempt to bridge a single hop over an increasing geometric distance during migration. The target node selection thus becomes worse in terms of path stability, and a larger fraction of migrations will fail. This leads to the situation where migrations will occur at a later point in time, thus increasing mean spatial coherence. This situation is improving once more two-hop migrations occur in greedy routing, leading to more stable paths again. We will encounter this behavior for both greedy and progressive migration also in some of the following simulations. In the case of DataMiP, the source routing-based variant performs better than the BPR variant due to the using of a more stable path that is based on the stability analysis in Section 4.4.2.2.

Figure 4.10 shows the spatial coherence as a function of the number of nodes. In the lower range of node densities we can observe that DataMiP (source routing) does not achieve the spatial coherence of the other approaches. From the viewpoint of the MDP, changing the number of nodes while maintaining the threshold distance is virtually equivalent to changing the threshold distance instead and maintaining the number of nodes, which was done in Figure 4.9. In both cases, the number of candidate nodes that the MDP considers in the selection of a target server effectively decreases. We can therefore attribute the performance degradation of DataMiP to the deliberate decisions in the MDP, as discussed with respect to Figure 4.9.

We further observe that DataMiP (BPR) has slightly better performance than DataMiP (source routing). This is because the source route employed by DataMiP (source routing) generally

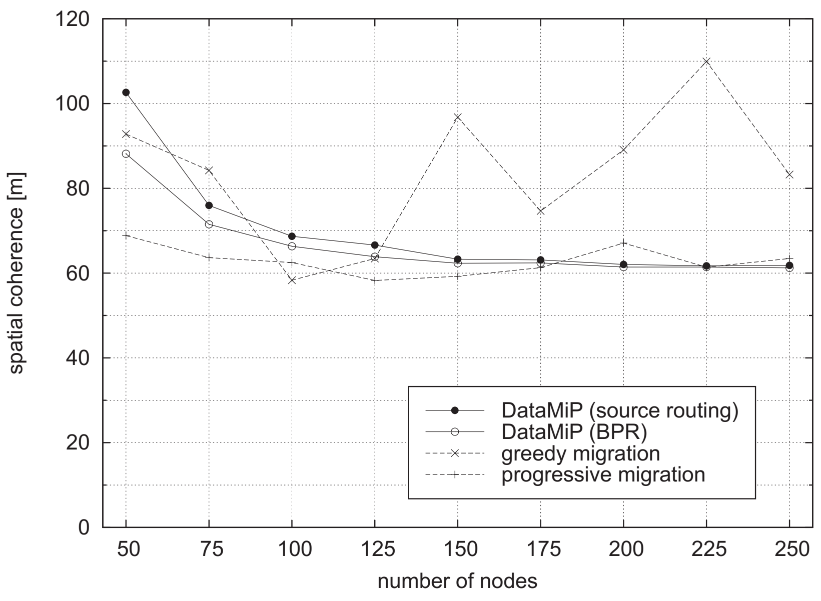


Figure 4.10: Spatial coherence as a function of the number of nodes.

involves a larger number of hops than the greedy route selected by DataMiP (BPR). This incurs a slightly larger migration duration (Figure 4.14, right graph) during which the magnitude of spatial coherence remains at a larger value, thus leading to a stronger increase in the mean spatial coherence of DataMiP (source routing). Regarding greedy migration we observe once more a deteriorating performance with an increasing number of nodes. This is due to the characteristics of greedy routing, as discussed in conjunction with Figure 4.9.

Figure 4.11 and 4.12 show spatial coherence as a function of the number of data items. Specifically, Figure 4.11 depicts the results for a maximum node speed of 15 m/s, the default value in our experiments. The graphs indicate that for any considered data size of 160 kB and below, all of the four approaches perform in a similar way.

In the region above about 640 kB, both DataMiP variants perform significantly better than the other mechanisms. The reason is related to a combination of two facts. Firstly, greedy and progressive migration do not differentiate between slow and fast moving nodes. Therefore, migrations occur more often (Figure 4.21). As a consequence, a data server will contribute to spatial coherence while being located further away from the reference coordinate for a larger number of migrations. The second fact is directly related to the increasing number of data items. Because the larger the data subset to be migrated, the longer the duration of an individual migration. Multiplied by the frequency of migrations and considering the previously given argument, the overall migration duration will increase more significantly for greedy and progressive migration (Figure 4.16). This leads to the fact that large values of spatial coherence are contributed to the mean spatial coherence over a longer period of time. This argumentation emphasizes that it is key to decrease as much as possible the number of migrations, which the MDP achieves by its preferring of slow over fast moving nodes. Note that the speed ratio is 1:5, thus only 20% of the nodes are moving at a low speed, which is sufficient for the MDP to provide a significant performance gain.

Figure 4.12 shows the same scenario as Figure 4.11, with the difference that the maximum node

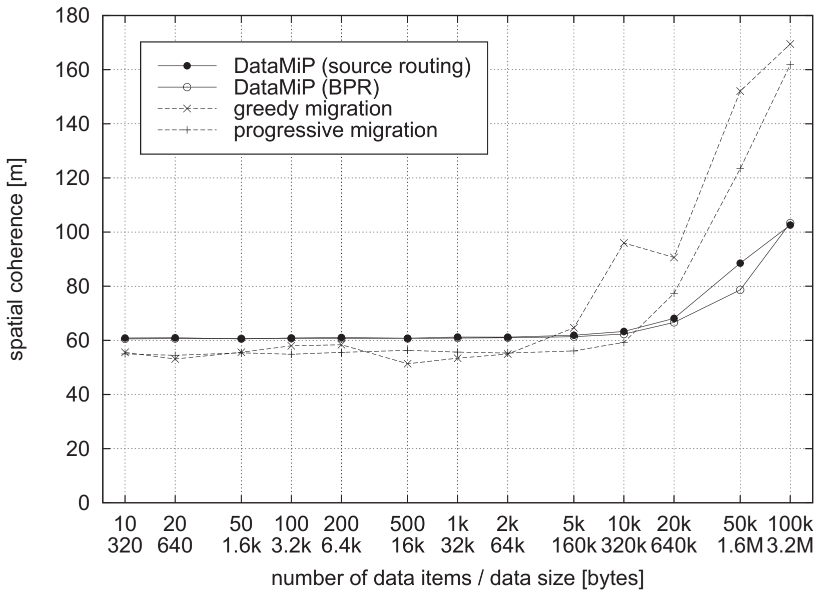


Figure 4.11: Spatial coherence as a function of the number of data items (maximum node speed: 15 m/s).

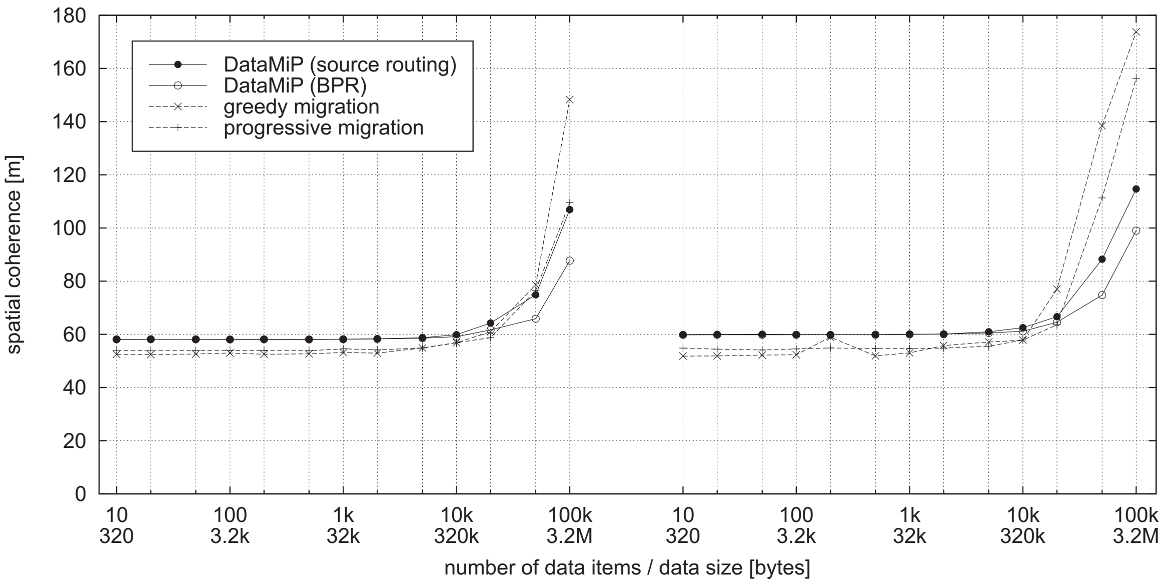


Figure 4.12: Spatial coherence as a function of the number of data items (maximum node speed: 5 m/s (left) and 10 m/s (right)).

speed in the speed ratio is 5 m/s (left) and 10 m/s (right). The results confirm the positive influence of DataMiP's MDP policy on spatial coherence, which increases with the speed of nodes. For lower speeds, the performance gain is smaller because the MDP's exploitation potential is smaller for nodes moving at 5 m/s than those moving at 10 m/s and 15 m/s. However, at 10 m/s the advantage of the MDP is already significant.

4.6.3 Migration Efficiency and Duration

Figure 4.13 through 4.18 show the results for migration efficiency and duration. Both performance metrics are presented side by side to point out their strong correlations. Figure 4.13 shows migration efficiency (left) and migration duration (right) as a function of the threshold distance. We observe that greedy and progressive migration have an advantage in the lower range of threshold distance. The argumentation follows that of Figure 4.9 with respect to the operational domain of DataMiP, which is a calibration issue only.

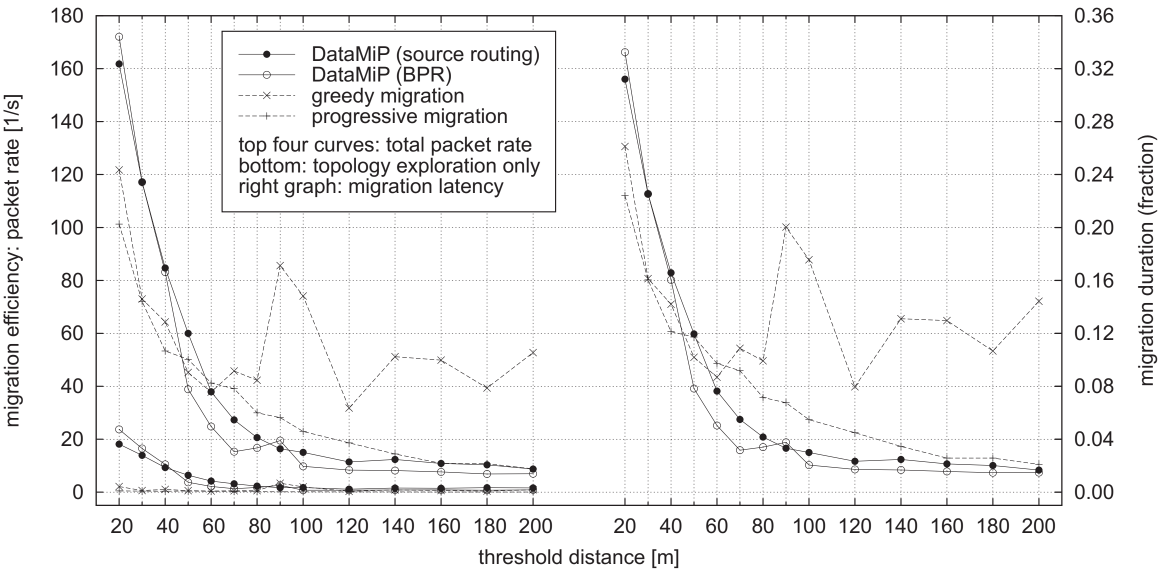


Figure 4.13: Migration efficiency and duration as a function of the threshold distance.

The advantage of DataMiP over greedy and progressive migration becomes apparent when considering the different gradients in the slopes of each curve. At a threshold distance of roughly 60 m and above, both the migration efficiency and duration of DataMiP are better than that of greedy and progressive migration. The steep slope of DataMiP can be explained using the following arguments. For small threshold distances, DataMiP is forced to perform a migration even when a more suitable server does not exist at all. This fact can be observed well by the large additional packet rate incurred by network topology exploration (left, bottom curves), which is present in both DataMiP variants. With increasing threshold distance, the set of candidate nodes from which an optimal node can be chosen increases accordingly, which leads to two situations. Firstly, a sufficient number of candidate nodes become available, such that target nodes that are more suitable than the current server can be determined in the first place. Second, the growing candidate node set will contain more nodes that move at a lower speed, and the MDP will prefer these nodes over the fast moving ones. Both effects add up and lead to a steeper slope than for greedy and progressive migration.

Figure 4.14 shows the migration efficiency and duration as a function of the number of nodes. We observe that DataMiP's migration efficiency increases while its duration decreases. This occurs because for a growing number of nodes, DataMiP's MDP is able to select suitable nodes based on a larger candidate node set. The steep slope in the lower range of node densities is due to the previously discussed effects in Figure 4.13.

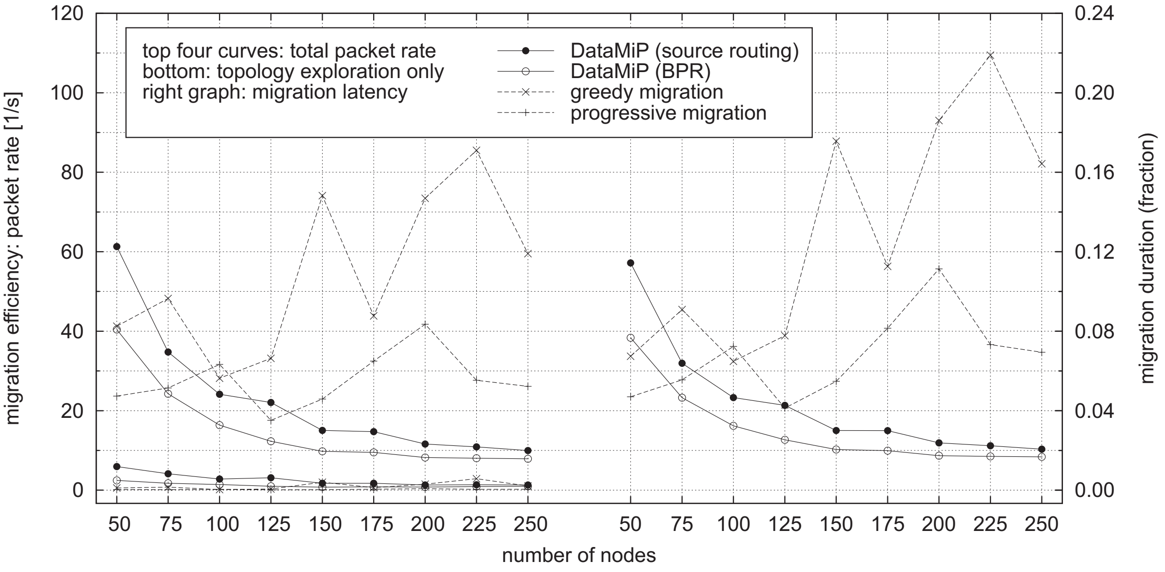


Figure 4.14: Migration efficiency and duration as a function of the number of nodes (maximum node speed: 15 m/s).

Both the graphs on the left and right side of Figure 4.14 further illustrate well the negative impact that a larger number of nodes has on the performance of greedy and progressive migration. The deterioration in performance is again due to the greedy routing problem associated with high node densities, as discussed in Figure 4.9 and also Figure 3.26.

Figure 4.15 presents the migration efficiency also for the maximum node speed of 5 m/s (left) and 10 m/s (right), which complements the right side of Figure 4.14. Observe the naturally larger overhead of network topology exploration at low node densities and higher node speeds due to the larger number of migrations required in both cases. For DataMiP, the gradient of the slope increases from 5 m/s to 10 m/s. This observation confirms that the stronger gradient is due to MDP's ability to better exploit large differences in node speed. The exploitation potential of higher node speeds by the MDP can be well observed as follows. Consider the migration efficiency at 50 nodes in Figure 4.15 (left) and 4.14 (left). From 5 m/s via 10 m/s to 15 m/s, the packet rate of DataMiP (source routing) increases by 20 packets/s in each step. In contrast, at 150 nodes, the packet rate increases only insignificantly. This confirms the exploitation potential of the MDP at higher node densities, where based on a sufficiently large candidate node speed, faster moving nodes can be skipped in the selection.

Next, increasing the node speed from 5 m/s to 15 m/s shows clearly the decrease in performance of greedy and progressive migration. This is because higher node speeds lead to neighbor tables becoming outdated more quickly. In conjunction with the greedy forwarding problem discussed in Figure 4.9, this means that especially those nodes that are selected as the next hop in greedy forwarding are, in many cases, not reachable anymore. Furthermore, greedy and progressive migration do not consider the different speed of nodes in their selection of target nodes, thus more frequently select fast moving nodes over slow moving nodes.

Figure 4.16 shows migration efficiency and latency as a function of the number of data items. In the small range of the number of data items, all approaches incur no significant overhead in the network. This is due to the fact that the migration of a very small number of data

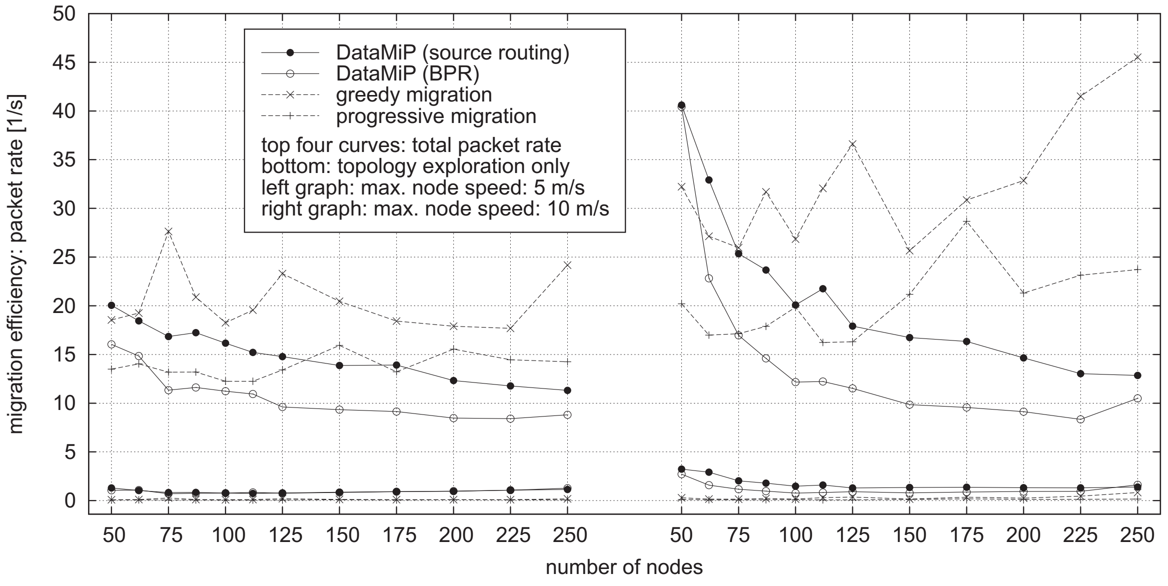


Figure 4.15: Migration efficiency as a function of the number of nodes (maximum node speed: 5 m/s (left) and 10 m/s (right)).

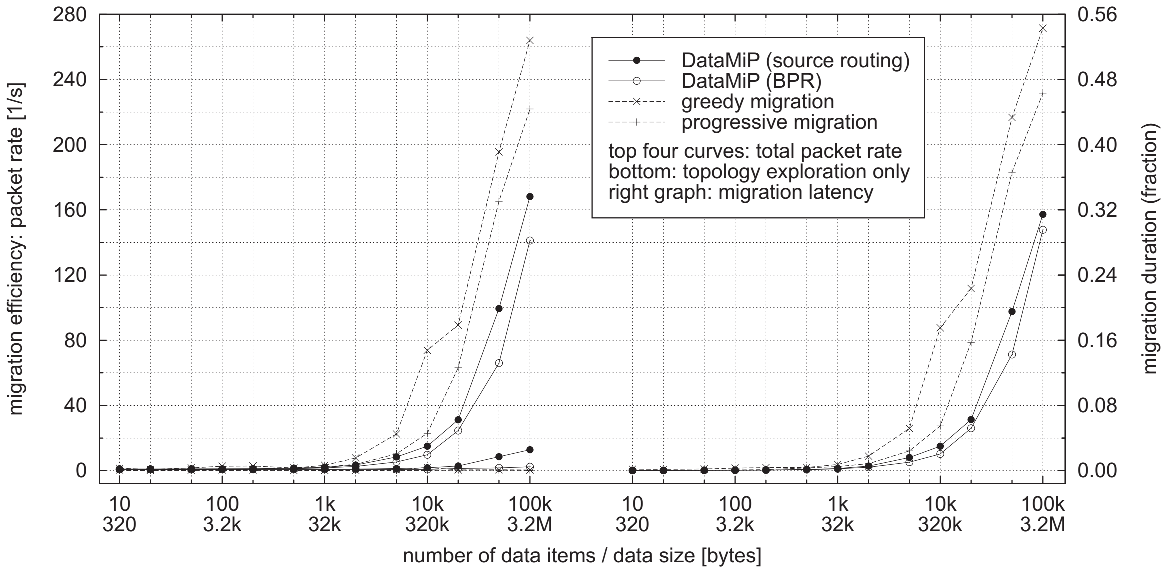


Figure 4.16: Migration efficiency and duration as a function of the number of data items.

items does not require any significant effort to succeed. According to Table 4.1, up to 47 data items can be carried in a single packet. Thus, for 1000 data items, roughly 21 data packets are required, which all approaches can handle without any problems.

When the number of data items is increased beyond 1000, DataMiP starts to show a clear performance advantage over greedy and progressive migration. This is due to the resilience of DataMiP's selected path based on the stability analysis in Section 4.4.2. Thereby, most of the initiated migrations can be completed successfully. This is in contrast to greedy and progressive migration, which repeat those migrations that have failed in the first attempt.

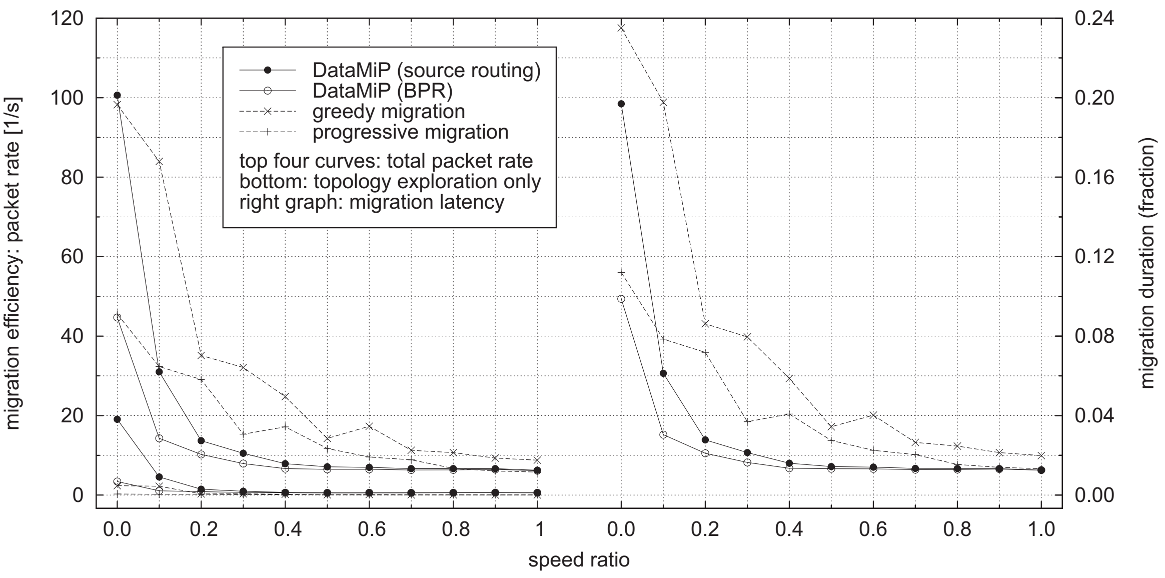


Figure 4.17: Migration efficiency and duration as a function of the speed ratio (maximum node speed: 15 m/s).

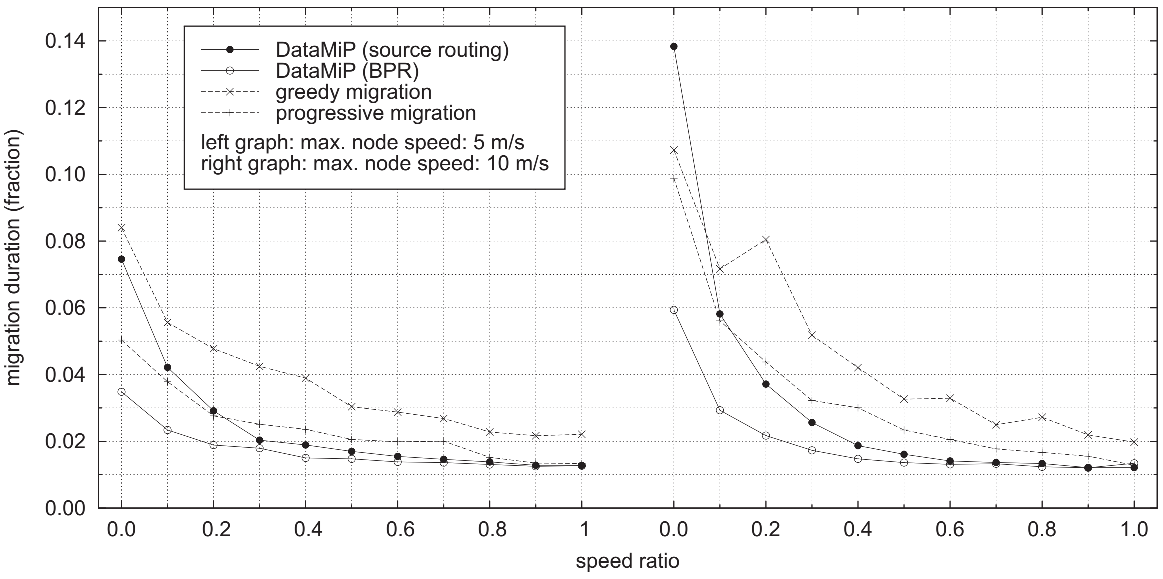


Figure 4.18: Migration duration as a function of the speed ratio (maximum node speed: 5 m/s (left) and 10 m/s (right)).

Further, we observe that DataMiP's cost for network topology exploration increases at 20000 data items and above. This is due to the fact that several topology explorations are required during a single migration in order to update the network path via which migration occurs. Also observe that in the range of large migration overhead and duration, a direct relation with Figure 3.29 and 3.40 of the core data storage analysis can be observed.

Figure 4.17 shows migration efficiency and duration as a function of the speed ratio. While DataMiP (BPR) performs best for all settings of the speed ratio, DataMiP (source routing) degrades noticeably at a speed ratio of zero. This is due to the presence of only fast-moving nodes, in which case the MDP's stability analysis yields paths with a larger number of hops in order to achieve longevity of paths (Section 4.4). However, the communication overhead of DataMiP (source routing) is justified when considering the benefit that can be achieved in migration stability. In Figure 4.21 and 4.22, for the default case of 10000 data items and a speed ratio of 0.2, virtually no recoverable nor fatal migration failures occur for DataMiP (source routing) due to the affectiveness of the stability analysis.

We further observe the steep gradient of both variants of DataMiP in Figure 4.17, which is responsible for DataMiP's performance advantage starting from a speed ratio of just 0.1. This is due to the MDP's election process that is based on the distance-based eligibility (Section 4.4.1.1) and sojourn-based eligibility (Section 4.4.1.2) of network nodes. Once a certain level of slowly moving nodes becomes available, DataMiP immediately prefers these over fast moving ones. This leads to a significantly smaller number of migrations and in turn to an immediate increase in migration efficiency and decrease in migration latency.

Figure 4.18 shows the migration duration as a function of the speed ratio for a maximum speed of 5 m/s (left) and 10 m/s (right), complementing the right part of Figure 4.17. We observe that with an increasing maximum speed in the speed ratio, the performance of DataMiP (source routing) not only increases faster, but also achieves better performance than greedy and progressive migration at a smaller speed ratio already. This is again because of the higher exploitation potential when faster nodes become available in the network.

4.6.4 Migration Robustness

We conclude the analysis of migration performance with Figure 4.19 through 4.22 to show DataMiP's strong robustness. Figure 4.19 shows a detailed view on the migration failures in relation to the overall number of migrations. For the default of 150 nodes and above, virtually no migration failures of any kind occur in the case of both DataMiP variants. The high performance can be attributed to the benefit of the MDP's stability analysis, which has great flexibility in determining a suitable path and target node in the presence of a large number of nodes. In contrast, greedy migration in particular but also progressive migration are subject to a comparatively large number of failures. This is due to the previously discussed problem of geometric forwarding (Figure 4.9 in Section 4.6.2) in both approaches.

In the lower range of the number of nodes we can observe that the number of recoverable failures increases noticeably for both DataMiP variants. The reason can be attributed to DataMiP's migration recommendation policy. At very low node densities, the critical distance d_{crit} is reached by the current server more frequently, which implies $\Pi = 1$ and thus forces the MDP to perform a migration. Such a migration will be attempted even when no target node can be found that is reachable from the current server via a relatively stable path. This situation has been previously discussed in conjunction with Figure 4.9 and suggests that the threshold and critical distances should be appropriately calibrated.

Figure 4.20 provides a zoom of Figure 4.19 to reveal the magnitude of fatal migration failures. The important difference to recoverable migration failures is that fatal failures result in the forming of redundant data servers. Because greedy and progressive migration do not implement a consolidation mechanism, this kind of failure is especially critical for these approaches.

While the magnitude of fatal failures is significantly smaller than the one of recoverable failures

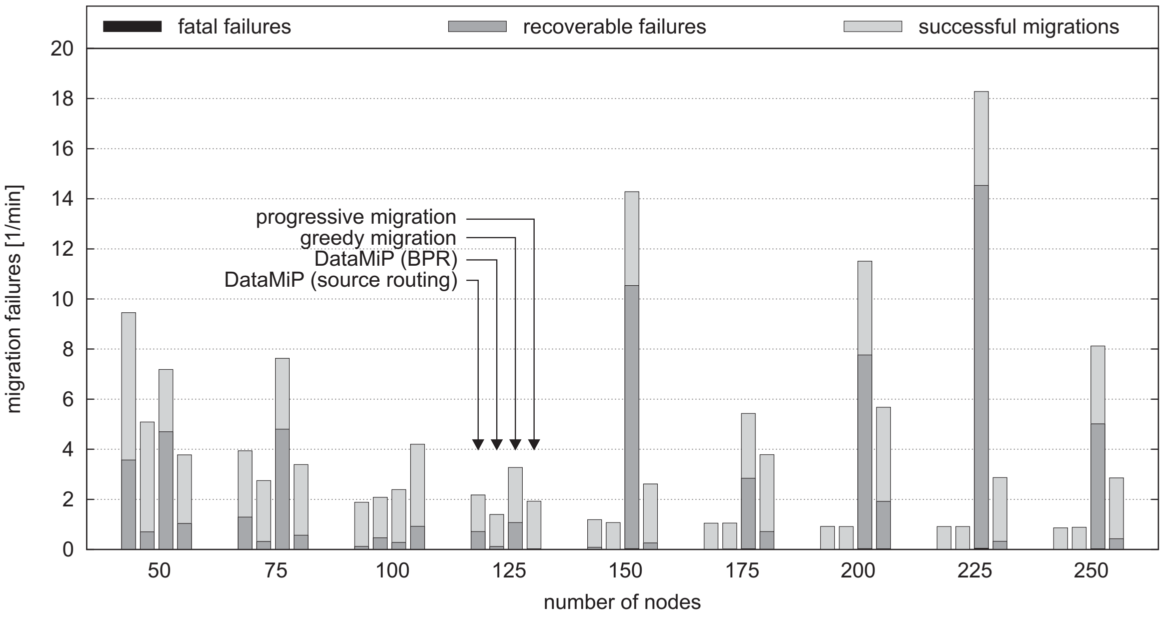


Figure 4.19: Number of migration failures as a function of the number of nodes.

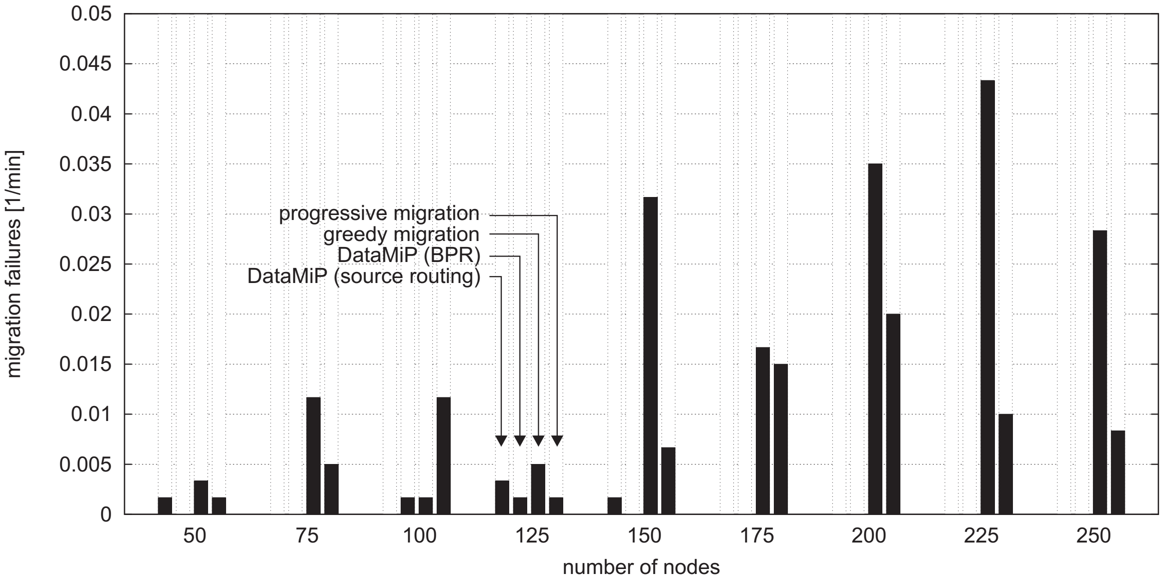


Figure 4.20: Number of fatal migration failures as a function of the number of nodes.

in our evaluation scenario, the following arguments show the significance in more realistic scenarios. Consider a total area of $2 \cdot 2$ km^2, divided into 100 cells, each measuring $200 \cdot 200$ m^2. This results in 100 reference coordinates, with 100 associated data servers in the case of no replication.[2] This scenario can be easily conceived, e.g., in a city center. Let us assume a fatal failure rate of just 0.005 per minute, which is roughly the best value for greedy and progressive migration achieved at the default setting of 150 nodes. This results, by average, in one fatal migration failure every 2 minutes, or 30 fatal failures in one hour. The key point is that these

[2]Applying replication makes our case even stronger.

failures are not resolved by greedy and progressive migration. Thus, after one hour, roughly 130 data servers will form significant undesired redundancy that even continues to grow.

Considering DataMiP, on the other hand, the number of fatal migration failures is lower than those of greedy and progressive migration in all but one of the cases, and vanishes completely for node densities of 175 and above. In contrast to greedy and progressive migration, DataMiP is able to recover also from fatal migration failures by its consolidation mechanism. Both facts together provide significant resilience for both types of failures and justify the increased communication cost in some cases that was observed in previous figures.

Figure 4.21 and 4.22 show the number of migration failures as a function of the number of data items. Let us first focus on the overall picture and the recoverable migration failures in Figure 4.21. We can observe that the number of recoverable failures for both DataMiP variants is virtually not present up to 10000 data items, and up to 20000 data items for DataMiP (source routing). In contrast, greedy and progressive migration both show a significant amount of failures even for small numbers of data items. In particular, greedy migration fails frequently even for a single packet to be migrated, which is due to the greedy forwarding problem discussed previously in Section 4.6.2. These observations definitely back up the significance of the stability analysis and path selection of the migration decision policy.

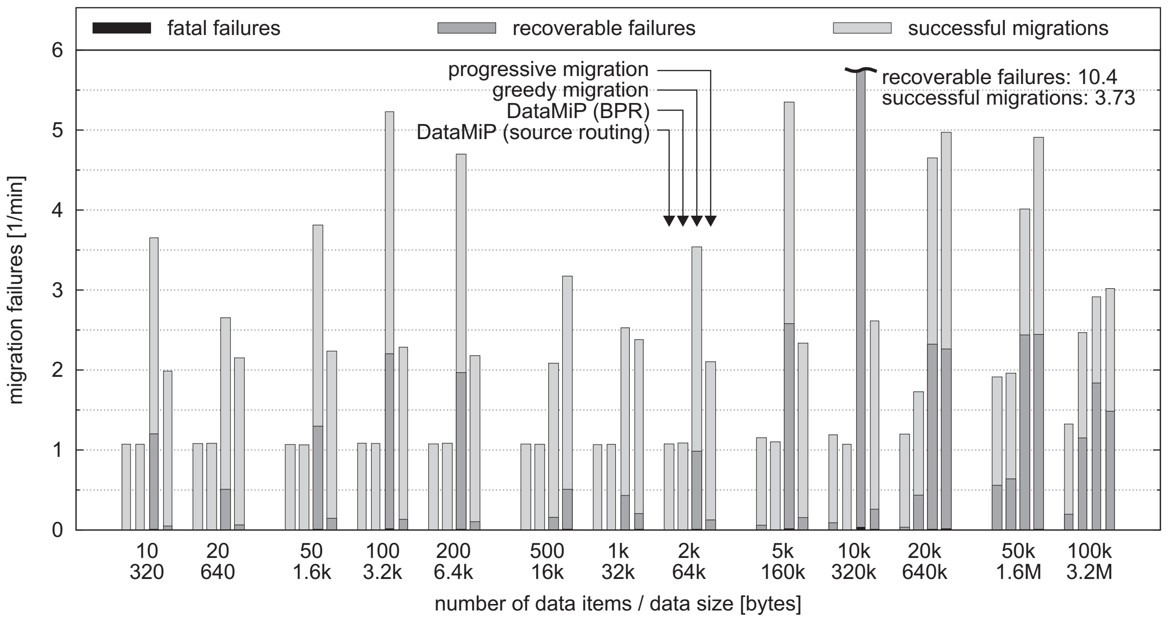


Figure 4.21: Number of recoverable migration failures as a function of the number of data items.

With an increasing number of data items, the number of recoverable migration failures increases for all approaches. However, in all cases, the number of failures is significantly smaller for both DataMiP variants and in particular for DataMiP (source routing). This can be attributed to the stability analysis one more time, which performs path recomputations that explicitly take into account the size of the data subset to be migrated.

Figure 4.22 provides a zoom into the fatal migration failure rate. The figure shows well the strength of DataMiP's stability analysis with respect to the number of fatal failures. While for greedy and progressive migration a significant number of fatal failures occur, virtually no failures are present for both DataMiP variants up to 10000 data items. Beyond 10000 data items, some failures occur for DataMiP (BPR), whereas DataMiP (source routing) does not show any failures up to the maximum of 100000 data items. This figure underpins that DataMiP (source routing) is the first choice for migration in the case of large data subsets.

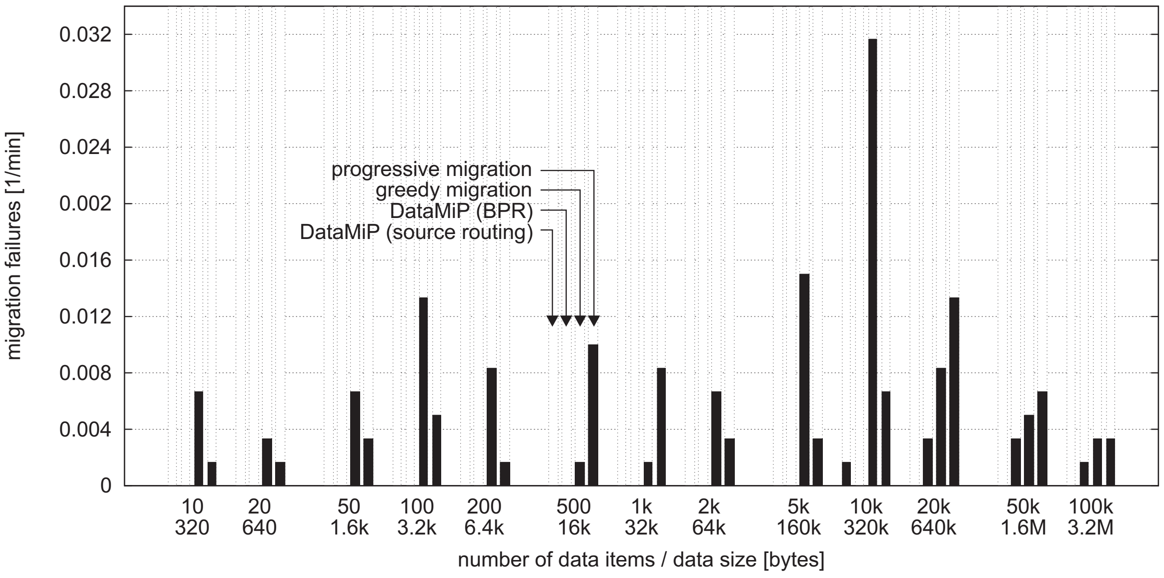


Figure 4.22: Number of fatal migration failures as a function of the number of data items.

4.6.5 Evaluation Summary: Data Migration

The performance analysis of data migration allows us to summarize the following key results. First of all, the resilience of the proposed migration mechanisms is extremely strong. This is due to the migration decision policy's strategies for selecting stable network paths that are used during migration. While some recoverable failures occur due to the fact that migration is sometimes forced by the migration recommendation policy, virtually no fatal failures occur, which would result in undesired server redundancies. In the few cases where redundancies still occur, the consolidation mechanism is able to resolve these redundancies eventually when partitions join to avoid the long-term forming of many redundant servers.

While both migration via source and geometric routing have similar performance in terms of resilience, using source routing based on the path that is output by the stability analysis is generally to be preferred. This choice holds especially for the case of large data subsets, where virtually no failures occur for source routing-based migration. However, when a smaller number of data items is to be managed, using migration via geometric routing delivers sufficient stability, and its additional performance gain over source routing then makes it the better choice.

DataMiP also shows superior performance when the set of nodes from which a target server is elected contains nodes that move at low speed. The performance gain already occurs for a speed ratio of just 1:5, where only 20% of the nodes move at a low speed. The stability analysis quickly exploits the difference in node speeds and selects better nodes, in contrast to greedy and progressive migration that do not distinguish between node speeds. This choice leads to fewer migrations, which contribute to the overall minimization of migration failures.

Chapter 5

Service Tier

The previous two chapters described the main contribution of this dissertation, consisting of the algorithms of the storage and routing tier according to the LCS framework in Figure 2.22. In close collaboration, both tiers provide robust, efficient, and scalable data storage based on the location-centric storage paradigm of Definition 2.2.

In this chapter, we show how the storage tier can be exploited by concrete services of the LCS framework's service tier, in order to deliver suitable base functionality to a variety of location-based applications. We begin by addressing Requirement (6) from Section 2.4 and introduce suitable location semantics in Section 5.1 that incorporate the natural imperfection of localization technology. In Section 5.2 we extend the system model from Section 3.1 assumed so far to also capture these semantics and to provide a storage structure on top of core data storage that is tailored to the extended location semantics.

Section 5.3 introduces a location updating algorithm that builds on the location semantics and the extended system model. The service tier of the LCS framework is complemented in Section 5.4 with the presentation of the semantics and algorithms for the processing of probabilistic spatial queries, including range and k-nearest neighbor queries. Evaluation results are discussed in Section 5.5 to show the performance of the proposed update and query algorithms.

5.1 Semantics of Inaccurate Locations

Let us now formalize the notion of *location* in the presence of sensor measurements that lead to inaccurate location information. We apply concepts of two-dimensional (bivariate) probability distributions, which are able to model inaccurate locations in a general way.

Definition 5.1. The *location probability density function* (location pdf) $\varrho(X)$ with $X = (x, y) \in \mathbb{R}^2$ is any two-dimensional pdf in the Euclidean plane $\mathbb{R}^2$ that satisfies

$$\int_{X \in \mathbb{R}^2} \varrho(X) dX = 1 \tag{5.1}$$

In many cases, the technical means by which position information is gathered lead to an indefinite area where the location pdf is greater than zero at any point X of that area. For example, localization technology based on the Global Positioning System (GPS) may, theoretically, possess an arbitrarily large error, but with a negligible probability, which is sufficient for most applications. Therefore, we provide a more practical notion of location that restricts the location pdf to a finite supporting area $L \subset \mathbb{R}^2$. Let $p \in [0,1]$ denote the probability by which a single position is contained inside of L. We can write

$$\int\limits_{X \in L} \varrho(X)dX = p \tag{5.2}$$

In general, $p < 1$ holds. Value p can be set close to 1 such that the supporting area L contains the corresponding physical object with a sufficiently high probability that is suitable for some specific application, such as the processing of spatial queries considered in Section 5.4.

The following example illustrates the application of the general concept. Let us consider the characteristics of a typical GPS sensor, using the state-of-the-art Garmin GPSMAP 76CSx hand-held device [Gar09], which uses the highly accurate SiRFstarIII chipset.

Inaccuracies in the position information of devices like the given one are usually specified in meters, indicating the so-called circular error probable (CEP). The CEP is a measure for a circle of radius r and specifies the probability by which a single position reading can be found within that circular area. If not stated otherwise, the CEP indicates the probability of 0.5, and the notation CEP_{95}, for instance, indicates that an object's position can be found inside of the considered location with a probability of 0.95.

In the case of a GPS sensor, it is valid to approximate the probability distribution of the position reading by a bivariate normal distribution, which in its general form is

$$f(X) = \frac{1}{2\pi\sqrt{|\Sigma|}} e^{-\frac{1}{2}(X-\mu)^T \Sigma^{-1}(X-\mu)} \tag{5.3}$$

In (5.3), Σ denotes the covariance matrix, and $|\Sigma|$ its determinant. The probability P_r to find a point inside of a circle with center μ and radius r is defined as:

$$P_r = \iint_{x^2+y^2 \leq r^2} f(X)dxdy \tag{5.4}$$

This equation can be integrated (cf. Appendix C.2 for details) and yields the closed form

$$P_r = 1 - e^{-\frac{r^2}{2\sigma^2}} \tag{5.5}$$

The CEP (50%) can be obtained by setting $P_r = 0.5$ in (5.5) and solving by r:

$$r_{50} = \sigma\sqrt{2\ln 2} \tag{5.6}$$

Any other CEP can be obtained by using the specific value for P_r.

The derivation of this equation is important in order to easily derive CEP values other than the 50% CEP that a GPS receiver yields. Coming back to the example, we require (5.5) to compute, e.g., CEP_{95} or any other CEP for larger probabilities. However, if a CEP is selected

Circular Error Probable	**Radius**	**Circular Error Probable**	**Radius**
1CEP = CEP_{50}	r_{50}	CEP_{95}	$2.08 \cdot r_{50}$
CEP_{60}	$1.15 \cdot r_{50}$	CEP_{96}	$2.15 \cdot r_{50}$
CEP_{70}	$1.32 \cdot r_{50}$	CEP_{97}	$2.25 \cdot r_{50}$
CEP_{80}	$1.52 \cdot r_{50}$	CEP_{98}	$2.38 \cdot r_{50}$
CEP_{90}	$1.82 \cdot r_{50}$	CEP_{99}	$2.58 \cdot r_{50}$
2CEP = $\text{CEP}_{93.8}$	$2 \cdot r_{50}$	3CEP = $\text{CEP}_{99.8}$	$3 \cdot r_{50}$

Table 5.1: Radius of the supporting circular location area in relation to CEP values. The CEP value can be obtained by using $1 - e^{-k^2 \ln 2}$ where k denotes a multiple of radius r_{50} for CEP_{50}. The calculation of r from a given CEP probability can be accomplished by using $k = -\ln(1-p)/\ln 2$, where k is the factor preceding r_{50} and p the CEP probability.

to high, then the radius r will increase to a value that will make location information inefficient to be managed by the update algorithm in Section 5.3. Table 5.1 lists radius values for selected CEP values. Assume, for example, that the CEP reported by the GPS device is 3 m. If CEP_{95} is required, then the radius to be used is 6.24 m.

We now define the location semantics that we use in subsequent sections based on the CEP notion, which is practical to characterize the required accuracy for a location, and which can be easily translated to different CEP values based on the reading of a GPS device.

Definition 5.2. The *location* **L** of an object is defined as the location pdf that models the localization system and a circular supporting area for which a particular CEP is given:

$$\mathbf{L} := (\varrho(X), \text{CEP}_Y) \tag{5.7}$$

In the previous definition, Y denotes the required probability that the object is located inside of the supporting location area defined by CEP_Y.

Let us use the GPS hand-held device for illustrating the notion of location. We assume bivariate normal distribution, in general notation $\mathcal{N}(\mu, \Sigma)$, where Σ denotes the two-dimensional covariance matrix. Assuming symmetric covariances in x and y, we can use (5.6) to compute σ^2 for a given $r_{50} = 3$ m, yielding $\sigma^2 \approx 6.49$. For an example probability value of $Y = 95\%$, the location of an object can be specified as follows:

$$\mathbf{L} := \left(\mathcal{N}\left(\begin{pmatrix} 2 \\ 4 \end{pmatrix}, 6.49\right), \text{CEP}_{95}\right) \tag{5.8}$$

In this example, the supporting location area is a circle with radius 6 m around position $(2,4)^T$. With a probability of 95%, a position that is acquired by the localization system is actually located inside of this circle in the physical world. In subsequent sections, L_i denotes the supporting location area of an object o_i's location $\mathbf{L}_i$.

5.2 System Model Extensions

We now extend the system model of Section 3.1 to accommodate a geometric overlay that supports the mapping of data objects o_i to reference coordinates $\mathbf{c}_r$ and in turn to data servers in the network. Let $A \subset \mathbb{R}^2$ denote a bounded service area that shall contain the network under consideration. We assume a subdivision of A into cells $C_j \subseteq A$, an approach used previously by many storage approaches in wireless multihop networks, e.g. [LJC+00, SH04, KFWM04]. The cell structure shall be known to all nodes inside of A. Cells C_j may be of arbitrary shapes, which specifically allows the adaptation to individual underlying topographies, such as in urban scenarios with buildings that form obstacles. We further demand that A is completely covered by the union of cells C_j. While it is allowed for cells to overlap, a structuring into non-overlapping cells is most appropriate for our purposes in this chapter.

Data items $o_i \in D$ are mapped to a set of reference coordinates $\mathbf{c}_r$ in two steps. First, each object o_i is mapped to a set of cells C_j based on location $\mathbf{L}_i$ of that object. The mapping takes an object's location as argument and is denoted as $\text{rel}_{\mathbf{C}}(\mathbf{L})$. The mapping is realized by considering the overlap between the supporting location area L_i of a location $\mathbf{L}_i$ and each cell $C_j \subseteq A$. Using $\mathbf{C} := \{C_j : C_j \subseteq A\}$, the set of cells C_j to which o_i is mapped is

$$\mathbf{C}_i := f_{\mathbf{C}}(\mathbf{L}_i) = \{C_j \in \mathbf{C} \ : \ L_i \cap C_j \neq \emptyset\} \tag{5.9}$$

In a second step, this mapping is concatenated with a mapping from cells to reference coordinates, the mapping being denoted as $f_{\mathbf{c}}$. This mapping may be arbitrarily defined, which allows great flexibility in associating a specific cell with any number of reference coordinates. In the following, we assume a one-to-one mapping in which each cell is mapped to a single reference coordinate. More complex mappings are possible in conjunction with replication, which is briefly discussed in the outlook in Section 6.2.1.

5.3 Location Updating

In the following, we derive the location updating algorithm that efficiently delivers location information to the servers of the underlying LCS infrastructure based on the location semantics defined in Section 5.1. While the focus is on location information only, the proposed concepts equally hold for many other types of dynamic information that is gathered by individual network nodes, for instance, temperature readings that are obtained from a node's local temperature sensor. Consequently, the following results have validity not only for location-based services, but also for more general context-based services and applications.

For a mobile object, we assume a location fix is performed by a client node after constant but arbitrary time intervals. In the case of the example GPS device, this interval is 1 s. Each location fix captures the current position of an object and constructs a location in the form of (5.7) from it. Using the results from Section 5.1, the value CEP_Y can be defined for any Y.

Because no restrictions are put on the duration of the location fixing time interval, it is necessary to remove old copies that are no longer valid after the most current location update has occurred. Upon the delivery of the most recent location update to a data server, some copies of stale location information might exist on a number of other servers that need to be removed. While a soft state approach is sufficient to eventually remove stale object copies, it may leave a potentially large number of outdated copies in the system even in the presence of more recently executed updates. The reason is that object information is generally shared by multiple servers, and the set of servers receiving an object update may change over time.

Therefore, the following location updating algorithm attempts to proactively remove copies that are known to be superseded. This leads to a significant decrease in the number of outdated copies and in turn increases overall data consistency. This is especially important for the complementary services, such as query processing, which are able to achieve accurate results only if the location updating mechanism makes sure that the data store is consistent in the first place. Still, proactive measures can only be efficient when best effort, which is due to the system model (Section 3.1) that assumes that messages may be lost and network partitioning may occur. We therefore assume, in addition to the proactive removal of object copies, an object timeout that is rather large, and that updates occur at least with a frequency such that an object is refreshed before that timeout occurs.

Let us now turn to the location update algorithm, shown in Listing 5.1. The algorithm can be roughly divided into the SendObjectUpdate procedure (lines 15-23) and the remaining procedures in lines 25-79. While the former is executed by any node in the network that performs object location fixes, the latter are executed by data servers only.

Upon the construction of location $\mathbf{L}_i$ for a data item o_i as a result of a location fix, SendObjectUpdate is invoked (line 15). This procedure essentially creates a record of type Object (lines 17-21) from the object identifier, observation time, and object location. The special data field $o_i.\mathbf{L}_{\text{prev}}$ (line 21) is used in conjunction with object tracing, which is elaborated later on. After the creation of the object record, the information is sent towards all reference coordinates that are associated with a cell of the set $f_{\mathbf{C}}(\mathbf{L}_i)$ (line 22). The mapping from cells to reference coordinates is accomplished by function $f_{\mathbf{c}}$ in the second argument of the SendObjectUpdate procedure. Note the definition of both $f_{\mathbf{C}}$ and $f_{\mathbf{c}}$ in Section 5.2.

Listing 5.1: Location updating.

```
1  module LocationUpdating
2  begin
3    type
4      Object = record
5        ID: integer
6        $t_{\text{obs}}$: Time
7        $\mathbf{L}$: Location
8        $\mathbf{L}_{\text{prev}}$: Location // for object tracing
9      end record
10
11   var storage: Set of Object
12   var $\mathbf{C}$: Set of integer // global set of cells
13   var $C_j$: integer // this node's associated cell
14
15   procedure SendObjectUpdate($\mathbf{L}_i$: Location)
16   begin
17     var $o_i$: Object
18     $o_i$.ID := $i$
19     $o_i.t_{\text{obs}}$ := CurrentTime()
20     $o_i.\mathbf{L}$ := $\mathbf{L}_i$
21     $o_i.\mathbf{L}_{\text{prev}}$ := null
22     for each $C_j$ in $f_{\mathbf{C}}(\mathbf{L}_i)$ do SendObjectUpdate($o_i$, $f_{\mathbf{c}}(C_j)$)
23   end
24
```

```
  procedure ReceiveObjectUpdate($o_i$: Object)
  begin
    var $\mathbf{C}_{\text{curr}}$, $\mathbf{C}_{\text{prev}}$, $\mathbf{C}'$: Set of integer
    var $C_{\text{next}}$: integer
    if $\exists o' \in$ storage: $o'.\text{ID} = o_i.\text{ID}$ then
      if $o_i.t_{\text{obs}} < o'.t_{\text{obs}}$ then return
      $o_i.\mathbf{L}_{\text{prev}}$ := $o'.\mathbf{L}$
      $\mathbf{C}_{\text{prev}}$ := $f_{\mathbf{C}}(o'.\mathbf{L})$
    else // no previous object copy exists
      $\mathbf{C}_{\text{prev}}$ := $\emptyset$
    end if
    $\mathbf{C}_{\text{curr}}$ := $f_{\mathbf{C}}(o_i.\mathbf{L})$
    ExecuteUpdate(storage, $o_i$)
    if $C_j = \min(\mathbf{C}_{\text{curr}})$ then
      if $\mathbf{C}_{\text{prev}} \neq \emptyset$ then
        $\mathbf{C}'$ := $\mathbf{C}_{\text{prev}} \setminus \mathbf{C}_{\text{curr}}$
        for each $C' \in \mathbf{C}'$ do SendObjectDelete($o_i$, $f_{\mathbf{c}}(C')$)
      else // no previous object copy exists
        if $\nexists C' \in \mathbf{C} \setminus \mathbf{C}_{\text{curr}} : d(C', o_i.L) < d_{\text{trace}}$ then return
        $\mathbf{C}'$ := $\mathbf{C} \setminus \{C_j\}$
        $C_{\text{next}}$ := $C' \in \mathbf{C}' : \forall C'' \in \mathbf{C}' : d(C', o_i.L) < d(C'', o_i.L)$
        SendObjectTrace($o_i$, $f_{\mathbf{c}}(C_{\text{next}})$)
      end if
    end if
  end

  procedure ReceiveObjectDelete($o_i$: Object)
  begin
    if $\exists o' \in$ storage: $o'.\text{ID} = o_i.\text{ID}$ then
      if $o_i.t_{\text{obs}} > o'.t_{\text{obs}}$ then ExecuteDelete(storage, $o_i$)
    end if
  end

  procedure ReceiveObjectTrace($o_i$: Object)
  begin
    var $\mathbf{C}_{\text{prev}}$, $\mathbf{C}'$: Set of integer
    var $C_{\text{next}}$: integer
    $\mathbf{C}_{\text{prev}}$ := $\emptyset$
    if $\exists o' \in$ storage: $o'.\text{ID} = o_i.\text{ID}$ then
      if $o_i.t_{\text{obs}} < o'.t_{\text{obs}}$ then return
      if $o_i.t_{\text{obs}} = o'.t_{\text{obs}}$ then // identical object copy
        $\mathbf{C}_{\text{prev}}$ := ($\mathbf{L}_{\text{prev}} =$ null) ? $\emptyset$ : $f_{\mathbf{C}}(o'.\mathbf{L}_{\text{prev}})$
      else // $o_i.t_{\text{obs}} > o'.t_{\text{obs}}$
        $\mathbf{C}_{\text{prev}}$ := $f_{\mathbf{C}}(o'.\mathbf{L})$
      end if
    end if
    if $\mathbf{C}_{\text{prev}} \neq \emptyset$ then
      $\mathbf{C}' = \mathbf{C}_{\text{prev}} \setminus f_{\mathbf{C}}(o_i.\mathbf{L})$
      for each $C' \in \mathbf{C}'$ do SendObjectDelete($o_i$, $f_{\mathbf{c}}(C')$)
    else // object tracing continues
      $\mathbf{C}'$ := $\{C' \in \mathbf{C} : d(C', o_i.L) > d(C_j, o_i.L)\}$
      $C_{\text{next}}$ := $C' \in \mathbf{C}' : \forall C'' \in \mathbf{C}' : d(C', o_i.L) < d(C'', o_i.L)$
      if $d(C_{\text{next}}, o_i.L) \leq d_{\text{trace}}$ then SendObjectTrace($o_i$, $f_{\mathbf{c}}(C_{\text{next}})$)
    end if
  end
end
```

After a data server has received an object update (line 25), it checks whether a copy of that object, denoted o', with the same ID already exists in the server's local storage (line 29). If yes, it is further checked whether the observation time of o' is more current than that of the received object o_i (line 30). If this is the case, no further actions are taken and the procedure returns. Otherwise, o' is a less current object copy whose location is buffered in the current object for future object tracing purposes (line 31). The set of cells with which the object's location overlaps is calculated by $f_{\mathbf{C}}(o'.\mathbf{L})$ and stored in the previous list variable $\mathbf{C}_{\text{prev}}$ (line 32). If no copy of the received object exists, the previous list is initialized with the empty set (line 34). In line 36 the current set of cells, $\mathbf{C}_{\text{curr}}$, is assigned the set of cells with which the location of the updated object o_i overlaps. Both lists are used for the subsequent proactive object removal process. The object update is finally executed on the local storage (line 37).

The proactive object removal process is handled in lines 38-48. Because an object's location may overlap with multiple cells, multiple servers may receive an update for the same object. It is therefore necessary to assure that only a single server takes responsibility in handling the removal of previous object copies. This selection is accomplished in line 38, where the server having the smallest associated cell ID is deemed responsible. If a previous object copy exists, which is indicated by a nonempty previous cell set (line 39), one object delete message is sent to each data server that is responsible for a cell contained in the set of cells denoted $\mathbf{C}'$ (line 40). This set contains all cells of the previous cell set, but not the cells of the current cell set, because the latter contains cells that are associated with servers that receive the current object copy. Object delete messages relating to each cell in $\mathbf{C}'$ are eventually received in line 51. Each delete is performed on the receiving server's local storage only if the object exists and that object's observation time is smaller than that of the received object.

The case where no previous object copy exists, indicated by an empty previous cell set (line 42), can be due to one of two reasons. Firstly, the location update on the object may be the first one that has ever occurred due to the object being observed for the first time. Second, the location update may have occurred on an object that has entered a new cell, in which case a previous object copy may still exist at a server that is responsible for another cell. Because the two cases cannot be distinguished, a tracing process is considered in the following lines of code. The condition in line 43 checks whether a cell exists that satisfies a specific tracing distance d_{trace}, but which is not one of the cells that overlaps the object's location. By setting d_{trace} to a value larger than the maximum distance an object may travel between two consecutive location updates, it is guaranteed that all cells with which the location of a potential previously updated object overlaps are considered. If no such object exists, the procedure returns. Otherwise, a set of cells, denoted $\mathbf{C}'$, is constructed, which contains all cells in the considered network except the current cell (line 44). From this set, the cell that is closest to the supporting location area of the current object o_i is selected (line 45). This cell is the one with the highest probability to find a previous object copy after an object has entered a new cell. The procedure terminates with the sending of an object trace message towards the reference coordinate that is associated with this cell (line 46). Note that based on the check in line 43 this cell is guaranteed to have a distance to o_i's location area that is smaller than d_{trace}.

After the reception of an object trace message at a data server (line 58), a previous cell set is first initialized with the empty set (line 62). Next, it is checked whether the object to be traced exists in the local storage of the server (line 63). If a matching object copy exists but its observation time is more current, the procedure returns (line 64). Otherwise, either an identical copy of the object exists (line 65) or a less current one (line 67). In the former case, the previous cell set is initialized with the set of cells with which the previous location of object o' overlaps (line 66). This set is empty if no previous location is available. In the latter case, the set is initialized with the set of cells with which the existing object's location overlaps (line 68). The subsequent program code (lines 71-78) determines whether the object can be deleted or is to be further traced. A non-empty previous cell set (line 71) indicates that the trace was successful, that is, a previous object was found and its deletion can be initiated. Just like in line 41 of procedure ReceiveObjectUpdate, one delete message is sent towards each reference coordinate (line 73) that is associated with each cell of the set of cells denoted $\mathbf{C}'$ (line 72).

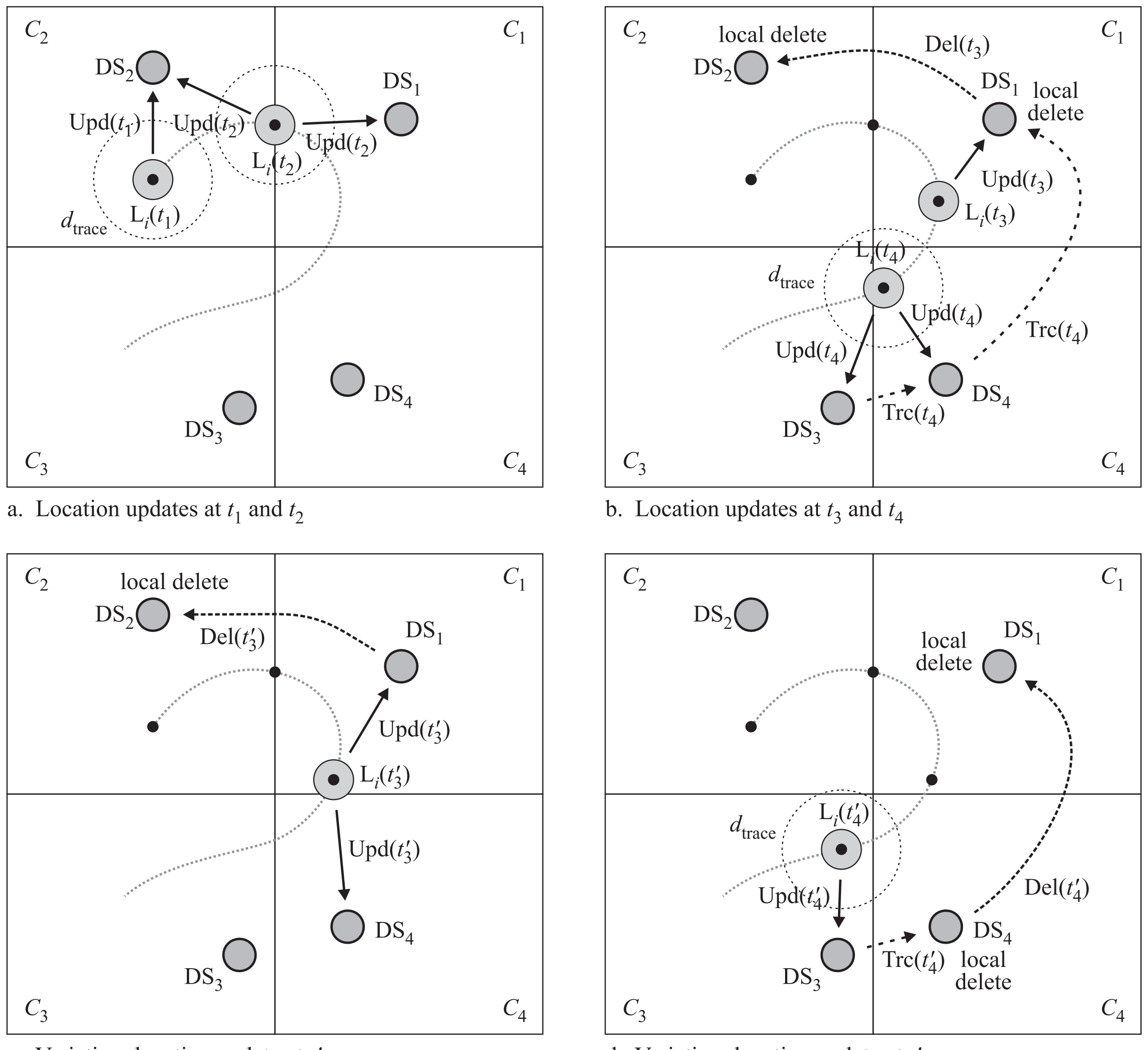
In the case where the previous cell set is empty, a previous object copy could not be found and the tracing process is continued (line 74). For that, the set of candidate cells is assigned only those cells that are located further from the current object's location than the current cell (line 75). This implies a total order of cells in a sequence of increasing distance that is followed for subsequent tracing attempts, consecutively selecting more distant cells. After choosing the next cell (line 76), the tracing process continues by sending an object trace message to the reference coordinate that is associated with this cell (line 77). Note that the tracing process eventually terminates when d_{trace} is no longer satisfied.

The following example, depicted in Figure 5.1, illustrates the operation of the location update algorithm. Note that the depiction is not to scale and cell borders are generally crossed much rarer than shown. Let $t_1, \ldots, t_4$ denote the time of a location fix, with $t_{i+1} > t_i$. We write $L_i(t_i)$ for the supporting location area of an object o_i at time t_i, and d_{trace} for the tracing distance. Let DS_i and C_i denote data servers and cells, respectively. Let further $\text{Upd}(t)$, $\text{Del}(t)$, and $\text{Trc}(t)$ denote an object update, delete, and trace, respectively, sent to one or more data servers.

Figure 5.1.a and 5.1.b show a sequence of location fixes along the physical path of the object o_i, indicated by the dotted line. In Figure 5.1.a, at t_1, object o_i is observed by a mobile node for the first time and an update is sent to DS_2. Upon the reception of the update at DS_2, the update is inserted into the server's local storage. Because no previous object copy exists at DS_2, an object trace process is considered (line 42 et sqq.). However, because the tracing distance spans a circle that is completely located inside of cell C_2, no tracing is performed. Recall that from the setting of this distance, it is guaranteed that no other cell exists that may have a previous copy of the object if the object had been observed previously.

At t_2, the second location fix occurs, which results in a supporting location area that overlaps with both C_2 and C_1. As a consequence, two updates are sent to DS_2 and DS_1 (line 22). By definition, the server that is associated with the lowest-ID cell, which is DS_1 in the example, handles the object removal process. Because DS_1 has not received an object update before, it considers to perform an object trace to find out about possibly existing previous object copies (line 42 et sqq.). According to the tracing distance, cell C_2 needs to be considered, but no other cell. However, the location associated with the update at t_2 also overlaps with C_2. Hence, according to line 43, there is no other cell within the tracing distance that can possibly store a previous object copy, and thus, no object tracing process occurs.

Continuing in Figure 5.1.b, the location fix at t_3 results in a single update that is sent to DS_1. This time, DS_1 stores a previous object copy (line 29). Because cell C_2 is not covered by the most recent update's supporting location area, the server associated with cell C_2 is the one from



a. Location updates at t_1 and t_2

b. Location updates at t_3 and t_4

c. Variation: location update at t'_3

d. Variation: location update at t'_4

Figure 5.1: Example: location update algorithm.

which a previous object copy must be removed (line 40). As a result, a single delete message is sent to DS_2, which deletes the object copy from its local storage.

The last update occurs at t_4. This time, the supporting location area overlaps with both C_4 and C_3, and two updates are sent to each of the servers DS_4 and DS_3. Because DS_3 is associated with the lowest-ID cell, it is responsible for handling the removal of old object copies. Because DS_3 has received the first update for the object, it considers to perform an object trace. DS_3 first determines if a cell exists that may hold a previous object copy that has not been overwritten in the meantime by the current object copy (line 43). In the example, C_3 and C_2 satisfy this condition. Excluding cell C_3, a trace message is thus sent to the nearest cell, C_4. Upon the reception of the trace message at DS_4, that server finds that it has also executed the current update on the object (line 63). Since the object copies are identical, the observation times match (line 65). Because the object has traversed the border from C_1 to C_4 completely, no previous location $\mathbf{L}_{\text{prev}}$ is known to DS$_4$. Thus, the previous cell set is empty (line 66). Consequently, object tracing continues (line 74) with the sending of a trace message to the next closer cell, C_1. At DS$_1$, the previous object copy is finally found. Because its supporting location area is fully contained inside of C_1, only a local delete is executed at DS$_1$.

Figure 5.1.c and 5.1.d show a variation of the third and fourth update, occurring at time $t'_3 > t_3$ and $t'_4 > t_4$. In Figure 5.1.c, the third update leads to location $L_i(t'_3)$, overlapping with both C_1 and C_4. Hence, updates are sent to DS$_1$ and DS$_2$. Because DS$_1$ knows about the previous copy of the object, it can directly issue a delete message to DS$_2$.

Shown in Figure 5.1.d, at t'_4, an update occurs with the object's supporting location area being completely located inside of cell C_3. Thus, DS$_3$ does not know of any previous object copy and considers to perform an object trace. According to line 43, C_4 is one candidate cell that is below the tracing distance and not among the cells with which $L_i(t'_4)$ overlaps. Thus, the tracing is actually performed. The nearest cell to $L_i(t'_4)$, excluding C_3, is C_4, thus, the trace message is sent to DS$_4$. At DS$_4$, the object copy still contains the previous copy's location $\mathbf{L}_{\text{prev}}$, which can be consulted by DS$_4$ to determine the servers from which the previous object copy is to be deleted. This is both C_4 and C_1, thus, a local delete occurs at DS$_4$ and a delete message is sent to DS$_1$, which then also performs a delete on its local storage.

5.4 Query Processing

This section describes the algorithms for processing spatial queries in mobile ad-hoc networks based on the LCS framework's core storage mechanisms. Two types of spatial queries, relevant for many location-based applications, are considered: On the one hand, a *range query* returns all objects that are located inside of a specific geometric region. On the other hand, a *k-nearest neighbor query* returns the k objects that are located closest to a given geometric reference position. Because the proposed query semantics and algorithms build on the location notion introduced in Section 5.1, the queries are referred to as *probabilistic* spatial queries.

Section 5.4.1 introduces the semantics of probabilistic range and k-nearest neighbor queries. The algorithms for processing both types of queries are described in Section 5.4.2. The discussion of a representative set of evaluation results of query algorithms in conjunction with the location update algorithm from Section 5.3 is presented in Section 5.5.

5.4.1 Semantics of Probabilistic Spatial Queries

5.4.1.1 Probabilistic Range Queries

We now define the semantics of probabilistic range queries (PRQ) using the notion of location. These semantics consist of two parts: the *inclusion condition* and the *accuracy threshold.* The inclusion condition decides whether or not an object is considered to be located inside of a geometric region R based on the object's supporting location area. By extending Equation (1) in [DGM$^+$04] to two dimensions in Cartesian space, this condition is defined as follows:

Definition 5.3. The *inclusion condition* for an object o_i is satisfied if and only if the probability that o_i is located inside of a geometric region R is greater than or equal to the *inclusion threshold* $0 < P_{\mathrm{PRQ}} \leq 1$. Having X denote any position inside of the supporting location area L_i, and ϱ_i the pdf of location $\mathbf{L}_i$, the inclusion condition is formally given by

$$P(X \in R) = \int_{R \cap L_i} \varrho_i(X) dX \geq P_{\mathrm{PRQ}} \tag{5.10}$$

An important fact not considered in previous work (Table 2.4 in Section 2.5) is that the position of some objects might be too inaccurate to be of use for some types of application. For example, in an application where the objects returned are to be located in the physical world, such as in the streets of a city center, a location too inaccurate might render it impossible for the user to find the object. To consider this fact, we introduce an additional query parameter that allows us to omit data objects from query processing if the degree of inaccuracy in the position information of these objects is too large.

Definition 5.4. The *accuracy threshold* defines the maximum degree of inaccuracy allowed for an object to take part in the evaluation of a probabilistic spatial query in general, and a probabilistic range query in particular. It is formally defined as a pair of values, (χ, υ)[1], where $\chi \geq 0$ and $\upsilon \geq 0$ denote the maximum circular error probable (CEP) index and maximum radius, respectively, that the supporting location area's CEP is allowed to have.

The combination of both parameters is essential, because the supporting location area of an object's location can have a small CEP radius with a very small CEP index, or vice versa.

Based on the aforementioned notions, we now define the probabilistic range query:

Definition 5.5. Let R denote the range query's target geometric region. Let $\mathrm{CEP}_{Y,i}$ and Y_i denote the circular error probable and probability of an object o_i's location, respectively. The *probabilistic range query result*, denoted $\mathrm{R}_{\mathrm{PRQ}}$, is defined as:

$$R_{\mathrm{PRQ}} = \{o_i \mid P(X_i \in R) \geq P_{\mathrm{PRQ}} \ \wedge \ \mathrm{CEP}_{Y,i} \leq \chi \ \wedge \ Y_i \leq \upsilon\} \tag{5.11}$$

In the following example, let $\mathbf{L} = (\varrho(X), \mathrm{CEP}_Y)$ denote an object's location, with

$$\varrho(X) = \begin{cases} Y/(100\pi \cdot \mathrm{CEP}_Y^2) & \text{if} \quad (X - M)^2 \leq \mathrm{CEP}_Y^2 \\ 0 & \text{otherwise} \end{cases} \tag{5.12}$$

specifying a uniform distribution over a circular location area with center M and radius CEP_Y. Let further R denote a rectangular region, the argument of the range query. Further assume the inclusion threshold $P_{\mathrm{PRQ}} = 0.75$, and $(\chi, \upsilon) = (3\text{ m}, 80)$.

Figure 5.2 illustrates the location of a number of objects. Object o_1 is included in the query result, because all restrictions in Definition 5.5 are satisfied. Object o_2 and o_3 do not satisfy

[1] χ and υ denote the greek letters Chi and Upsilon.

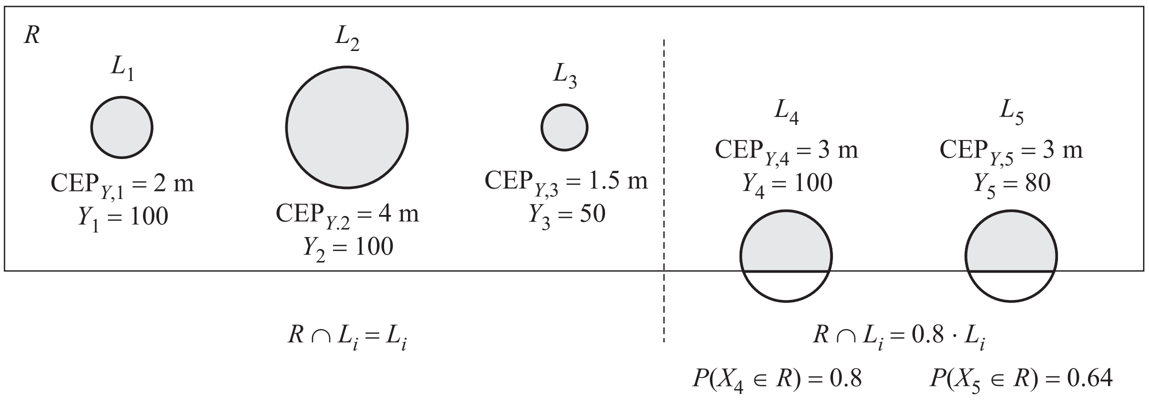


Figure 5.2: Example: probabilistic range queries.

the range query, because the CEP radius of o_2's location is too large, and o_3 does not satisfy υ. Object o_4 again satisfies the range query, because it fulfills the restrictions of both χ and υ and the probability of it being located inside of R is greater than P_{PRQ}. Object o_5 does not satisfy the range query, which is due to the smaller Y_5 in comparison to Y_4 of o_4, resulting in a probability of only $0.64 < P_{\text{PRQ}}$ of being located inside of R.

5.4.1.2 Probabilistic k-Nearest Neighbor Queries

Let us now define the semantics of probabilistic k-nearest neighbor queries (PNQ). As in the case of range queries, no accuracy threshold is considered in the related work. Furthermore, previous work does not consider a probabilistic model for the case where $k > 1$ (Table 2.4 in Section 2.5). The following semantics define exactly k objects that are considered nearest to a reference position and which are returned in the query result.

Let us first determine the probability by which an object o_j is closer to a reference position than o_k. Let $p_{\text{NN}} = (x_p, y_p)$ denote the reference position of the k-nearest neighbor query from which the k nearest objects are to be determined. To simplify notations, we introduce a polar coordinate system whose origin is p_{NN}. Figure 5.3.a illustrates the location area L_j of an object in a polar coordinate system. The location pdf is then rewritten as $\varrho_j(X)$, with $X = (r, \varphi)$, where $r^2 = (x - x_p)^2 + (y - y_p)^2$ and $\tan\varphi = (y - y_p)/(x - x_p)$.

The notion of nearer is illustrated in Figure 5.3.b for the location areas L_j, L_k of two objects o_j, o_k. Intuitively, o_j is closer to p_{NN} than o_k if o_j is closer in "most of the cases". In Figure 5.3.b, o_j is closer to p_{NN} for all pairs of positions $X_1 \in L_j, X_2 \in L_k$ for which X_1 is closer to p_{NN} than X_2. By integrating over products $\varrho_j \cdot \varrho_k$ of the location pdfs of o_j, o_k for all pairs of positions for which o_j is nearer to p_{NN} than o_k, we obtain the following definition:

Definition 5.6. The estimated probability $P_{jk}|_{p_{\text{NN}}}$, by which an object o_j is located closer to the reference position p_{NN} than o_k, is defined as:

$$P_{jk}|_{p_{\text{NN}}} = \iint\limits_{\substack{r_1=r_{\min}\\ \varphi_1=\varphi_{\min}}}^{\substack{r_1=r_{\max}\\ \varphi_1=\varphi_{\max}}} \varrho_j(X_1) \left[\iint\limits_{\substack{r_2=r_1\\ \varphi_2=\varphi_{\min}}}^{\substack{r_2=r_{\max}\\ \varphi_2=\varphi_{\max}}} \varrho_k(X_2)dR_2 \right] dR_1, \tag{5.13}$$

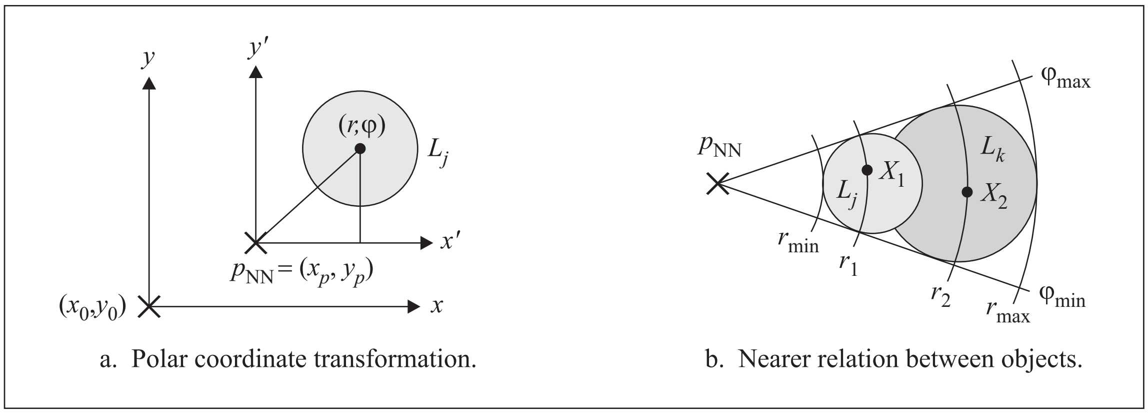


Figure 5.3: Semantics of probabilistic k-nearest neighbor queries.

where $X_1 = (r_1, \varphi_1)$ and $X_2 = (r_2, \varphi_2)$, as well as $dR_1 = r_1 dr_1 d\varphi_1$ and $dR_2 = r_2 dr_2 d\varphi_2$.

The term *estimated* indicates that the probability computations might not be complete in the following sense. Recall that the location semantics defined in (5.2) in Section 5.1 may lead to $p < 1$. This implies that the sum of the probabilities by which two objects are mutually closer to each other is *not* necessarily 1, but assumes the maximum value

$$P_{\max} = P_{jk}|_{p_{\text{NN}}} + P_{kj}|_{p_{\text{NN}}} \tag{5.14}$$

Using (5.13) and (5.14) we can now define the nearer relation between two objects.

Definition 5.7. For two objects o_j, o_k, object o_j is nearer than o_k to the reference position p_{NN} if and only if the probability that o_j is nearer than o_k is greater than one half of the maximum possible probability. Written formally:

$$o_j < o_k|_{p_{\text{NN}}} \iff P_{jk} > 0.5 \cdot P_{\max} \tag{5.15}$$

Based on Definition 5.7, it is also possible that o_j and o_k have equal distance to p_{NN}:

Definition 5.8. Two objects o_j, o_k, have equal distance to p_{NN} if the following holds:

$$o_j = o_k|_{p_{\text{NN}}} \iff P_{jk}|_{p_{\text{NN}}} = P_{kj}|_{p_{\text{NN}}} \tag{5.16}$$

The definition of the k-nearest neighbor query result R_{PNQ} is based on Definition 5.7 and 5.8. In addition, we use the accuracy threshold (χ, υ) of Definition 5.4 to only include objects in the query processing whose location satisfies that threshold.

Definition 5.9. The *probabilistic k-nearest neighbor query result*, denoted R_{PNQ}, is defined as:

$$\begin{aligned} R_{\text{PNQ}} &= \{o_1, \ldots, o_k\} \ : \ \forall o_r \in R_{\text{PNQ}} : \text{CEP}_{Y,r} \le \chi \ \wedge \ Y_r \le \upsilon \ \wedge \\ &\forall o_r \in R_{\text{PNQ}}, o_s \notin R_{\text{PNQ}} : \text{CEP}_{Y,s} \le \chi \ \wedge \ Y_s \le \upsilon \Rightarrow o_r \le o_s\,|_{p_{\text{NN}}} \end{aligned} \tag{5.17}$$

Note that if property (5.16) in Definition 5.8 holds for two objects o'_k, o''_k that are both k'th distant objects to p_{NN}, then the query result is non-unique.

5.4.2 Probabilistic Query Algorithms

This section introduces the algorithms for processing probabilistic range and k-nearest neighbor queries in mobile ad-hoc networks based on the location-centric storage framework described in Section 2.4 and shown in Figure 2.22. While the underlying core data storage algorithms address the efficiency and robustness-related Requirement (1) to (4) in Section 2.4, the query algorithms focus on Requirement (1), (5) and (6) in order to provide efficient and scalable query processing based on the query semantics defined in Section 5.4.1.

5.4.2.1 Probabilistic Range Queries

The distributed processing of probabilistic range queries (PNQs) is rather straightforward due to the following two reasons. Firstly, according to the location-centric storage paradigm (Definition 2.2 in Section 2.3), objects are stored in close proximity to where they are observed. This property is implemented by the mapping from an object's supporting location area to cells by function $f_{\mathbf{C}}$ and in turn to reference coordinates by function $f_{\mathbf{c}}$ (Section 5.2). Second, the spatial parameter of a range query, R, is a static geometric region, which can be mapped directly to the set of cells and in turn to the relevant data servers that contain data subsets that need to be considered in the evaluation of the query.

Figure 5.4 illustrates the processing of probabilistic range queries on the service tier (top plane) in collaboration with the storage tier (bottom plane) within the location-centric storage framework. The figure shows that the cell structure employed in the service tier is not visible at the storage tier, but translated by function $f_{\mathbf{c}}$. Likewise, the reference coordinates used by the core storage algorithms are not visible at the service tier.

The processing of a PRQ is initiated by the issuing of the query on the client tier on some arbitrary node in the network, in the following referred to as the query client (Figure 5.4). Because the query client may be located at any distance from the geometric region specified in the query, and thus, relevant data subsets may be located on distant data servers, it is inefficient to process the query from this node. Therefore, a query proxy that is located near the query region is selected first, which processes the range query on behalf of the query client. A simple yet effective method is to select the mobile node that is nearest to some coordinate inside of the query range, for instance, the center of a rectangular geometric region. The query is then sent to the query proxy (❶ in Figure 5.4). To do so, it is handed over to the storage tier, which uses geometric routing to relay the query from u_1 towards the reference coordinate. For that, bidirectional perimeter routing (Section 3.2) is applied to determine the node u_2 closest to the query region's reference coordinate. Upon delivery of the message to the service tier, the current node is selected by the service tier as the query proxy.

The aggregation of the query result is accomplished on the query proxy by sending a number of partial range queries (❷ in Figure 5.4) to data servers that possibly store objects that are located inside of the query region. The set of relevant cells is determined based on the overlap between cells and the query region. The query proxy subsequently determines, using mapping $f_{\mathbf{c}}$, to which reference coordinate each cell maps and hands over the query to the storage tier. The storage tier in turn forwards each partial query via a node that holds an advertisement (ADV) record (Section 3.3.1) to its corresponding data server using the request forwarding

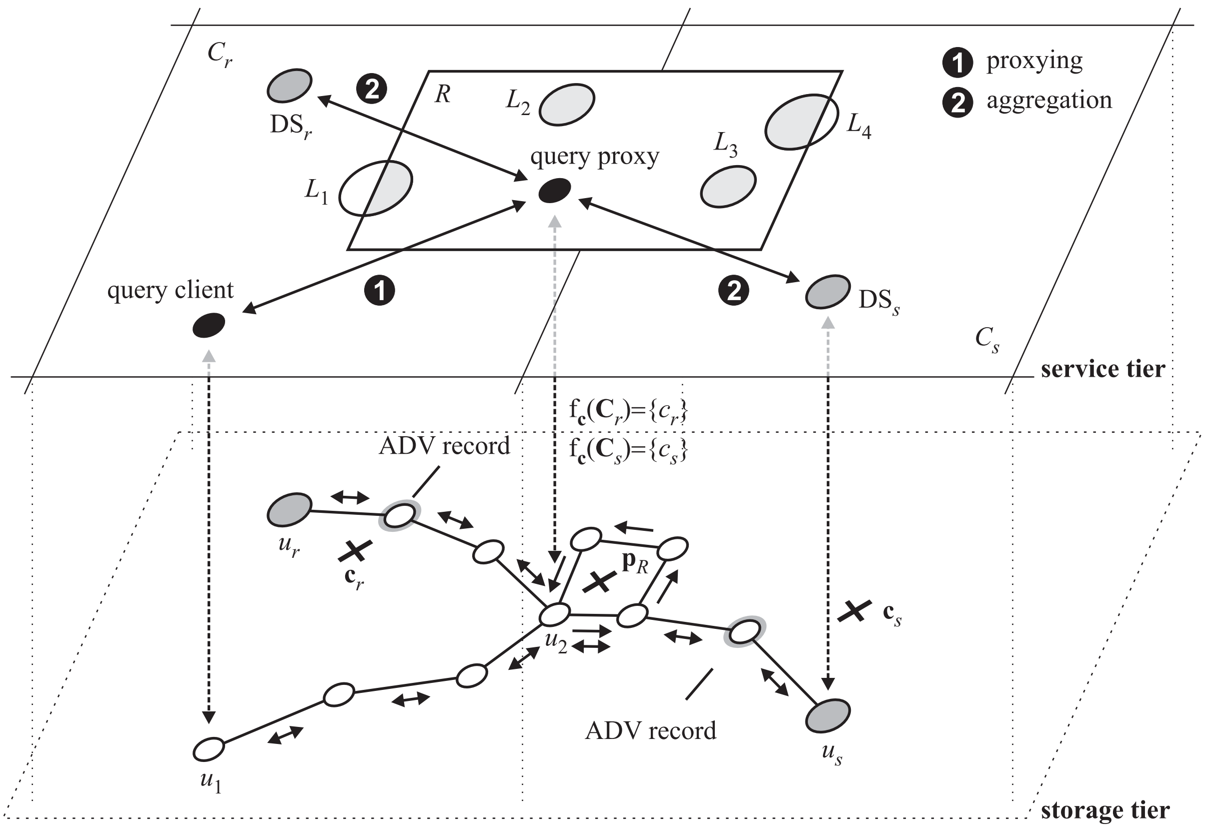


Figure 5.4: Probabilistic range query algorithm.

algorithm in Section 3.3.2. While the partial queries are under way, the query proxy stands by for receiving query results to be aggregated.

On reception of a partial query, each data server processes the range query locally based on Definition 5.5. The partial range query result is then returned to the query proxy by invoking the storage tier's forwarding mechanisms. The storage tier is able to route the result to the query proxy based on its address and last known geometric position, which we assume was previously transmitted in the partial query. For each partial query result received, the query proxy aggregates the data objects and constructs the final query result. Note that for each object it it possible to determine independently whether or not the object is included in the query result. In the event that two copies of the same data object are returned by multiple partial queries, the most recent one is included in the final result. Once all partial queries have been received, the aggregated range query result is returned to the query client by making use of the storage tier's forwarding mechanisms one more time.

5.4.2.2 Probabilistic k-Nearest Neighbor Queries

In contrast to range queries, the processing of probabilistic k-nearest neighbor queries (PNQs) is more complicated because a bounding region that contains the k nearest objects is unknown beforehand. For that reason, the PNQ algorithm uses the following two-phase approach. During the *heuristic phase*, k objects are preselected to determine the maximum range within which additional objects must be considered to compute the final query result. The selection strategy is realized by using partial k-nearest neighbor queries executed in a sequence that allows to find k candidate objects quickly and whose distance to the real k nearest neighbors is small. This step is essential to minimize additional communication cost required to complete the query during the *aggregation phase*. In this phase, all data servers that may contain objects that are closer to p_{NN} than any of the objects determined during the heuristic phase are queried by means of additional partial k-nearest neighbor queries. The final query result is then aggregated by using the nearer relation in Definition 5.7 and 5.8.

The principal operation of the algorithm for processing probabilistic k-nearest neighbor queries is shown in Listing 5.2 and Figure 5.5 on page 198. For simplicity, we assume that the query client is also the query proxy and derives the query position p_{NN} from its own location. In the case where a query client is located anywhere in the network and uses a query reference position p_{NN} that is not its own position, routing between query client and proxy is accomplished just as in the case of range query processing in Section 5.4.2.1 (❶ in Figure 5.5). In the case of a k-nearest neighbor query, the proxy is the node closest to the query position p_{NN}.

Listing 5.2: Processing of probabilistic k-nearest neighbor queries.

```
module QueryProcessing
begin
  var storage: Set of Object
  var C: Set of integer // global set of cells
  var $C_j$: integer // this node's associated cell
  var $R_{\mathrm{PNQ}}$: Set of Object // final query result, Object structure see Listing 5.1
  var C': PriorityQueue // remaining set of cells ordered by distance to $p_{\mathrm{NN}}$
  var phase: (heuristic, aggregation)
  var aggregationTimer: Timer
  var $r_{\mathrm{lim}}$: real

  procedure ProcessPNQ($k$: integer; $p_{\mathrm{NN}}$: Point; $\chi$: real; $v$: real)
  begin
    $R_{\mathrm{PNQ}}$ := $\emptyset$
    phase := heuristic
    C' := SortByDistanceTo(C, $p_{\mathrm{NN}}$)
    SendPartialQuery($k$, $p_{\mathrm{NN}}$, $\chi$, $v$, $f_{\mathbf{C}}$(GetAndRemoveNext(C')))
  end

  procedure ReceivePartialResult($R^{\mathrm{part}}_{\mathrm{PNQ}}$: Set of Object)
  begin
    // all objects in partial result satisfy the accuracy threshold
    for each $o_i$ in $R^{\mathrm{part}}_{\mathrm{PNQ}}$ do
      if not Contains($R_{\mathrm{PNQ}}$, $o_i$) then
        if $|R_{\mathrm{PNQ}}| < k$ then
          $R_{\mathrm{PNQ}}$ := $R_{\mathrm{PNQ}} \cup o_i$
        else // $|R_{\mathrm{PNQ}}| = k$
          if $\exists o_j \in R_{\mathrm{PNQ}} : o_i < o_j|_{p_{\mathrm{NN}}} \wedge \forall o_k \in R^{\mathrm{part}}_{\mathrm{PNQ}} : o_i \leq o_k|_{p_{\mathrm{NN}}}$ then
            Replace($R_{\mathrm{PNQ}}$, $o_j$, Remove($R^{\mathrm{part}}_{\mathrm{PNQ}}$, $o_i$))
          end if
        end if
      end if
    end for
```

```
34       if phase = heuristic then
35          if $|R_{\mathrm{PNQ}}| < k$ then
36             if $d(\mathrm{GetNext}(\mathbf{C}'), p_{\mathrm{NN}}) < r_{\mathrm{lim}}$ then
37                SendPartialQuery($k$, $p_{\mathrm{NN}}$, $\chi$, $\upsilon$, GetAndRemoveNext($\mathbf{C}'$))
38             else
39                DeliverResult($R_{\mathrm{PNQ}}$) // no more cells to aggregate from
40             end if
41          else // $|R_{\mathrm{PNQ}}| = k$
42             phase = aggregation
43             AggregateResult($k$, $p_{\mathrm{NN}}$, $\chi$, $\upsilon$)
44          end if
45       else // phase = aggregation
46          if AggregationComplete() then
47             DeliverResult($R_{\mathrm{PNQ}}$) // deliver result to client tier
48          end if
49       end if
50    end
51
52    procedure AggregateResult($k$: integer; $p_{\mathrm{NN}}$: Point; $\chi$: real; $\upsilon$: real)
53    begin
54       var $C_{\mathrm{agg}}$: Disk($p_{\mathrm{NN}}$,0)
55       var $r_{\mathrm{agg}}$, $d_{\mathrm{max}}$: real
56       $r_{\mathrm{agg}}$ := 0
57       for each $o_i \in R_{\mathrm{PNQ}}$ do
58          $d_{\mathrm{max}}$ := $\mathrm{Max}_{p \in L_i}\{d(p, p_{\mathrm{NN}})\}$
59          $r_{\mathrm{agg}}$ := $\mathrm{Min}\{\mathrm{Max}\{r_{\mathrm{agg}}, d_{\mathrm{max}}\}, r_{\mathrm{lim}}\}$
60       end for
61       if not Overlaps(GetNext($\mathbf{C}'$), Disk($p_{\mathrm{NN}}$,$r_{\mathrm{agg}}$)) then
62          DeliverResult($R_{\mathrm{PNQ}}$) // no more cells to aggregate from
63       else
64          while Overlaps(GetNext($\mathbf{C}'$), Disk($p_{\mathrm{NN}}$,$r_{\mathrm{agg}}$)) do
65             SendPartialQuery($k$, $p_{\mathrm{NN}}$, $\chi$, $\upsilon$, GetAndRemoveNext($\mathbf{C}'$))
66          end while
67          SetTimer(aggregationTimer, $t_{\mathrm{agg}}$)
68       end if
69    end
70
71    procedure TimerExpired()
72    begin
73       DeliverResult($R_{\mathrm{PNQ}}$) // deliver result to client tier
74    end
75 end
```

The processing of a PNQ starts at the query proxy by a call to procedure ProcessPNQ (line 12). After initializing the final query result, R_{PNQ}, with the empty set (line 14), the heuristic phase is entered (line 15) and a copy of the global cell set, $\mathbf{C}'$, is created (line 16). This set is a priority queue that contains cells in ascending order of distance to the query reference position p_{NN}. The first partial query is sent towards the reference coordinate that is associated with the nearest cell (line 17, ❷ in Figure 5.5). This is the point where the partial query is handed from the service to the storage tier, as was shown previously in Figure 5.4.

After the processing of a partial query at a data server in accordance with Definition 5.9, the corresponding partial query result is received by the query proxy and handled in procedure

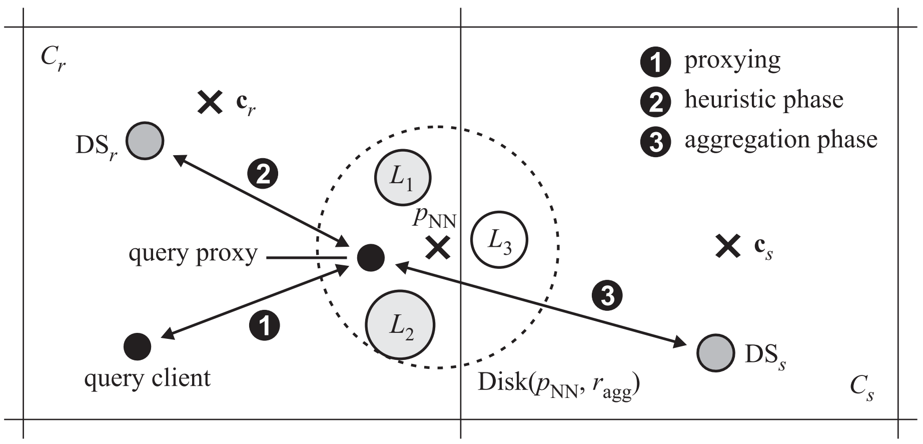


Figure 5.5: Probabilistic k-nearest neighbor query algorithm.

ReceivePartialResult (line 20). In Figure 5.5, after the issuing of a 2-nearest neighbor query, two objects with location area L_1, L_2 are returned by the heuristic phase's first partial query from DS_r to the query proxy. For both the heuristic and aggregation phase, the incoming partial result is merged into the final query result (lines 23-33). For this, every object o_i in $R^{\mathrm{part}}_{\mathrm{PNQ}}$ is tested for containment in R_{PNQ} (line 24). Only if a copy o_i does not already exist in R_{PNQ} and R_{PNQ} contains less than k objects (line 25), o_i is inserted (line 26). If R_{PNQ} already contains k objects (line 27), o_i replaces some o_j in R_{PNQ} only if o_i is nearer to p_{NN} than o_j and o_j is the object farthest away from p_{NN} from all other objects o_k in $R^{\mathrm{part}}_{\mathrm{PNQ}}$ (lines 28-29).

The heuristic phase continues by testing if k objects were not yet found (line 35). In this case, it is further tested whether more cells exist that may contain data objects whose distance to p_{NN} is less than the maximum search radius r_{lim} (line 36). This radius is essential to limit the search scope of a PNQ in case object density is low or for large values of k. It may either be specified by the query client or upper bound by the algorithm to avoid the dissemination of partial queries across the complete network. If additional relevant cells within r_{lim} exist, the query proxy sends consecutive partial queries towards the cells in $\mathbf{C}'$ in order of increasing distance to p_{NN} (line 37). Otherwise, the query terminates and delivers a final result that contains less than k objects to the client tier (line 39). If k objects were found during the heuristic phase, the aggregation phase is entered by a call to AggregateResult (line 41-43).

The objective of the aggregation phase is to find any other object that might be closer to p_{NN} than any of the objects gathered during the heuristic phase and currently stored in R_{PNQ}. Such objects may exist due to the fact that the cell structure in the network can be arbitrary and servers may thus store objects that are closer to the ones already collected. An example is shown in Figure 5.5, where the location area L_3 belongs to an object that is stored at DS_s and which is closer to p_{NN} than L_1. The first task of the aggregation phase is to determine the set of additional cells that must be queried to complete the final query result. For that, the smallest disk, $\mathrm{Disk}(p_{\mathrm{NN}}, r_{\mathrm{agg}})$, with center p_{NN} and radius r_{agg} is constructed and which contains the location area of every determined object in R_{PNQ} (lines 56-60). For each object (line 58), the most distant point inside this object's location area is determined and the aggregation radius r_{agg} is updated (line 59). Note that the radius of the disk is in any case limited by the maximum radius r_{lim}. Figure 5.5 illustrates the construction of $\mathrm{Disk}(p_{\mathrm{NN}}, r_{\mathrm{agg}})$ for the two objects with location areas L_1 and L_2. In the example, the disk contains the location area L_3 of another object that is stored on DS_s and thus, was not retrieved during the heuristic phase.

Based on the determined radius r_{agg}, the next potential cell from the priority queue is checked for overlap with $\mathrm{Disk}(p_{\mathrm{NN}}, r_{\mathrm{agg}})$ (line 61). Note that all cells that were already queried in the heuristic phase are left out in the aggregation phase, which is guaranteed by the fact that for each partial query sent during the heuristic phase, the corresponding cell is removed from the priority queue (line 17 and also 37). If no overlap occurs, the query result is already complete and is delivered to the client tier (line 62). Note that because the priority queue contains cells in increasing order to p_{NN}, it is not possible that another cell exists that overlaps with $\mathrm{Disk}(p_{\mathrm{NN}}, r_{\mathrm{agg}})$. In the case where the disk overlaps with at least the next cell in the priority queue (line 63), one partial query is sent for each overlapping cell to the server that is associated with this cell. In Figure 5.5, only cell C_s overlaps with $\mathrm{Disk}(p_{\mathrm{NN}}, r_{\mathrm{agg}})$ and thus, the associated data server DS_s is queried (❸). The sending of a partial query to this server makes sure that the corresponding object is eventually retrieved and merged into the final query result according to lines 23-33. After the sending of the partial queries, the aggregation timer is set to guarantee that the aggregation phase terminates eventually (line 67).

Once the partial queries of the aggregation phase are being propagated and processed at the corresponding server(s), the aggregation phase completes in either one of two cases. The regular case occurs in procedure ReceivePartialResult, lines 45-49. With the reception of each partial query result, it is checked whether the aggregation is complete (line 46). This can be accomplished by matching the number of received partial query results with the number of partial queries sent during the aggregation phase. If the result is complete, the query result is delivered to the client tier (line 47). Otherwise, the procedure finishes and the query proxy waits for the outstanding partial query results.

In the second case, aggregation is terminated due to the expiration of the aggregation timer, which was set in line 67 and which occurs prior to the reception of all outstanding partial query results. Upon expiration, procedure TimerExpired is invoked (line 71). In Listing 5.2, the query result, which may be suboptimal, is immediately delivered to the client tier (line 73). In the presented algorithm, only best-effort delivery of query messages is supported. It is straightforward to extend the algorithm by any level of additional robustness to increase accuracy, also in collaboration with the storage tier. However, these details are omitted in this dissertation. Instead, it is shown in Section 5.5 that even under best-effort delivery, a large degree of query accuracy can be achieved in typical mobile ad-hoc network scenarios.

5.5 Performance Analysis

This section discusses a representative set of evaluation results of the service tier. We will show that the service functionality provided for context-based applications can be supported in an efficient way based on the storage tier introduced in Chapter 3 and 4 and in conjunction with the location updating mechanism elaborated in Section 5.3.

When considering the algorithms for probabilistic range and k-nearest neighbor queries in Section 5.4, we can observe that both types of queries are decomposed into a number of partial queries that process the overall query result. While the individual strategy of each query varies considerably, the decomposition in the case of range queries is much more deterministic than for k-nearest neighbor queries, where the query parameter k and the node density both influence the required number of partial queries. We can therefore understand the performance of range queries well in qualitative terms from the analysis of k-nearest neighbor queries. In the following, we will therefore restrict our analysis to k-nearest neighbor queries only.

In the performance evaluation of k-nearest neighbor queries, we make use of comparative analysis in a way that allows us to understand in particular the impact of position inaccuracy on the accuracy of query results. For that, we define two optimum cases that will be used as a reference for our comparison. The details of these cases will be presented in conjunction with the performance metric of query accuracy in Section 5.5.1.

Table 5.2 shows the default system parameters used in this section in addition to the general system parameters that were introduced in Table 3.3.

System Parameter	**Default Value**
Simulation time	360 s
Number of cells	4, equally-sized
Cell size	$300 \cdot 300$ m^2
Number of objects	300
Mobility model (for objects)	random waypoint
Object speed (fixed for all objects)	1.5 m/s
Pause time (fixed for all objects)	30 s
Sensing range	20 m ($\approx$ RFID)
object observation interval	3 s and upon discovery
Position inaccuracy	5 m ($\approx$ GPS)
object lifetime	10 s (soft state)
Query parameter k	3
Number of queries	300
Query execution interval	$[30$ s$, 330$ s$]$
Query frequency	1 query/s

Table 5.2: Query processing: system parameters.

The simulation time is set to 360 s, which is sufficient to issue a large number of queries in order to achieve statistical stability in the simulation results. In contrast to core data storage and data migration, the total simulation area is subdivided into four disjoint cells, each $300 \cdot 300$ m^2 in size and covering the total simulation area.

The evaluation scenario further considers, in addition to network nodes, observable objects that are located inside of the simulation area. These objects are assumed to be observed by sensor technology embedded in the nodes, and they may represent any type of other entities located in the vicinity of network nodes. These objects are used to show how query accuracy is impacted by inaccurate position information and an incomplete view of the physical world in the model data stored in the network. We assume 300 observable objects by default, each moving according to the random waypoint mobility model, with a fixed speed of 1.5 m/s and a fixed pause time of 30 s. These values correspond to the settings used for nodes.

We assume that the sensing range of embedded sensors is 20 m. Any observable object located within the sensing range of a node is observed by that node. An object is observed for the first time when it enters the sensing range of a node and is thereby discovered by that node, then in regular time intervals of 3 seconds until it leaves the sensing range of the node again.

We assume that a node determines the position of an object upon observation and is able to do so with a position inaccuracy of 5 m, which corresponds to values that can be obtained from typical GPS receivers. After each observation, a data object is created and sent to the server that is located in those cells whose area overlaps with the location area of the object's determined location according to Listing 5.2. Any object that is not updated within at most 10 seconds at a server is removed from that server by a soft state approach.

5.5.1 Performance Metrics

We define three key performance metrics to assess the performance of k-nearest neighbor queries: query accuracy and query offset, query latency, and query cost.

Query Accuracy and Query Offset

Query accuracy and *query offset* are two related performance metrics that describe the quality of the result returned by a k-nearest neighbor query. We express both metrics with respect to two optimum cases, which we term model-based optimum and real world-based optimum.

The purpose of comparing the k-nearest neighbor query result to the *model-based optimum* result is to assess the pure performance of the query algorithm, that is, which level of accuracy it is able to achieve with respect to a perfect reference implementation. Such an implementation assumes that a query is processed instantaneously, based on the most recently observed information that is stored at a single server at query time. This reflects the best possible case that is, in theory, achievable in a mobile ad-hoc network at all.

Let $R_{\text{mod}} = \{o_1, \ldots, o_s\}$ and $R = \{o_1, \ldots, o_t\}$ denote the model-based optimum result and the result returned by the k-nearest neighbor query implementation, respectively. The accuracy of a single k-nearest neighbor query with respect to the model-based optimum result, denoted A_{mod}, is defined as $A_{\text{mod}} = |R \cap R_{\text{mod}}| \ / \ |R_{\text{mod}}|$.

The query offset is a second intuitive way to assess the quality of a k-nearest neighbor query result and it expresses by how much the objects returned in the result are "off" from the reference result. Assume, for instance, that in a 3-nearest neighbor query, a single object is missed and the fourth-nearest instead of the third-nearest object is returned. The query offset in this case is one. The important property is that for highly inaccurate query results, the query offset can still have a small value and indicates how "far", literally, the returned objects are from the reference result. In other words, even if the query accuracy is very low, the offset may indicate that the result is still useful for specific applications. The offset of a single k-nearest neighbor query with respect to the model-based optimum result is defined as follows.

Let w.l.g. $\forall i : d(\mathbf{r}_i, \mathbf{c}_r) < d(\mathbf{r}_{i+1}, \mathbf{c}_r)$ for a reference coordinate $\mathbf{c}_r$. Let further $R'_{\text{mod}} = \{o_1, \ldots, o'_s\}$ denote the smallest set of objects for which $R \subseteq R'_{\text{mod}}$. The offset of a single k-nearest neighbor query with respect to the model-based optimum result, denoted O_{mod}, is defined as $O_{\text{mod}} = |R'_{\text{mod}}| - |R| \geq 0$. Note that $O_{\text{mod}} = 0 \Leftrightarrow A_{\text{mod}} = 1$.

The *real world-based optimum* result reveals the impact of incomplete and inaccurate observations of the physical world on query performance. It is defined as the result that would be returned by a "perfect" k-nearest neighbor query that is executed instantaneously based on the configuration of physical objects in the real world. The query accuracy A_{phy} and query offset O_{phy} with respect to this optimum are defined analogous to the model-based optimum.

Query Latency

According to the query algorithm described in Section 5.4, a query is first sent to a query proxy, which aggregates the query result from a number of partial query results, before it sends the result back to the proxy again. Because the latency from the original query client to the proxy and back corresponds to a request, whose performance we have evaluated in detail in Section 3.5, we exclude it from the computation of the query latency. Rather, we define *query latency* as the mean time from the beginning of the heuristic phase to the termination of the aggregation phase of a number of k-nearest neighbor queries. This value yields the pure performance of the distributed algorithm that actually processes the query result.

Query Cost

As in the case of query latency, query cost only considers packets that are part of the distributed query processing algorithm. Thus, *query cost* are defined as the mean number of packets sent during the heuristic and aggregation phase of a number of k-nearest neighbor queries.

5.5.2 Query Accuracy and Query Offset

Figure 5.6 through 5.8 show the results for query accuracy and offset. Figure 5.6 shows both metrics as a function of the number of objects. Note that according the the observation model with the parameters given in Table 5.2, a specific number of objects corresponds to a certain update frequency, which is displayed in the second row below the abscissa.

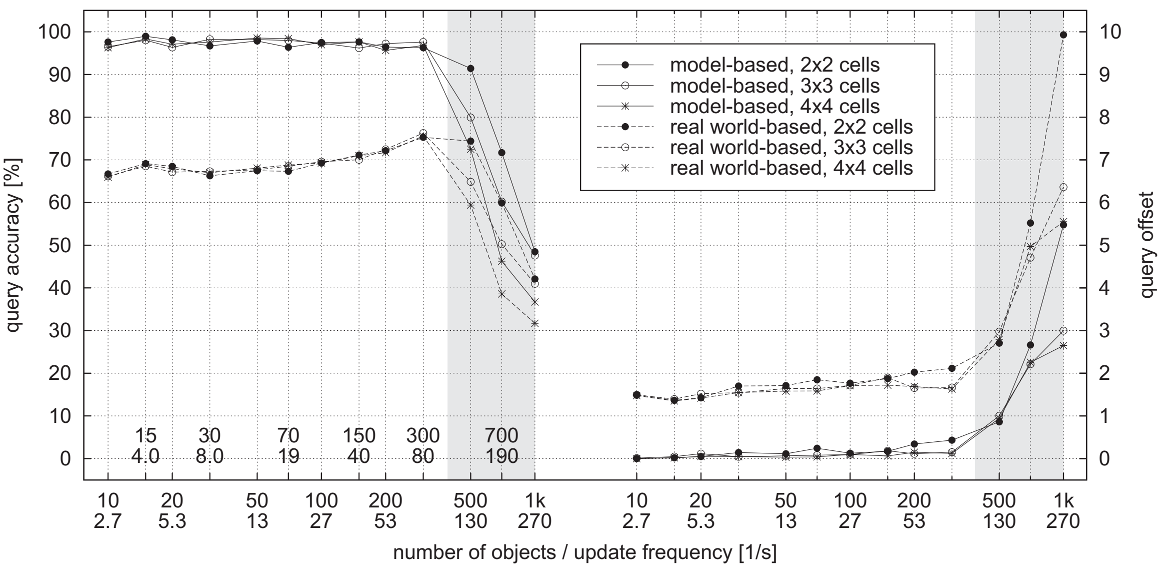


Figure 5.6: Query accuracy and offset as a function of the number of objects / update frequency.

The first observation is that both query accuracy and offset with respect to the model-based optimum show strong performance for all considered cell sizes. This holds in the range of up to about 300 objects, which corresponds to approximately 80 updates per second. In this range, objects are at a sufficient distance from each other such that position inaccuracies do not have an impact on performance in terms of query accuracy and offset.

Above 300 objects, in the region that is indicated by the grey-shaded area, query accuracy with respect to both optima drops significantly, and query offset increases. From Figure 5.6 alone we cannot deduce directly whether this is due to congestion-related packet loss that occurs at higher update frequencies or due to position inaccuracies. We can, however, distinguish this fact by consulting Figure 5.9 and 5.10. In Figure 5.9, we can observe that for 500 and more objects, latencies increase significantly, which clearly supports congestions due to high packet rates. However, Figure 5.10 does not confirm that these congestions also lead to packet loss, because in the considered higher range of update frequencies, the decrease in the query cost follows the regular behavior that we discuss later on with respect to this figure.

Having excluded the above possibility, we can conclude that the drop in the query accuracy and increase in query offset is due to the influence of position inaccuracies at high object densities. With increasing object densities, the chance of overlapping location areas of different objects also increases. Due to limited update frequencies, the configurations of overlapping location areas also develop in the form of discrete events over time. For example, objects whose location areas have not overlapped before a position update might do so after the update has occurred. This fact together with the observed latency in Figure 5.9 explains the steep gradient in the query accuracy with respect to both the model-based and real world-based result. With increasing query latency, the processing of a query extends over a longer period of time during which many position updates occur. However, in both optimum cases, a query is processed instantaneously and is not exposed to interleaved position updates.

A further observation is that the query accuracy with respect to the real world-based optimum is significantly smaller than with respect to the model-based optimum. This is due to the incomplete observation of the real world. When considering the query offset, we observe that the difference between both optima is only about 1.5. This effectively means that only between 1 and 2 objects are missed by average in each query result in comparison to the real world-based optimum. This is an important result with respect to the fact that a complete observation of the physical world is limited by the sensing technology's coverage and cannot be changed. Even in the case of an incomplete view of the physical world, the k-nearest neighbor query result as *perceived* by the user still reflects an acceptable result for applications that do not rely on perfect information, which holds for many kinds of location-based applications.

Examining the different settings for the cell size, we identify a slight advantage for larger cell sizes. This observation is due to an undesired but unavoidable artefact in the simulation. Any queries executed near the border of the simulation area incur smaller communication cost, because they do not have neighboring cells in some directions. For fewer cells, this effect has a larger impact and decreases query cost to a larger extent.

The final observation in Figure 5.6 is that the slope of query accuracy with respect to the model-based query result increases slightly between about 30 and 300 objects. This occurs because for a very low number of objects, a larger number of partial queries are required per query. This increases the chance that at least one partial query per k-nearest neighbor query fails, in turn leading to the missing of some objects from a specific data server.

Figure 5.7 shows the query accuracy and offset as a function of the query frequency. We observe that query accuracy is high and query offset low in the range where there is no congestion. Further, the difference between the model-based and real world-based result shows up clearly. In both cases, the same arguments as in Figure 5.6 hold.

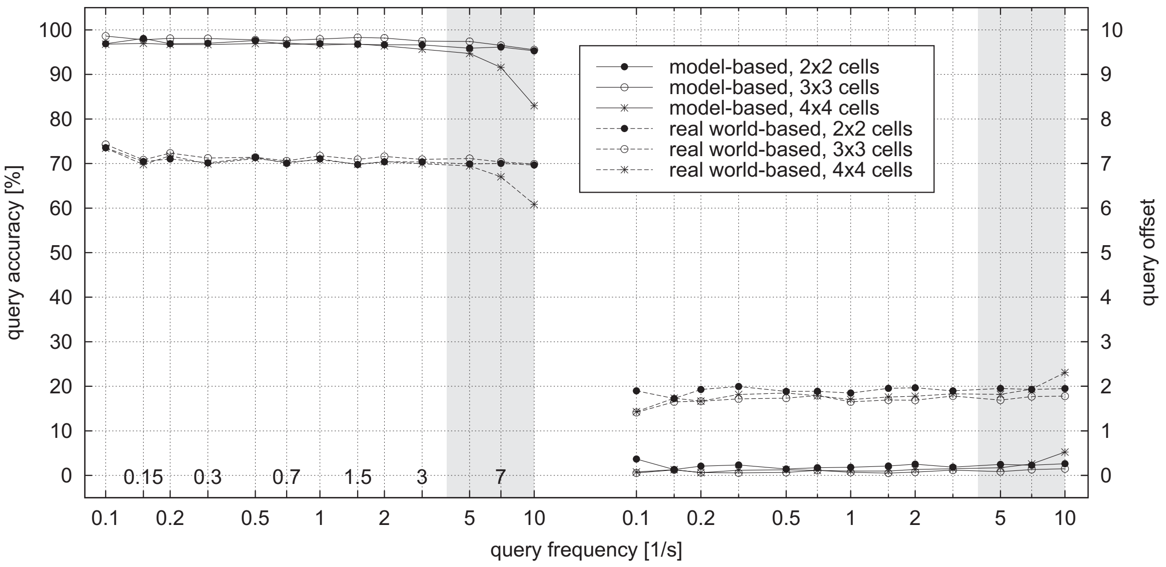


Figure 5.7: Query accuracy and offset as a function of the query frequency.

Similar to increasing the number of objects and in turn the number of requests in Figure 5.6, an increase in the number of queries eventually leads to congestion. For the examined query frequencies, this occurs only for the case of 4×4 cells. This frequency is in fact extremely high and can be pictured by imagining that every single one of the 150 nodes in the scenario is able to execute a 3-nearest neighbor query every 15 seconds.[2] The region of congestion further indicates that larger cells are to be preferred over smaller cells. This fact can be understood in terms of query cost and will be discussed in more detail in conjunction with Figure 5.10.

Figure 5.8 depicts the results for query accuracy and offset as a function of the query parameter k. The first observation is that with respect to the model-based optimum, parameter k has no significant influence on query accuracy. Even for $k = 15$, accuracy is well above 95%. Note the slight and linear increase in the corresponding query offset. This is due to the fact that with larger k, the probability to miss objects increases linearly, because objects are roughly homogeneously distributed in the network due to the random waypoint mobility model. Note that even at $k = 15$, the query offset is only about 0.5, indicating that in at least 50% of the queries, all 15 objects match the model-based optimum result.

The second observation is that the query accuracy with respect to the real world-based optimum decreases noticeably for smaller values of k and shows an asymptotic behavior with larger values

[2]Note that the continuous observation of a certain number of nearest objects is in fact a different class of queries, namely, continuous queries, which we do not consider in this dissertation. For such queries, rather than a continuous sequence of k-nearest neighbor queries being executed at a high frequency, very specialized algorithms are required to deliver sufficient performance.

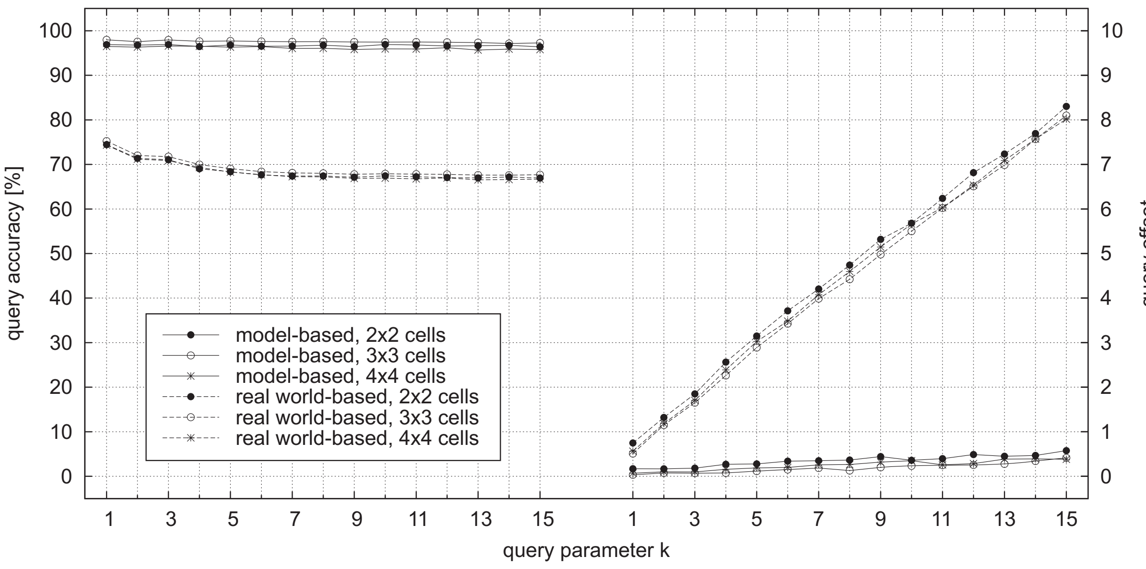


Figure 5.8: Query accuracy and offset as a function of the query parameter k.

of k. This is because for small values of k, the chance of only a single object being missed in the query result increases. With increasing k, the number of the missed objects then increases linearly, because the missed objects are roughly homogeneously distributed in the network. The linear increase is confirmed by the query offset relative to the real world-based model.

5.5.3 Query Latency and Query Cost

Similar to migration efficiency and migration cost, a strong correlation can be observed between query latency and query cost. We therefore present both simulations together in Figure 5.9 through 5.14. Figure 5.9 and 5.10 show query latency and cost as a function of the number of data items. The update frequency, which is a function of the number of objects, is displayed in the second line below the abscissa of both figures. Each bar has two parts, showing the latency and cost portions that are due to the heuristic and aggregation phase.

We observe that with an increasing number of objects, the total query latency decreases and reaches its optimum of between 80 ms and 190 ms, depending on the considered cell size. These values are remarkably small given a distributed query algorithm that is based on two sequentially executed phases. The decrease of query latency is due to an increase in the object density, where most objects relevant for the query evaluation can be retrieved already in the single cell that contains the query position. In turn, fewer partial queries need to be sent, which can be seen in the correlating decrease of query cost in Figure 5.10.

This performance behavior becomes especially clear when considering the latency incurred by the heuristic phase only. In this phase, the latency decreases significantly, because for small object densities, partial queries of the heuristic phase may have to be disseminated in sequential order to more distant cells, based on the algorithm in Listing 5.1. In contrast, the latency of the aggregation phase drops, however, slower than the latency of the heuristic phase. The decrease is due to the fact that the heuristic phase is able to determine a set of candidate objects that

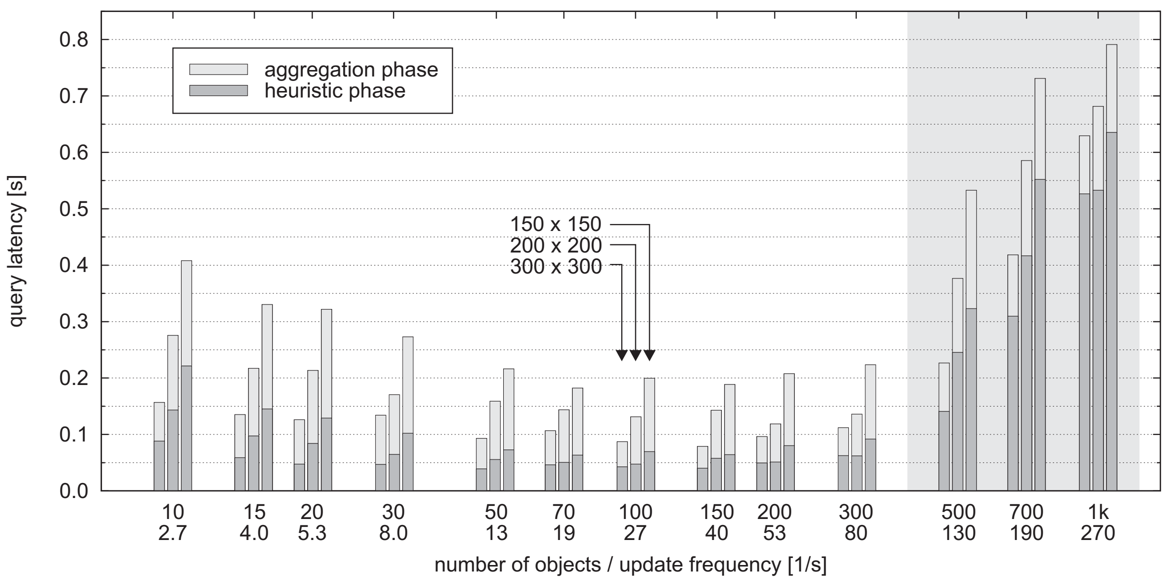


Figure 5.9: Query latency as a function of the number of objects / update frequency.

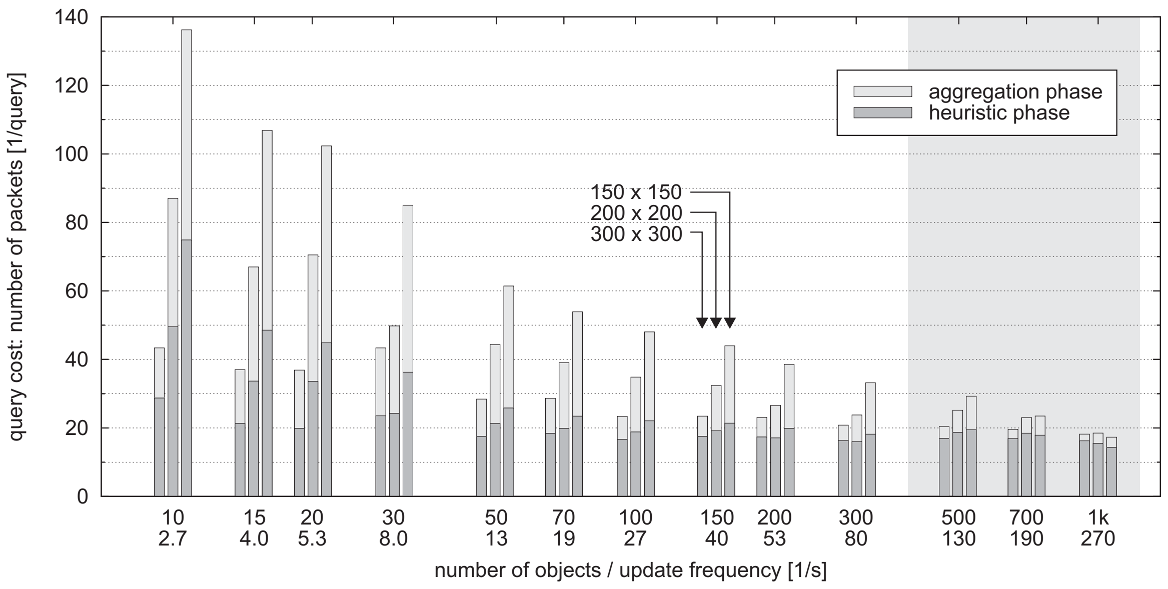


Figure 5.10: Query cost as a function of the number of objects / update frequency.

are closer to the actual result with increasing number of objects. Thus, fewer partial queries have to be sent during the aggregation phase to collect the remaining relevant objects. The magnitude of the drop is smaller because the partial queries of the aggregation phase are all sent simultaneously to the servers associated with the remaining cells.

In contrast to query latency, Figure 5.10 shows that the magnitude of the decrease in query cost with increasing number of objects is at a similar magnitude for both the heuristic and aggregation phase. This is because whether partial queries are executed sequentially or in parallel does not have any effect on the query cost.

Coming back to Figure 5.9, we observe a slight decrease in query latency starting from about

200 objects. The magnitude of query latency becomes very significant at 500 and more objects, which is indicated by the congestion region. As we have discussed previously in conjunction with Figure 5.6, the latency does not imply congestion-related packet loss. This can be understood in more detail by considering the mechanisms that cause packet latency in the first place. While 802.11's CSMA/CA protocol in conjunction with the RTS/CTS scheme avoids many collisions of data frames on the wireless medium, a growing number of packets yet to be transmitted become queued up in a node's packet queue. Thus, query latencies that increase with the number of objects are due to increasing queueing delays, which are in turn the result of larger update frequencies that incur many packets sent over the shared medium. The observation that no congestion-related packet loss occurs is confirmed by Figure 5.10, which shows that query cost decrease up to the maximum number of 1000 objects.

Comparing different cell sizes, we observe that larger cells are to be preferred over smaller ones in terms of both query latency and cost. The arguments for the advantages in both the aggregation and heuristic phase are identical to the case where node density increases, because increasing either cell size or node density has the same effect on the query algorithm.

Figure 5.11 and 5.12 show query latency and cost, respectively, as a function of the query frequency. We observe that both query latency and cost are virtually constant for all settings of the cell size up to three queries per second. For the largest cell size, query latency remains below 150 ms, a good value taking into account the quite high query frequency.

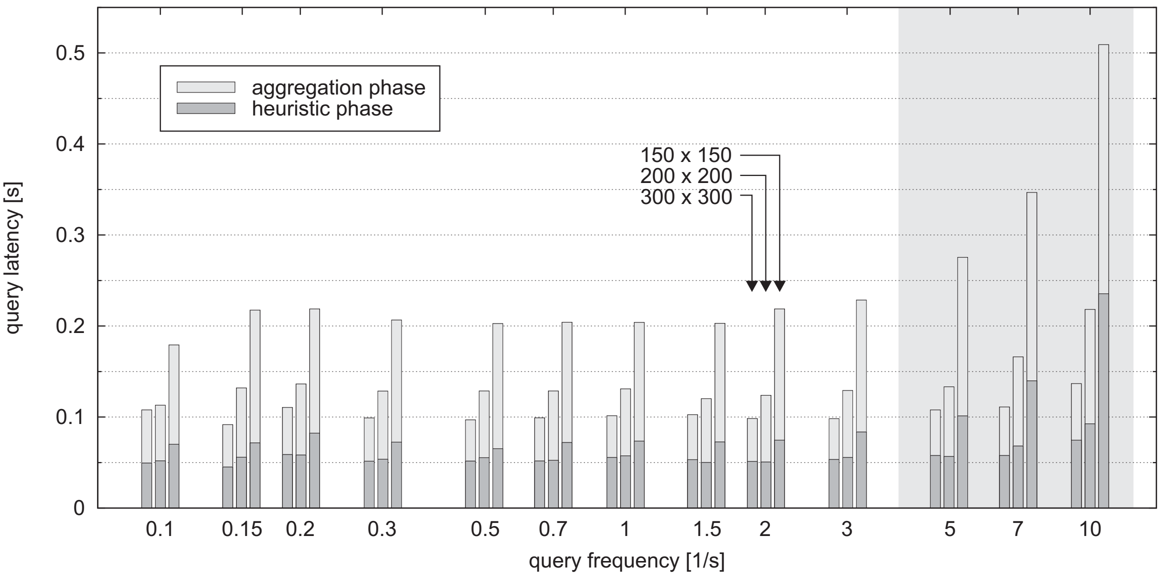


Figure 5.11: Query latency as a function of the query frequency.

In the upper range of query frequencies, congestions occur for cell sizes of $200 \cdot 200$ m^2 and $150 \cdot 150$ m^2, in which cases a significant increase in the query latency can be observed. However, only the latency occurring for the smallest cell size impacts query accuracy, as shown in Figure 5.7. As discussed in conjunction with Figure 5.10, the impact on query accuracy stems from the increased number of interleaved position updates that occur during a longer query processing time. Query cost are, nevertheless, not adversely impacted in the congestion region of Figure 5.12. This is because latencies occur as a result of queuing delays and do not lead to packet loss, as elaborated in detail in the discussion of Figure 5.10.

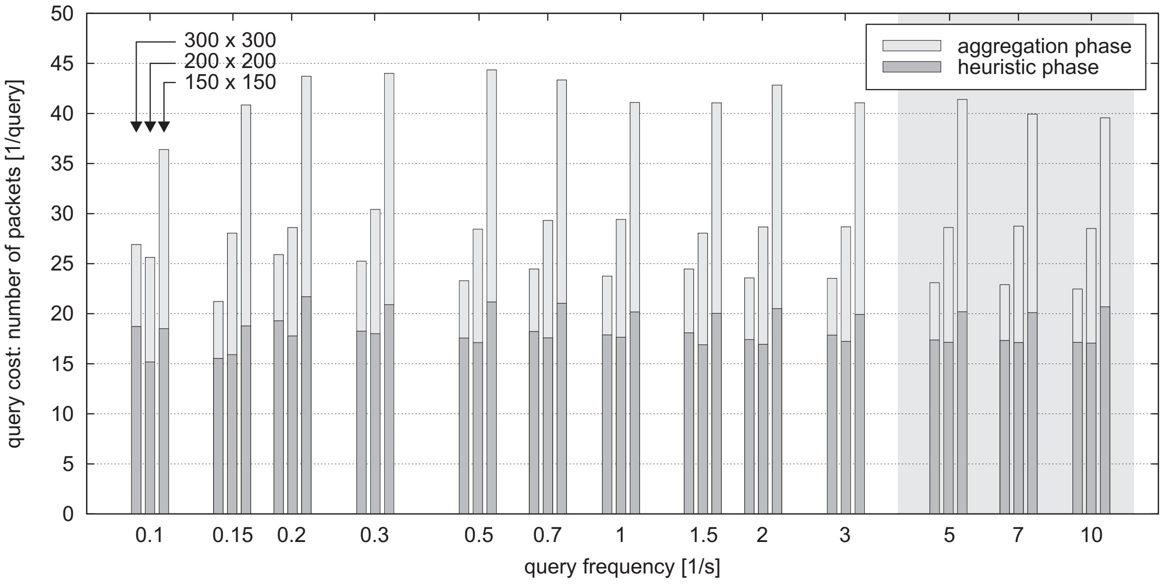


Figure 5.12: Query cost as a function of the query frequency.

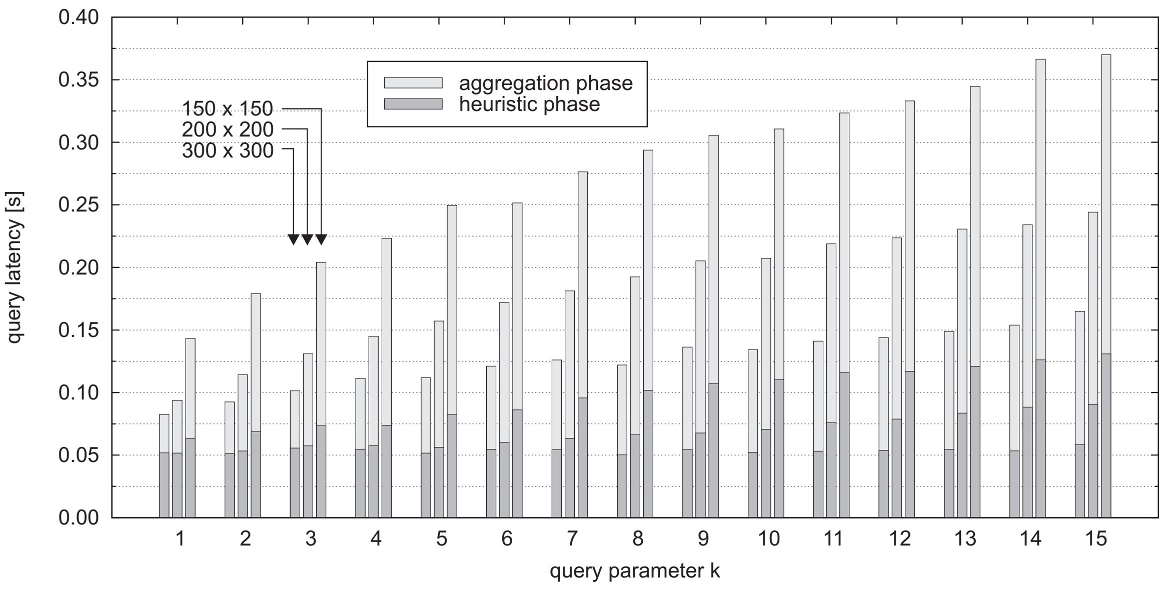


Figure 5.13: Query latency as a function of the query parameter k.

Figure 5.13 and 5.14 show the query latency and cost, respectively, as a function of the query parameter k. We observe that with larger values of k, query latency and cost increase noticeably. This is because a larger number of objects to be included in the result implies more distant objects that need to be considered. Thus, a larger number of partial queries have to be processed to retrieve such objects from a larger number of adjacent cells.

We further observe that the query latency and cost of the heuristic phase increase noticeably slower than in the case of the aggregation phase. This behavior is due to the fact that the heuristic phase's first partial query is mostly sufficient to retrieve the required k initial objects. With growing k, the need for additional queries increases only slowly. However, as k increases,

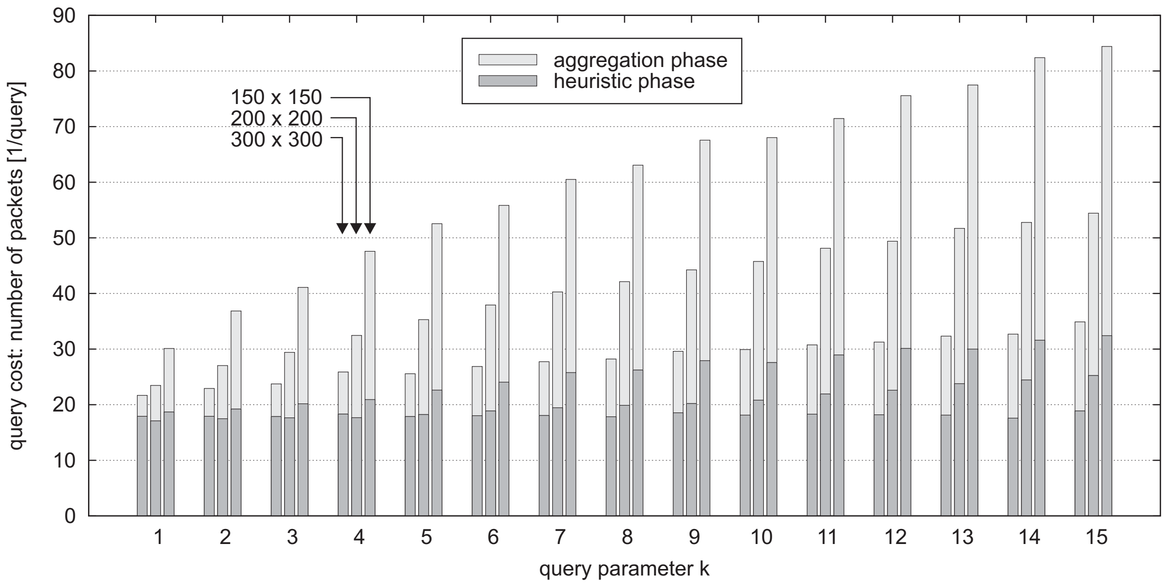


Figure 5.14: Query cost as a function of the query parameter k.

the aggregation phase must query a larger number of more distant cells. In order to reach servers in these cells, a longer routing path must be traversed, which incurs increased latency and cost that contribute to the overall latency and cost of the aggregation phase. Additionally, the chance of failures of partial queries that are routed via longer paths increases gradually, which leads to a slight increase in the probability of aggregation timeouts. This effect additionally contributes to the increase in the query latency of the aggregation phase.

Finally, Figure 5.13 and 5.14 show that large cell sizes incur significantly lower query latency and cost, which confirms that larger cells are generally to be preferred over smaller ones.

5.5.4 Evaluation Summary: Service Tier

From the evaluation of the processing of k-nearest neighbor queries we are able to summarize the following key observations. Firstly, the results show that the algorithms for probabilistic k-nearest neighbor queries perform generally well over various system parameters and a wide range of parameter values. Specifically, query accuracy, the key metric that is immediately visible to the user, shows good results with respect to what the algorithm implementation in a mobile ad-hoc network is able to achieve within the limitations of inaccurate position information and an incomplete view of the physical world.

Second, the results show that larger cells are generally to be preferred over smaller cells. This behavior is inverse to the observations of the core data storage and data migration algorithms in Section 3.5 and 4.6, respectively, and necessitates further studies that examine the most suitable cell size in specific scenarios, which is beyond this dissertation.

Third, we can confirm that the significantly smaller query accuracy achieved with respect to the real world-based result is due to the incomplete view of the physical world available in the stored model of the databases distributed in the network. The key fact is that this limitation is of a general kind and cannot be directly improved by algorithm extensions. Instead, additional technical modifications become necessary, such as the increase of sensor coverage to capture a larger fraction of objects. Because the optimum cases are idealizations based on a global view and cannot be determined in real systems, a fundamental additional research challenge is how *completeness* of the data that is available in the distributed storage of the mobile ad-hoc network can be measured by potential indirect means. This is essential for providing applications with an idea of what is known about the physical world at all. This topic is clearly a research challenge of its own right and beyond the scope of this dissertation.

Fourth, we have observed that for a large number of objects, accuracy is strongly impacted with respect to both the model-based and real world-based optimum results. We were able to attribute this to the impact of inaccuracy that eventually hinders the resolution of the relative location of objects in the physical world. The decrease in accuracy is amplified by the influence of query latency, which adds a significant temporal window that allows for more inaccuracies to aggregate during the processing of a single query. These results have clearly shown that latency is to be treated as a key system parameter, due to its indirect impact on query accuracy. While many algorithms for wireless multihop networks are mainly designed towards achieving efficient communication, latency becomes similarly important when it comes to increasing the accuracy of spatial queries. This shift in priorities suggests to consider a more detailed analysis of query cost versus query latency, e.g., by introducing concurrent sending of partial queries also during the heuristic phase of the k-nearest neighbor query algorithm.

While achieving perfect query results in terms of query accuracy was shown to be unfeasible due to an incomplete view of the physical world and significant position inaccuracies, it still remains possible to achieve query results with a practical offset. This is an important result since applications that are able to work with slightly "off" k-nearest neighbor query results can still use these results to a sufficient extent for many useful scenarios.

Chapter 6

Conclusion

6.1 Summary and Conclusions

With the advancements in computer, communication, and sensor technology, it is becoming feasible to collect and process large amounts of dynamic information from the physical world and to distribute it to context-based applications. In order to provide such information in an efficient and scalable way and to support its sharing among many types of applications, storage facilities that take over the management of such data are indispensable.

This dissertation pursued the goal of managing dynamic context information in infrastructureless networks. To this end, a framework was proposed that implements a set of algorithms for the efficient, robust, and scalable management and querying of dynamic data in mobile ad-hoc networks. In particular, the dissertation provides the following innovative contributions:

(1) We have discussed and organized the specific characteristics of mobile ad-hoc networks with respect to their impact on the management of dynamic data in these networks. As a result of weak network connectivity and device mobility, network partitioning occurs, which we have analyzed in detail in quantitative terms.

(2) Based on (1), we have defined a number of key requirements that act as constraints for the management of dynamic data in infrastructureless networks. A suitable data storage framework was proposed, comprising tiers for routing, storage, and services, each addressing a subset of the identified requirements. A thorough study of the relevant literature was presented in connection with the framework.

(3) We have designed fundamental mechanisms for data storage in mobile ad-hoc networks that form the main contribution of this dissertation. A set of algorithms for data storage and data migration was presented in conjunction with routing tier mechanisms that collaboratively achieve efficient and robust storage of dynamic data subsets at dedicated geometric locations in large-scale mobile ad-hoc networks.

(4) We have devised an analytical model of previously unmatched detail for the assessment of the communication cost incurred by the storage tier algorithms. We have provided a taxonomy and analytical study of a wide range of related storage approaches in comparison to our own approach and identified the suitable and advantageous operational ranges of the storage tier in terms of the most relevant system parameters.

(5) On the level of the storage framework's service tier, we have defined a probabilistic model for inaccurate position information for the realistic case where only inconclusive statements regarding a physical object's whereabouts are possible. On this basis, new semantics for spatial queries were defined, including probabilistic range and k-nearest neighbor queries, which give useful results based on the probabilistic location notion.

(6) Using the probabilistic models for inaccurate position information, we have proposed a corresponding position updating algorithm that builds on a geometric overlay on top of the storage tier. Consistent with location information updating, scalable algorithms for creating accurate range and k-nearest neighbor query results were developed to operate in accordance with the probabilistic query definitions under (5).

(7) Complementing the analytical results in (4), we have carried out an extensive comparative framework evaluation that assesses key performance metrics in dependence of the most relevant system parameters based on a uniform simulation methodology. The presented simulation results show the excellent performance of the developed algorithms and confirm that the storage framework meets its design requirements.

6.1.1 Network Characteristics and Network Partitioning

As an important prerequisite to the understanding of which factors need special consideration in the management of dynamic data in mobile ad-hoc networks, we have discussed in detail the particular characteristics of this type of network. In strong contrast to fixed networks, we have identified node mobility and network density as being the most important parameters. This is true because both parameters strongly impact the connectivity of infrastructureless networks, which is vital to the ability of nodes to communicate with each other when they are not within mutual transmission range. An important observation was that even highly dense networks are likely to form multiple network partitions at a time, allowing only the nodes in the same partition to communicate with one another. Due to the importance of this observation, we have conducted a detailed quantitative study of the partitioning behavior in mobile ad-hoc networks. For that purpose, we have introduced a compact set of metrics that meaningfully describe various properties of network partitioning. For instance, the rate at which partitions split and join was a suitable observable to judge whether the network is sufficiently stable to rely on static management structures, e.g., distributed aggregation trees.

While node mobility and partitioning in wireless multihop networks are generally undesirable properties, we were able to conclude that their acting together can also be beneficial. From the analysis of the metrics describing dynamic partition characteristics, such as the mentioned partition change rate, we found that the joining of once separated parts of the network occurs with a certain regularity under the assumption that the nodes' mobility is rather unpredictable. Because this is the general case in mobile ad-hoc networks formed by autonomous mobile users, we consistently took these results into consideration in the storage tier, and in particular, in the data migration approach. We advocate that network partitioning is worth to be more explicitly considered also in other domains, and that the proposed partition metrics will provide a valuable basis to be exploited in the improvement of algorithm performance, e.g., in the domain of delay-tolerant data dissemination and communication.

6.1.2 Location-centric Storage Paradigm and Framework

Equipped with an understanding of the relative importance of network characteristics and network partitioning, a framework for data management in mobile ad-hoc networks was developed. Three important additional constraints had to be considered with respect to the characteristics of the data to manage. Firstly, we focused on highly dynamic data, whose efficient management is challenging due to the need for frequent updates. Second, because of the natural imperfection of sensor technology, the sensing of data inevitably leads to degradation in the observed data, which requires to consider data quality aspects. Third, the specific context type *location* was considered due to its importance for location- and context-based applications. The third aspect lead us to the definition of the paradigm of location-centric storage, according to which data items are stored in close vicinity to a geometric location in the network.

Based on the discussion of network and data characteristics and the concept of location-centric storage, a framework for dynamic data storage in mobile ad-hoc networks was introduced. The framework is composed of three tiers, namely, routing, storage, and service, each addressing specific problems that relate to the network and data characteristics. On one side, the routing and storage tier contain the necessary mechanisms to make data storage efficient, robust, and scalable. On the other side, aspects of data quality were considered in the service tier, without impacting the more basic mechanisms in the storage tier.

In conclusion, we agree with the related literature on the general layout of a data management framework for wireless multihop networks. However, a stronger emphasis on the lower tiers is still needed, specifically, on the storage tier. This is largely due to the mobile ad-hoc networks' peculiarities that have an immediate impact on the lower tiers' performance. From the discussion of literature, we see significant open research challenges, some of which we raise in Section 6.2. We further state that while location-centric storage was used in conjunction with only geometric coordinates, it is flexible enough for being used also with more general location models, such as symbolic and hybrid ones, which we briefly discuss in Section 6.2.2.

6.1.3 Core Data Storage and Data Migration

The main contribution of this dissertation is the design of the framework's storage tier mechanisms. To tackle the complexity of this task, the storage tier was structured into two closely collaborating parts around the paradigm of location-centric storage. Based on geometric reference coordinates, core data storage algorithms implement the necessary mechanisms to store data on suitable nodes in the vicinity of defined geometric locations. In contrast to related approaches in the literature, the proposed approach is independent of any critical geometric or topological assumptions. Additional mechanisms in the routing tier make sure that requests are reliably routed between clients and storage nodes. To address the mobility of data when storage nodes are on the move, a data migration approach provides mechanisms to relocate data between nodes. This functionality is essential to retain data close to its assigned locations in order to be in line with the location-centric storage paradigm. The fact that network partitions may occur but can be expected to eventually join again was exploited in both the core data storage and data migration approach. In both cases, network partitioning was tolerated without corrupting overall efficiency. Upon the joining of partitions, suitable detection and consolidation mechanisms were implemented to recover from undesired redundancies.

The proposed storage and routing tier have shown to outperform related approaches in terms of efficiency, robustness, and scalability in the targeted range of operation. Comparative evaluations have revealed that the strength stems in particular from the independence of specific geometric assumptions. In contrast, related approaches heavily rely on geometric and topological structures and are inefficient for dynamic data storage in our context. The results clearly show the feasibility of implementing storage mechanisms that are able to deal with a large number of dynamic data items that lead to many updates in the network. It was also shown that the storage tier is able to maintain performance when the critical system parameters are varied over a wide range of settings. This fact makes the algorithms suitable for mobile ad-hoc networks that are unpredictable due to the autonomy of network nodes.

It is nevertheless obvious that mobile ad-hoc networks pose a number of unsurmountable limits with respect to data management performance. Firstly, the high geographical dispersion of network nodes and the wireless communication channel's capacity limitations eventually lead to efficiency problems that are due to increasing network loads. Second, some physical limitations are still applicable to the network topology, and it is impossible to guarantee connectivity in a pure ad-hoc network. While the proposed algorithms are able to tolerate network partitions of finite duration and it is conceivable to use forms of adaptation to further improve efficiency, it is naturally impossible to recover from permanent network partitioning. Thus, in a pure ad-hoc network model, uncertainties will remain with respect to which degree of accuracy, timeliness, and consistency of the managed data can be provided. It will be worthwhile to see how hybrid system structures can be used to support the ad-hoc network in critical situations. This kind of *assisted* ad-hoc will be briefly discussed in Section 6.2.4.

6.1.4 Probabilistic Location Updating and Query Processing

To show the suitability and practical value of the proposed storage tier algorithms for higher-level services and applications, we have implemented algorithms in the service tier for location updating and spatial query processing. To account for the imperfection of physical sensor technology, we have defined a probabilistic model for location information that considers the fact that the position of an object cannot be pinpointed with absolute certainty in a bounded spatial region. Based on these semantics, we have defined compatible semantics for probabilistic spatial queries, focusing on range and k-nearest neighbor queries as two of the most important types of spatial queries in many location-based application scenarios.

To support the storage of location information with the proposed inaccuracy semantics, a geometric overlay was introduced to map location information of the service tier to the reference coordinates of the storage tier. A key benefit of this overlay is that it can be set up independently from the storage tier's management structures, thereby allowing its layout to be adapted, e.g., to topographical constraints that are more appropriate in the environment where the network is deployed. We have shown that probabilistic range and k-nearest neighbor queries can be processed efficiently even under a probabilistic model for location information. Further, it was shown that accurate results can be delivered within the limits of the accuracy model.

Two important conclusions can be drawn from the analysis of probabilistic location updating and query processing. Firstly, due to the location inaccuracies, it is impossible to achieve query results that perfectly match the physical reality. We found that with increasing object density, certain types of queries, such as the considered k-nearest neighbor query, introduce unavoidable uncertainties in the query result. The reason was due to the relative location between objects that cannot be resolved with perfect confidence. Second, it was shown that the managed data is prone to incompleteness due to only partial sensor coverage. A subset of objects in the physical world may simply not be available at some times and thus, incompleteness becomes an important metric to consider. While many applications are able to deal with query results that diverge from the physical reality due to incomplete stored data, more critical applications may as well be unable to do so. However, the incompleteness aspect of managed data could only be revealed in the evaluations, and it is an important part of the more general consideration of data quality, which we briefly discuss in Section 6.2.2.

6.1.5 Analytical and Simulative Performance Evaluation

The algorithms proposed in this dissertation were extensively evaluated by means of analytical and simulative methodologies. A significant contribution to the research of analytical performance evaluation is the proposition of an analytical model that allows to analyze the communication cost of data-centric storage-related approaches. In contrast to existing ones, the proposed model significantly exceeds the detail at which the local routing overhead in the vicinity of storage nodes is modelled. The results obtained from the analytical evaluation of many approaches relating to our storage tier showed that the local communication overhead is a significant contribution to the overall overhead. While exploiting the ability of the analytical model to perform evaluations of system parameters on many orders of magnitude, we were able to identify the most appropriate domains of operation of our algorithms in terms of data and query dynamics. We found that the domains where the storage tier consistently outperforms the analyzed related approaches match well the design goals of this dissertation.

The analytical model was complemented with an extensive comparative simulation study of all proposed algorithms to understand the algorithms' performance also in terms of metrics other than the communication cost. The considered metrics describe the properties that are directly related to the efficiency, robustness, scalability, and accuracy requirements. In conjunction with the analytical results, the overall performance evaluation presents a sound picture of the performance of all framework tiers and confirms what has been claimed in the requirements. However, the simulations have also revealed the limits of any data management approach in mobile ad-hoc networks that are due to small node densities on one side and high network mobility on the other side. In such extreme situations, no sufficient connectivity can be achieved on the routing tier between distant nodes. This fact emphasizes to consider data management approaches that are supported by hybrid system structures (Section 6.2.4).

6.2 Promising Research Directions

Promising research directions that follow from the proposed location-centric storage framework are extensive. While there is potential for elaborations and optimizations of various details of the devised concepts, strategies, and algorithms, we outline in the following some of the broader research topics worth considering, without any claim of completeness.

6.2.1 Data Replication

Data replication is probably the most straightforward extension to the proposed framework's storage tier mechanisms. For its consideration, it helps looking at the two major purposes of replication in mobile ad-hoc networks. Firstly, replication increases data availability. This property is especially relevant in mobile ad-hoc networks due to the forming of network partitions (Section 6.1.1). For instance, nodes in different partitions are unable to share data that each of the nodes may need for query processing. Such problems can be improved by replicating data in strategic places to make it available in other network partitions. Our quantitative analysis of network partitioning already gives some hints on how suitable replica placement strategies could look like. For example, the number of partitions that can be expected in average is valuable information for deciding on a suitable number of replicas stored in the network. It will be beneficial to understand more generally how replica placement strategies can take advantage of information that is derived from observing a network's partitioning behavior.

Second, replication increases robustness. While the system model in this dissertation considers that nodes do not fail, taking into account and tolerating certain degrees of node failures requires replication so that data stored at failing storage nodes does not become permanently lost. Because node failures are just another unpredictable property of mobile ad-hoc networks, probabilistic approaches will most likely be the strategies of choice. How such approaches will be parameterized to predict just the right number of replicas for a certain degree of failure resilience is a challenging problem that is still open. Furthermore, it is conceivable to reconstruct dynamic data items even in the case of complete data losses. For which degrees of data dynamics reconstruction methods are suitable is just one other question to be answered.

When considering replication, questions follow on how to integrate data replication with the proposed migration mechanisms. As a starting point, some elementary mechanisms are already provided by data migration that can be used for replication purposes right away. For instance, the mechanisms allow to duplicate the copy of a data subset at another node and to merge two copies in the case of a partition join. This is a good basis for looking into a general storage approach that consistently combines migration and replication mechanisms.

From the introduction of replication, significant questions arise that deal with the consistency of the managed data and query results. In this dissertation, we have assumed only eventual consistency, which is tolerable for applications that do not rely on a strict chronological ordering of the data objects involved in the processing of the considered spatial query types. However, for some more critical applications, such ordering is likely to be more important. For instance, it might be required that the second of two consecutive nearest neighbor queries is processed based on object copies that are at least as recent as the object copies used in the first query. Related suitable consistency concepts that are also feasible to be implemented in mobile ad-hoc networks are certainly a research topic worth to be considered in more detail.

6.2.2 Extensions of Data and Model Characteristics

Extensions pertaining to the characteristics of the managed data and data model correspond to the design space discussed in Section 1.4 of this dissertation and comprise, among others, the dimensions of data dynamics, location model, and data quality.

With respect to the first extension, the following two details are possible candidates for further study. Firstly, it is yet to be understood more completely which storage approach is suitable along the transition from dynamic to virtually static data. Especially the latter class of data is very common, because it includes many types of geographical data, like city maps and building plans, which are relevant for location-based applications. The results of the analytical study (Section 6.1.5) are a good starting point and indicate some transitions where different storage approaches become more appropriate in relation to others. Second, this dissertation did not consider the relation between data dynamics and query dynamics. In the present state, all considered data items are updated to storage nodes, independent of how frequently they might be requested by spatial queries. There is clearly a large potential for adaptation, where the relative magnitudes of update and query rate may be used to adapt the strategy with which certain data items are managed. For instance, if a data item is queried only very rarely, it is possible to use a storage approach where that data item is stored locally at the node that created the item in the first place. The analytical evaluation also gives some valuable hints in this direction by its considering both a variation of the update and query rate.

Regarding location models, the proposed concepts in this dissertation build solely on geometric location models that are founded on Cartesian coordinates. However, depending on the environment where a mobile ad-hoc network forms, symbolic location models that describe locations by an expressive name (e.g. “Room 2.01”, “2nd Floor”) are more intuitive for users than their geometric counterpart. Because users are likely to move between domains that use either one of these model types, the combination of both into hybrid location models is also relevant. The fundamental question is how efficient, robust, and scalable data management structures can be implemented in mobile ad-hoc networks also in these hybrid models. In contrast to geometric location models, it is impossible to store data nearby a geometric reference coordinate, because an Euclidean metric is not immediately available. This is complicated by the fact that symbolic coordinates may cover geographic areas of different shape and extent, such as the rooms of a building in contrast to the radio cells of a mobile communication network. Extending the considerations to hybrid location models, the development of suitable management structures that are compatible with both geometric and symbolic model portions is an open problem that impacts all tiers of this dissertation’s data storage framework.

Regarding data quality, it was shown in this dissertation that both the inaccuracy and incompleteness of the data captured by a node’s embedded sensor lead to a deterioration in the quality of spatial query results. A systematic approach is needed to consider data quality-related questions in more general terms, specifically for the case of data management in mobile ad-hoc networks. A key question is how data and query results can be provided not only with best effort, but with defined quality characteristics. For that, suitable metrics that are able to quantify the specific quality dimension of data and query results are required in the first place. A first step in this direction may be to analyze how more fundamental characteristics of mobile ad-hoc networks, including coverage, local object density, and mobility, can be used to derive more abstract quality metrics. With the availability of suitable metrics, it is then possible to adapt mechanisms of the storage and service tier to provide data and query results with specific quality requirements that are given, e.g., by the user’s application. Because the updating of data is only necessary to the extent of the required data and query quality, but not more, it can be expected that there is also significant potential to apply adaptation techniques to further tune the performance of the data storage algorithms.

6.2.3 Extension of Service Functionality

In this dissertation, we have considered two types of spatial queries: range and k-nearest neighbor queries. It is straightforward to consider several other types of spatial queries, for instance, group and reverse nearest neighbor queries. The importance of such query types is supported by the large body of related literature, which considers implementations for processing these queries mainly based on centralized index structures (Section 2.5.3). However, it is unexplored how such queries can be processed in mobile ad-hoc networks in conjunction with probabilistic location semantics. Further generalizations of supported query types eventually lead to approaches where spatial or even more general queries are expressed by a dedicated query language. The design of a suitable query processing framework for mobile ad-hoc networks that supports such a general approach is a largely open research challenge.

While queries are synchronous operations that are immediately processed by the query system, the consideration of asynchronous services is also of interest. Spatial event management is one important instance of such services and describes, by means of event predicates, when objects in the real world assume certain spatial constellations. In contrast to query processing, event observation works entirely different and introduces many additional questions to the storage and service tier. For instance, the suitable strategy of how to observe the predicate of an event that triggers when two objects meet at a given distance depends on the current distance the two objects maintain to each other. For small distances, a single storage node may be able to observe the predicate alone because it stores information about both objects. With increasing distance, two objects are likely to be known to different servers and their information must be brought together in order to detect possible occurrences of the event. With further increasing object distance, also this approach becomes impractical due to updates that need to be propagated over many network hops. In this latter case, it might be more appropriate to make use of dead reckoning-based approaches. This example illustrates the bandwidth of possible approaches to observe just a single type of event under different object constellations. In mobile ad-hoc networks, the observation of events is a challenging and still open task.

In Section 6.2.1, data replication was proposed as an extension of the storage tier in order to increase the robustness of the data store. Due to the fact that event observation is stateful, it makes sense to consider the replication of the event observation functionality itself to make it more robust against node failures in wireless multihop networks. In the situation of an event occurring in the real world, it is then possible to observe the occurrence of the event even if a node that contains essential observation state has failed. While applying replication to event observation sounds promising, it is absolutely unclear how to guarantee that event observation is always consistent. For example, it is possible that based on inconsistent data, observations taking place in parallel at different servers lead to predicate evaluations that contradict each other. A framework of algorithms for the scalable and consistent observation of spatial events in mobile ad-hoc networks is a complex research field worth further consideration.

6.2.4 Hybrid System Structures

In this dissertation we have focused on the design and implementation of a framework for location-centric storage in mobile ad-hoc networks. In Section 6.1.3 and 6.1.5 we have concluded that the data management performance in pure mobile ad-hoc networks is limited by the physical constraints that the network may impose, such as limited node density and network connectivity. Therefore, the ultimate extension to the purely infrastructureless system model occurs along the system structure dimension in the design cube in Figure 1.5, Chapter 1. In a hybrid networked system, both ad-hoc and fixed network structures are combined to complement each other for the purpose of collaborative data management.

Figure 6.1 illustrates the layout of a conceivable hybrid system structure that is derived from a combination of our location-centric storage framework in Figure 2.22, Section 2.4, and the Nexus system architecture in Figure 1.4, Section 1.3.2. Along the figure's central vertical axis (dashed line), we have added three components: "Hybrid: Service I/O", "Hybrid: Data I/O", and "Assisted Ad-hoc". These components mediate on different functional levels between the infrastructure-based Nexus management facilities on one side, and the infrastructureless location-centric storage framework's components on the other side.

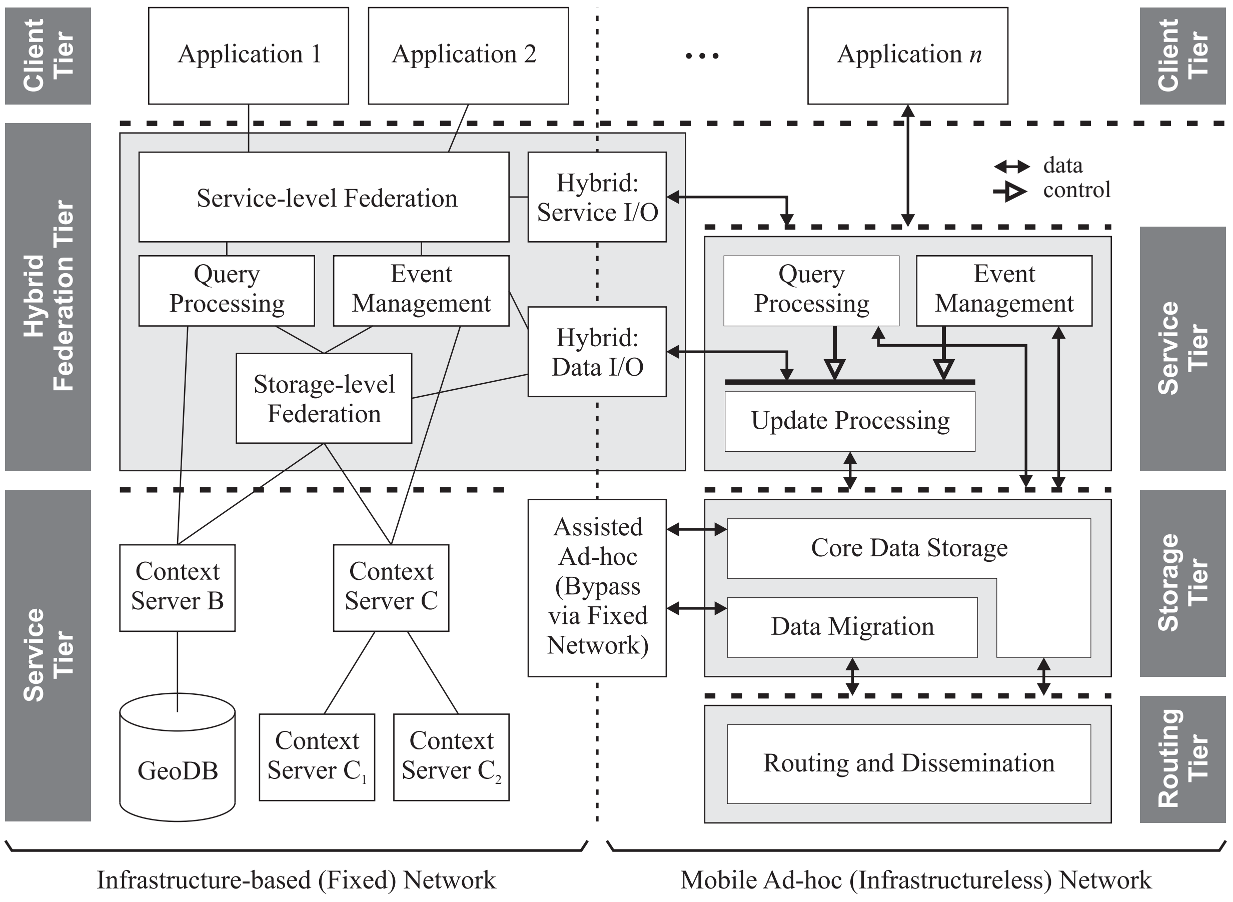


Figure 6.1: Hybrid system structure.

As a first step towards a hybrid system structure, we propose the introduction of an *assisted* ad-hoc mode. While in this mode data is retained for storage on the nodes in the mobile ad-hoc network, infrastructure-based communication links may be used to bypass, for instance, congested parts of the network or network partitions. If such bypass links are wisely used, we can expect to improve the performance of data storage in the mobile ad-hoc network considerably in situations of low node density and many network partitions. Clearly, this means that some mobile devices must support networking technology that allows to setup a wireless uplink to the infrastructure. Many handheld devices today indeed support different wireless communication technologies, as we have illustrated in our brief survey on technological trends in Section 1.2. This fact makes the assisted ad-hoc mode feasible from a technological point of view. However, many issues remain open on the algorithmic side. For instance, it is unclear how to locate suitable downlink nodes that are close to the geographic region to which some data or query is to be delivered. A first idea may be to randomly select a set of beaconing nodes in the ad-hoc network, which announce to the infrastructure their presence in a geometric region and optionally, additional information about their neighboring nodes.

In the hybrid management case, both infrastructureless and infrastructure-based network components are interconnected to collaboratively manage data and provide service functionality. Two levels can be considered at which integration may take place (Figure 6.1). "Hybrid: Data I/O" refers to all functions required for interconnecting both system structures on the level of data storage. For instance, a mobile ad-hoc network could be regarded as just another type of context server that is accessed via its update processing interface. A node in the mobile ad-hoc network may take the role of a federation proxy that mediates updates between the infrastructure and ad-hoc network in both directions. "Hybrid: Service I/O" refers to functionality that allows for collaboration on the level of services. For example, a spatial query may be split into partial queries that are independently processed in the infrastructure and ad-hoc network until their results are joined in the infrastructure again. The partial query processing mechanisms proposed in this dissertation may be a valuable starting point in this direction. Splitting techniques are also conceivable for event management. For example, the observation of the parts of a more complex event predicate (Section 6.2.3) may be delegated to either infrastructure-based or infrastructureless observation nodes. Finally, to provide a uniform view on the hybrid storage and service facility to location- and context-based applications, additional components for both storage- and service-level federation are likely to be required.

In conclusion, the open research challenges discussed in this section could only scratch the surface of the complexity and may raise even more questions towards hybrid data management systems. Until the fixed mobile convergence will have reached a point where the deployment of full-fledged context-based services would be technically feasible, many efforts to attain rich functionality, sound performance, and maximum user experience are yet required.

Appendix A

List of Abbreviations

Abbreviation	**Description**
1G	First Generation (in telecommunications)
2G	Second Generation (in telecommunications)
3G	Third Generation (in telecommunications)
4G	Fourth Generation (in telecommunications)
ACAN	Ad hoc Context Aware Network
ADV, adv	(server) advertisement
AFR	Adaptive Face Routing
ANVS	accumulative node visibility set
AODV	Ad hoc On-Demand Distance Vector (Routing)
AWML	Augmented World Modelling Language
AWQL	Augmented World Query Language
BPR	Bidirectional Perimeter Routing
BFS	Breadth First Search
CACK	consolidation acknowledgement
CAP	Consistency, Availability and Partition Tolerance
CASM	context-aware service middleware
CASP	Context-Aware Service Platform
CASS	Context-Awareness Sub-Structure
CEP	circular error probable
CHR	Cell Hash Routing
C-IUQ	Constrained Imprecise Location-dependent Range Query over Uncertain Objects
C-LCS	canonical location-centric storage
CLDP	Cross-Link Detection Protocol
CNN(Q)	constrained nearest neighbor (query)
CNVS	continuous node visibility set
Context CF	context component framework
C-PART	cell-based partitioning
C-PNN	Constrained Probabilistic Nearest Neighbor Query

Abbreviation	**Description**
CPU	Central Processing Unit
C-REP	cell-based replication
CREQ	consolidation request
C-SC	cell-based single-copy
CSMA/CA	Carrier Sense Multiple Access/Collision Avoidance
DACK	data acknowledgement
DataMiP	data migration protocol
DCS	data-centric storage
DCS/GPSR	data-centric storage via Greedy Perimeter Stateless Routing
DHT	distributed hash table
DIFS	Distributed Index for Features in Sensor Networks
DIKNN	Density-aware Itinerary KNN query processing
DIM	Distributed Index for Multi-Dimensional Range Queries
ENNQ	(Entity-based) Probabilistic Nearest Neighbor Query
ERQ	(Entity-based) Probabilistic Range Query
FAR	Face-aware Routing
FNNQ	federated k-nearest neighbor query
FP	funding period
GB	gigabyte
Gbit/s	gigabit per second
GCLP	Geographic Content Location Protocol
GDSTR	Greedy Distributed Spanning Tree Routing
GeoDB	Geographic Database
GJU	Galileo Joint Undertaking
GLS	Geographic Location Service
GNN(Q)	group nearest neighbor (query)
GOAFR	Greedy Other Adaptive Face Routing (“gopher”)
G-PART	global partitioning
GPS	Global Positioning System
GPSR	Greedy Perimeter Stateless Routing
G-REP	global replication
G-SC	global single-copy
GSM	Global System for Mobile Communications
HLS	Hierarchical Location Service
IBM	International Business Machines Corporation
IEEE	Institute of Electrical and Electronics Engineers
I/O	input/output
IUQ	Imprecise Location-dependent Range Query over Uncertain Objects
kB	kilobyte
kB/s	kilobyte per second

Abbreviation	**Description**
kbit/s	kilobit per second
k-NNMP	k nearest neighbors for moving query point
k-NN(Q)	k-nearest neighbor (query)
LCS	location-centric storage
LLS	Locality-aware Location Service
MAC	medium access control
MANET	mobile ad-hoc network
MB	megabyte
Mbit/s	megabit per second
MDM	mobile data management
MDP	migration decision policy
MRP	migration recommendation policy
NLLS	nonlinear least-square
NN(Q)	nearest neighbor (query)
NTE	network topology exploration
OD-GPSR	On-Demand Greedy Perimeter Stateless Routing
PC	personal computer
PDA	personal digital assistant
pdf	probability density function
PGNN	Probabilistic Group Nearest Neighbor Query
PLS	Prediction Location Service
PNNQ	probabilistic Nearest Neighbor Query
PNQ	probabilistic k-nearest neighbor query
PRP	Perimeter Refresh Protocol
PRQ	probabilistic range query
PTRQ	Probabilistic Threshold Range Query
QoS	Quality of Service
RAM	Random Access Memory
R-DCS	Resilient Data-centric Storage
RED	redundancy notification
RFID	Radio Frequency Identification Tag
RLS	Randomized Location Service
RNN(Q)	reverse nearest neighbor (query)
RQ	range query
RR	Rendezvous Regions
RTS/CTS	Request to Send/Clear to Send
SFB	Sonderforschungsbereich (German)
SIG	Special Interest Group
SOCAM	Service-Oriented Context-Aware Middleware
SR-DCS	Structured Replication-Data-centric Storage
STQP	spatiotemporal query processing
STVN	Segment-Tree Virtual Network

Abbreviation	**Description**
TCP	Transmission Control Protocol
TTL	time-to-live
UMTS	Universal Mobile Telecommunications System
VANET	vehicular ad-hoc network
VDPS	Virtual Home Region-based Distributed Position Service
VPCR	Virtual Polar Coordinate Routing
VPCS	Virtual Polar Coordinate Space
WGS84	World Geodetic System 1984
WiFi	Wireless Fidelity
WiMAX	Worldwide Interoperability for Microwave Access
WLAN	wireless local area network
WMN	wireless mesh network
WSN	wireless sensor network

Appendix B

Network Partitioning: Addendum

In this appendix we provide additional results for the simulation study of network partitioning in Section 2.2. In the following, we examine the impact of different values for the transmission range r_{tx} on each of the partition metrics for the random waypoint mobility model. In qualitative terms, the results show similar behavior independent of the transmission range. However, in quantitative terms, the results of most of the metrics vary significantly across several orders of magnitude, even for the variation of r_{tx} being only linear. This general observation has a strong influence on the performance of many kinds of algorithms.

Figure B.1 and B.2 show the results for the average number $|\Pi|_{\text{avg}}(T)$ and average size $S_{\text{avg}}(T)$ of partitions, respectively, as a function of the number of nodes in the network. For $r_{\text{tx}} = 250$ m, the largest considered transmission range in our experiments, $|\Pi|_{\text{avg}}(T)$ decreases sharply and is below 2 for networks with more than 200 nodes inside a 4 km^2 area. As a consequence, $S_{\text{avg}}(T)$ increases almost linearly for more than 200 nodes. On the other side of the spectrum, i.e. for $r_{\text{tx}} = 50$ m, $|\Pi|_{\text{avg}}(T)$ is virtually equal to the number of nodes in the network for about up to 100 nodes and remains above 100 even for as many as 2400 nodes. This observation directly relates to $S_{\text{avg}}(T)$, which grows slower for smaller transmission ranges. This large number of partitions for an already large network with 600 nodes/km^2 and five times the transmission range of standard Bluetooth seems considerably high.

Figure B.3 and B.4 show the results for the average partition change rate $R_{\text{part}}(T)$ and the average node partition change rate $R_{\text{part}}(n_k, T)$, respectively, as a function of the number of nodes. The results of $R_{\text{part}}(T)$ show a large discrepancy between transmission ranges. For example, networks with 1000 nodes experience one partition change on average every 1000 seconds for a transmission range of 250 m. On the other hand, the same number of nodes in a network with 50 m transmission range is affected by more than 10 changes per second. This is a difference of 4 orders of magnitude for the rate while using a transmission range that is only five times higher. The average partition change rates from the perspective of a given node, $R_{\text{part}}(n_k, T)$, are significantly smaller for both, low transmission ranges over the complete spectrum of the number of nodes and the higher transmission ranges with low node count. This matches the observation that in these cases many small partitions exist where only a small number of nodes is affected by each change at a time. Whenever the number of partitions is low, the average node partition change rate comes close to the network-wide rate.

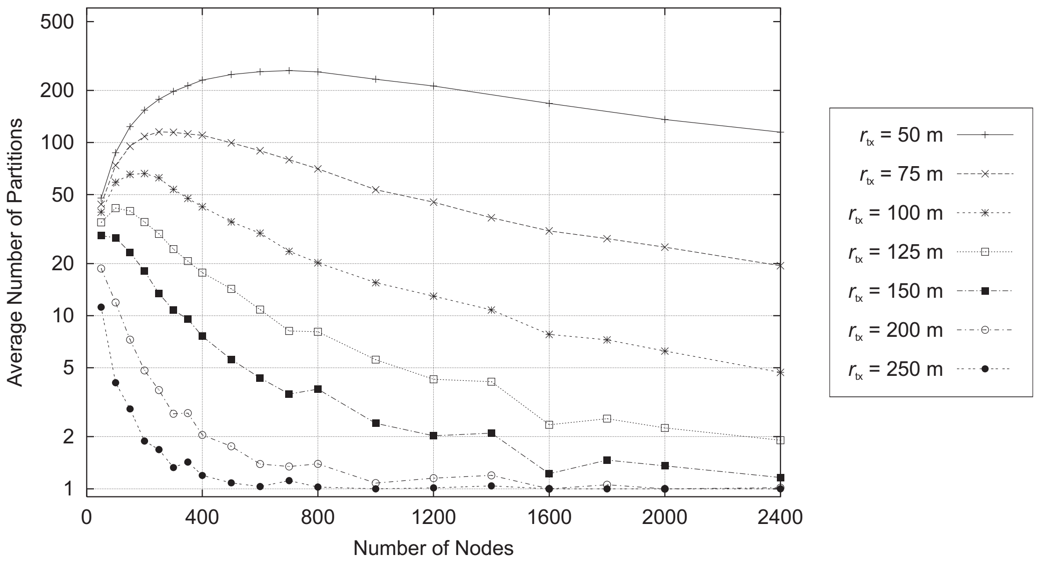


Figure B.1: Average number of partitions $|\Pi|_{\text{avg}}(T)$ as a function of the number of nodes n for different transmission ranges r_{tx} and the random waypoint mobility model.

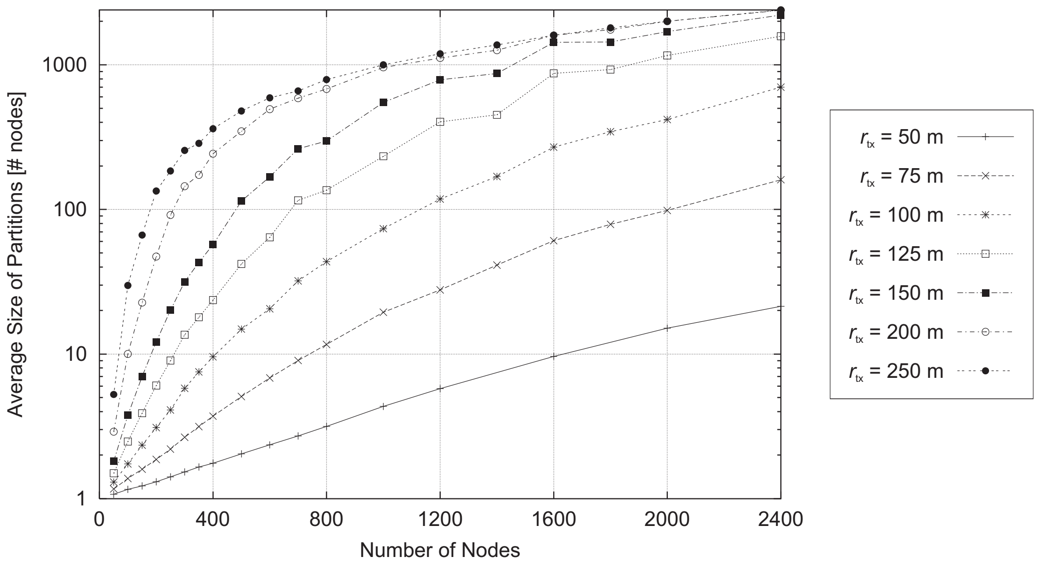


Figure B.2: Average size of partitions $S_{\text{avg}}(T)$ as a function of the number of nodes n for different transmission ranges r_{tx} and the random waypoint mobility model.

The results for the average partition size ratio as a function of the number of nodes shown in Figure B.5 indicates that for very different transmission ranges, between 100 m and 250 m, the partition size ratio converges towards 1 very quickly as the number of nodes increases in the network. This means that typically, the sizes of two partitions being joined or split is very uneven and does not depend on r_{tx}. However, Figure B.1 shows that the average number of partitions in our experiments is very different. For example, in that figure, it ranges from 1 to 20 partitions at 800 nodes for transmission ranges of 100 m to 250 m, while the partition size ratio in Figure B.5 is larger than 0.9 in all of these cases.

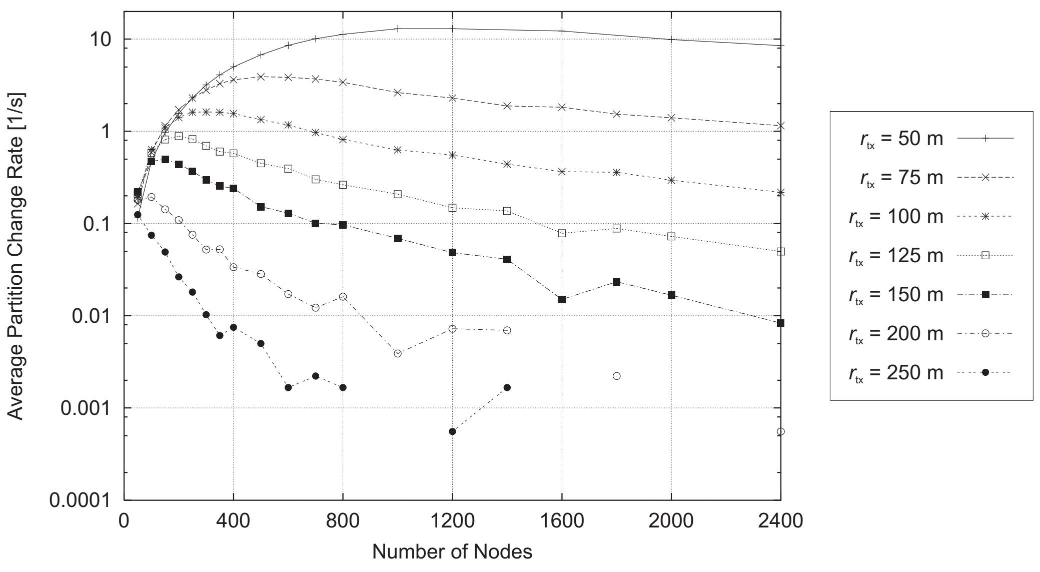


Figure B.3: Average partition change rate $R_{\text{part}}(T)$ as a function of the number of nodes n for different transmission ranges r_{tx} and the random waypoint mobility model. Missing points for $r_{\text{tx}} = 200$ and 250 m indicate that $R_{\text{part}}(T) = 0$.

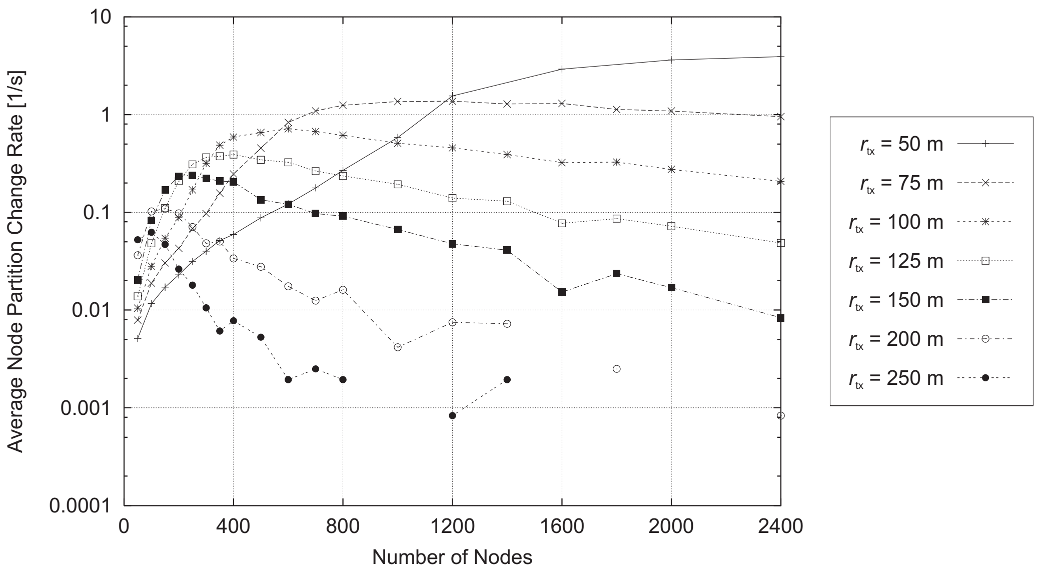


Figure B.4: Average node partition change rate $R_{\text{navg}}(T)$ as a function of the number of nodes n for different transmission ranges r_{tx} and the random waypoint mobility model. Missing points for $r_{\text{tx}} = 200$ and 250 m indicate that $R_{\text{part}}(T) = 0$.

The average node separation times for pairs of nodes shown in Figure B.6 decrease sharply for different transmission ranges. For networks with a transmission range of 75 m and 800 nodes, the average separation time of node pairs is almost half of the overall simulation time. For networks with 250 m transmission range the average separation time converges towards zero quickly as the number of nodes in the network increases. Observe that for small transmission ranges of 125 m and less, the average separation time remains at almost its maximum and starts decreasing with a larger number of nodes for smaller transmission ranges.

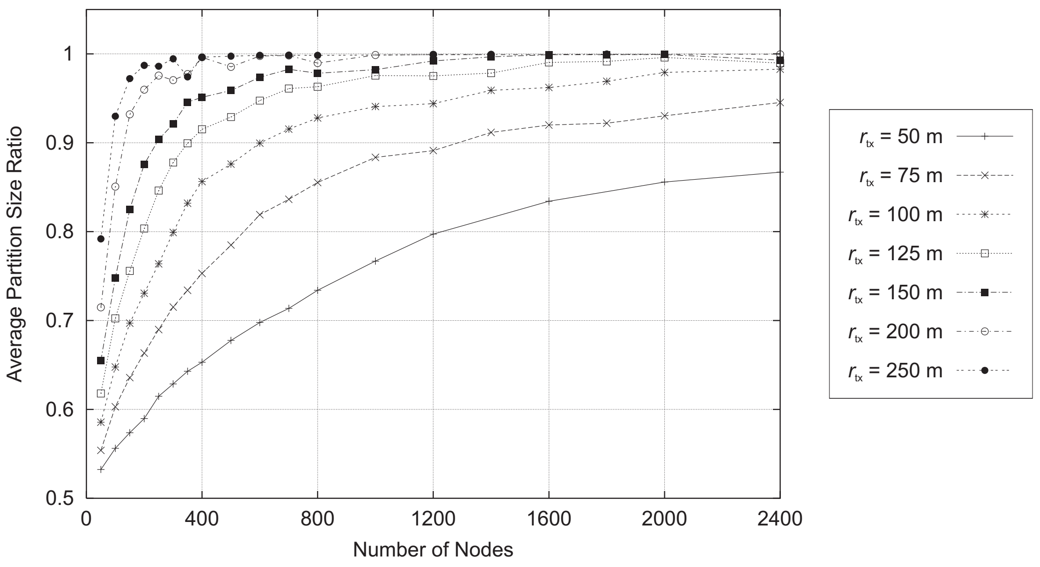


Figure B.5: Average partition size ratio $\mathrm{PSR}_{\mathrm{avg}}(T)$ as a function of the number of nodes n for different transmission ranges r_{tx} and the random waypoint mobility model.

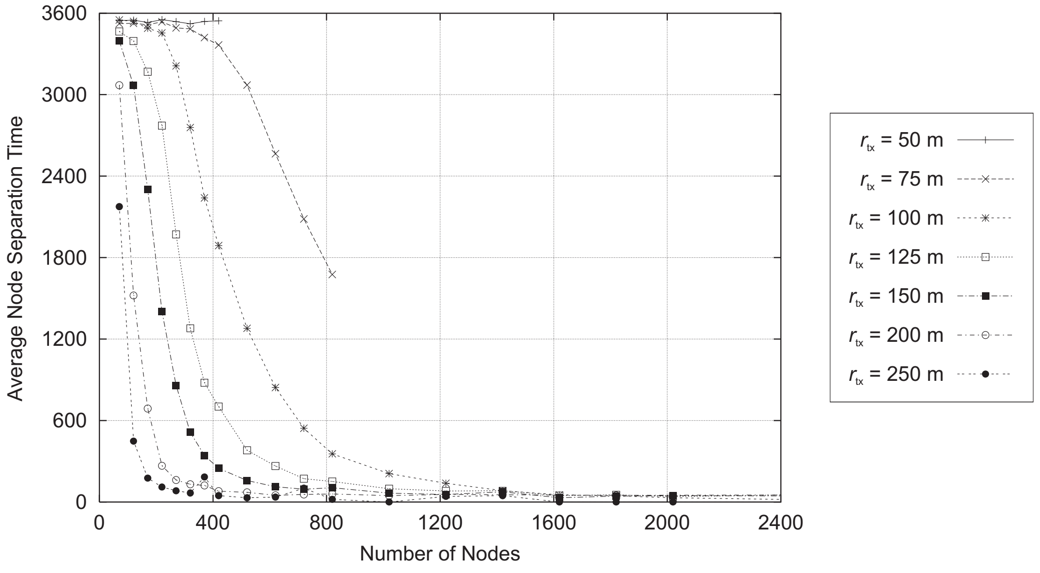


Figure B.6: Average over the sum of node separation times $\tau_{\mathrm{sep}}(n_1, n_2, T)$ for $n/2$ disjoint pairs of nodes (n_1, n_2) as a function of the number of nodes n for different transmission ranges r_{tx} and the random waypoint mobility model.

Figure B.7 shows the results for the average size of the continuous node visibility set $N_{\mathrm{cont}}(n_k, T)$ over time for 400 nodes. The gradient of the individual functions describes how fast the group of nodes found permanently in the same partition decays. The general observation for all transmission ranges is the extreme slope at the beginning of the considered time interval. This is an artifact of the random waypoint mobility model, where small groups of nodes break apart from the main group of nodes in the center. At first, the average is very high, because all individual visibility sets of each node are the size of the single large partition. When small groups of nodes are separated from the large partition, the value of $N_{\mathrm{cont}}(n_k, T)$ for the nodes

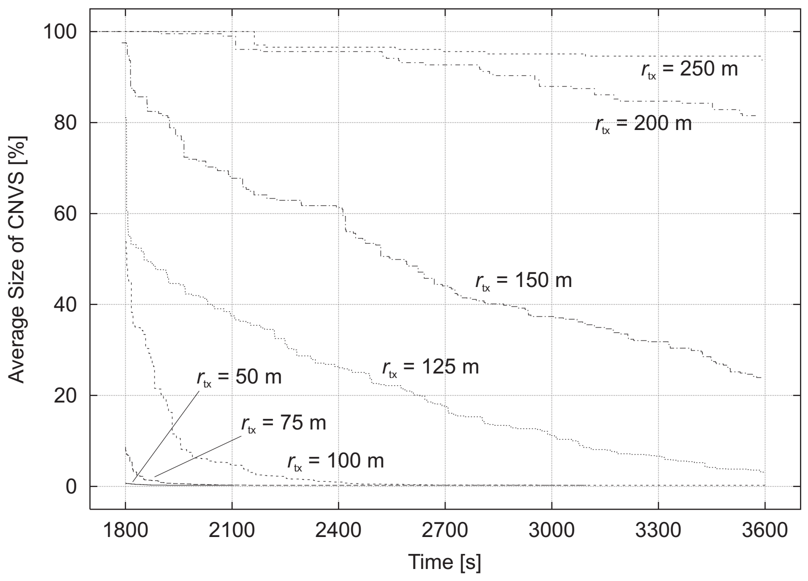


Figure B.7: Average size of n continuous node visibility sets $N_{\text{cont}}(n_k, T)$ as a function of T for $n = 400$ nodes, different transmission ranges r_{tx} and the random waypoint mobility model.

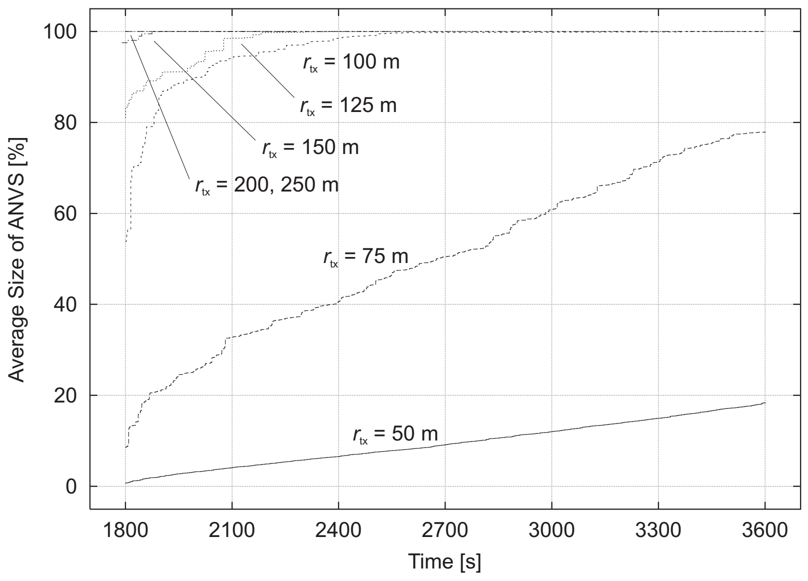


Figure B.8: Average size of n accumulative node visibility sets $N_{\text{acc}}(n_k, T)$ as a function of T for $n = 400$ nodes, different transmission ranges r_{tx} and the random waypoint mobility model.

of these groups drops to a very small number, which strongly pushes the average size of the node visibility set towards smaller values. At later times, many small visibility sets already exist, and additional separations of nodes have little influence on the average size of the metric. The steep gradient of the continuous node visibility set for the lower transmission ranges indicates that reasonably stable groups of nodes are small and can be assumed to be available only for small time spans of 100 seconds or less. For higher transmission ranges the speed of decay slows down significantly and the size of the stable groups grows.

Figure B.8 shows the average size of the accumulative node visibility set over time for networks with 400 nodes at different transmission ranges. Obviously, the accumulative node visibility set grows faster for larger transmission ranges. For $r_{\mathrm{tx}} = 50$ m, the results show that even after 30 minutes of simulation time, only 20 percent of the nodes have been contained in a node's same partition. This is due to the small average number of partitions (Figure B.1). While moving, nodes encounter only a small number of other nodes in the same partition. Because the partitions are small with respect to their geographical extent, they provide a small contact area, so that the fusion of individual partitions occurs only rarely. Also note the strong initial growth for higher transmission ranges. The reason is similar to the $N_{\mathrm{cont}}(n_k, T)$ case in Figure B.7. At the point of initialization of the accumulative node visibility set, a few isolated nodes or very small groups of nodes exist. At the same time, one large partition exists, in which all nodes have a large accumulative node visibility set consisting of the nodes in that partition. The average size is relatively small because of the small visibility sets. After the addition of the small groups of nodes to the large partition, their individual accumulative visibility sets grow rapidly, resulting in a large increase of the average during the first seconds. These phenomena are observed independent of a variation in the transmission range (Figure B.7) or in the number of nodes for a given transmission range (Figure 2.21 in Section 2.2.5).

Appendix C

Preliminaries

C.1 LCS Core Mechanism

This section provides additional preliminary results, which are used in the analytical study of the core data storage approach in Section 3.4.

C.1.1 Correlation between Topology and Geometry

Section 3.4.3 makes use of functions $H_1, \ldots, H_8$ in the derivation of the communication cost of each discussed approach. These functions capture basic cost terms using different configurations of geometric shapes and communication patterns. Figure C.1 to C.4 show the graphs of each of these functions, computed to an error of below one percent. Each graph gives the mean route length in number of hops. For any of the H_i, a base area of $1200 \cdot 1200$ m^2 is assumed, with 600 nodes placed randomly according to a uniform distribution.

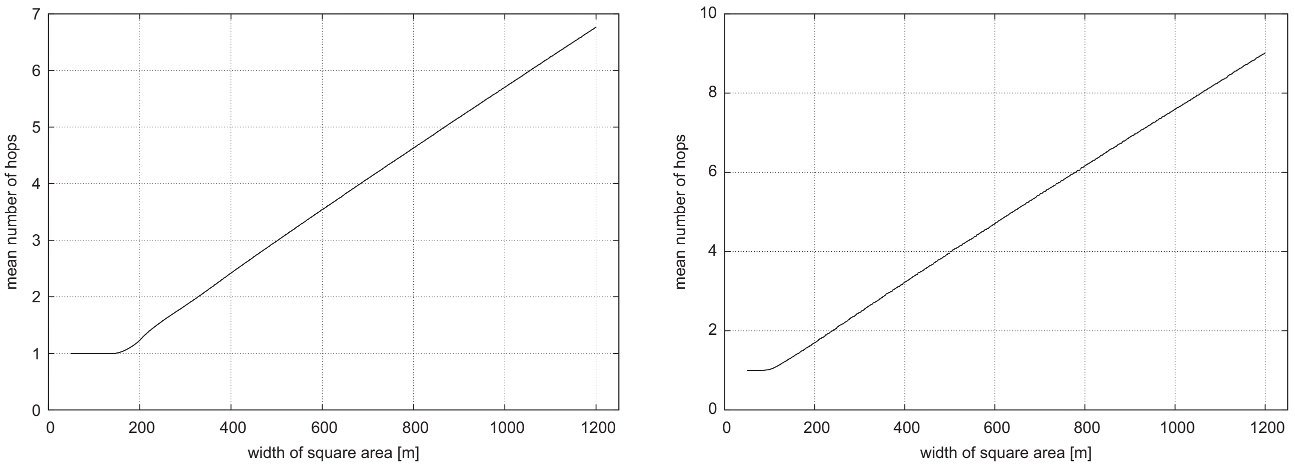
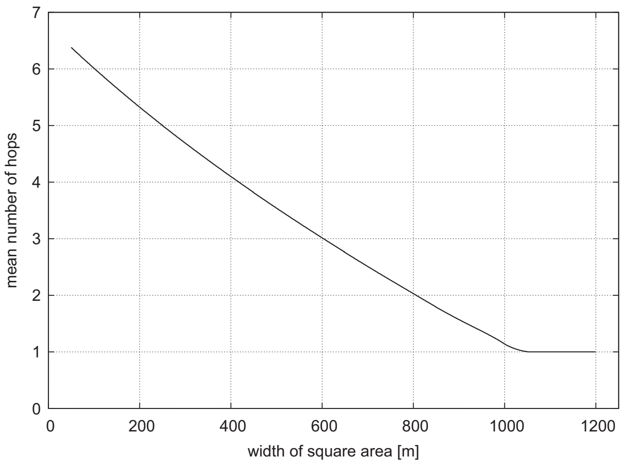


Figure C.1: Left: Mean route length $H_1(a)$ between node in square $A = a^2$ and node at center of A. Right: Mean route length $H_2(a)$ between any two nodes in A.



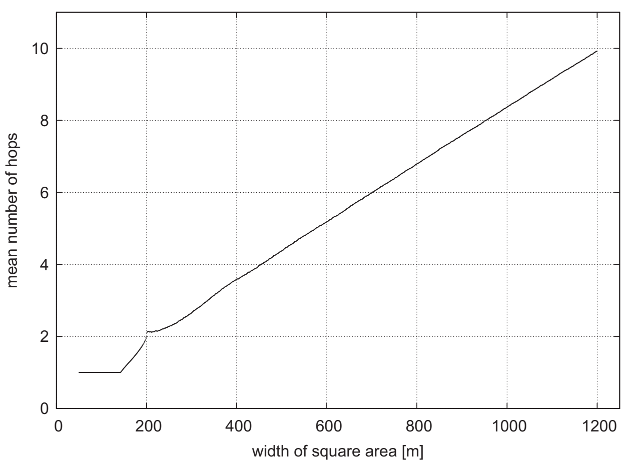
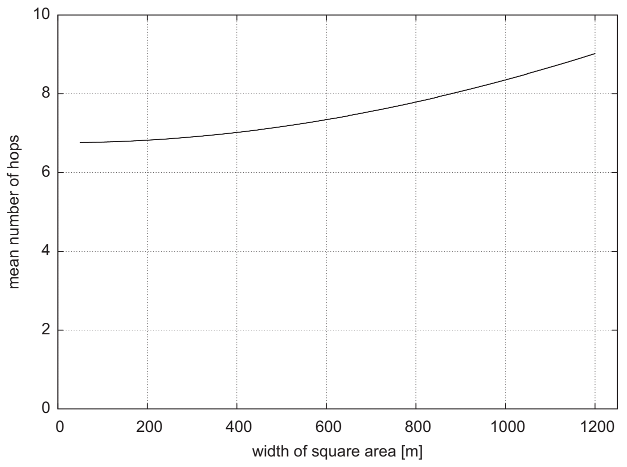


Figure C.2: Left: Mean route length $H_3(a)$ between node located outside of $A = a^2$ to any node inside of A, and vice versa. Right: Mean route length $H_4(a)$ between node located on the border of A to node located at the center of A.



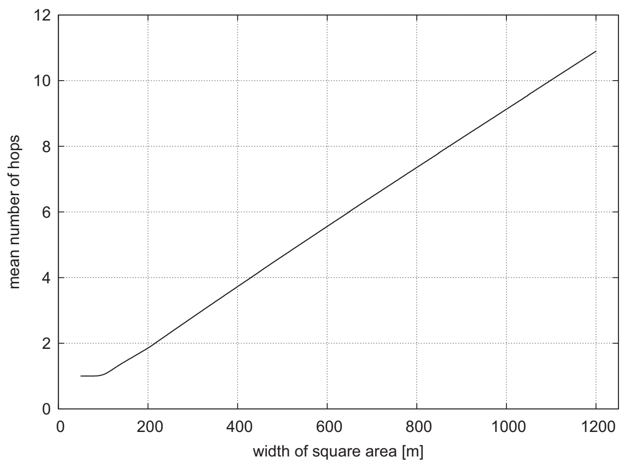
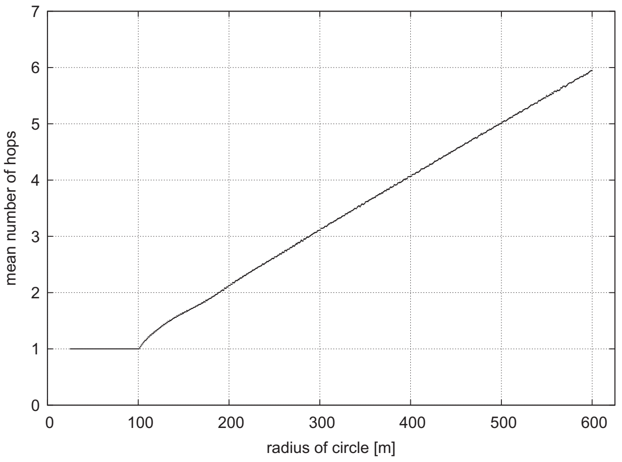


Figure C.3: Left: Mean route length $H_5(s)$ between a node located in region $A = a^2$ and node located in $S = s^2$, centered in A. Right: Mean route length $H_6(a)$ between a node located in the outer stripe of square $A = a^2$ with width $r_{\text{tx}}/2$ and any other node inside of A.

C.1.2 Derivation of Traversal Distances

In the cell-based approaches of the analysis, we require some preliminary results that are related to the time a node requires to traverse a region or to reach the edge of a region. The basic integrals are based on the work of [Dun97].

The first magnitude is the mean distance of an object travelling on a straight line inside of a cell. This distance can be determined by quantifying the number of cells that are traversed in horizontal and vertical direction. We assume that objects travel in straight lines, and that every direction of travel is chosen with equal probability. Figure C.5 contains the necessary symbols that we use in the following calculations.



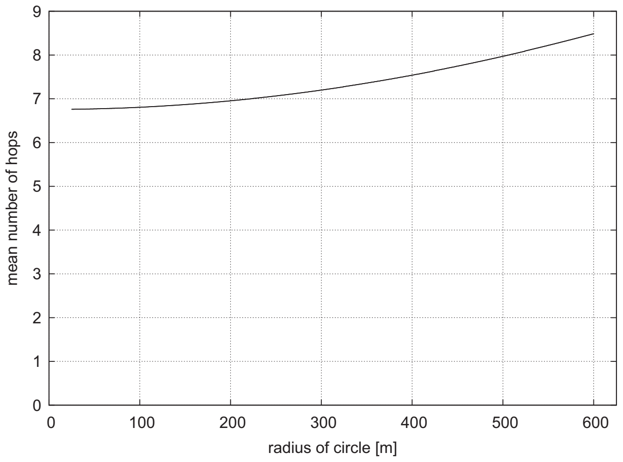
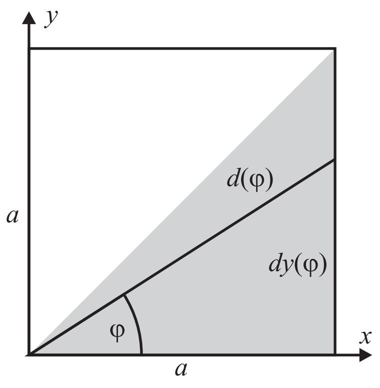
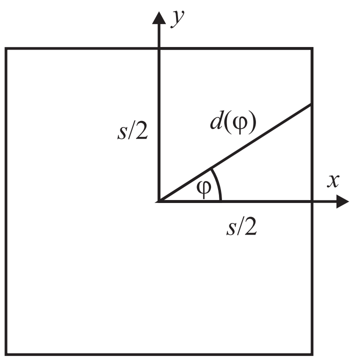


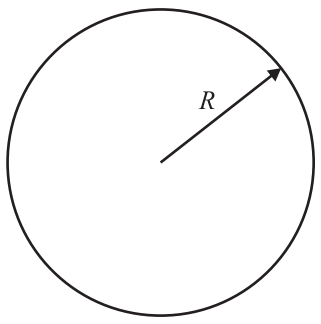
Figure C.4: Left: Mean route length $H_7(r)$ between a node inside of circle with radius r to the center of that circle. Right: Mean route length $H_8(r)$ between a node located in region A and a node located in the circle with radius r, centered in A.



a. Computations on square A



b. Computations on square S



c. Computations on circle C

Figure C.5: Derivation of Square/Circle Traversal Distances

We will compute the number of cells that are traversed horizontally and vertically when travelling distance $d(\varphi)$. Due to symmetry, we only have to consider the grey-shaded region indicated in Figure C.5.a. It holds that $d(\varphi) = s/\cos\varphi$, and $dy(\varphi) = s\tan\varphi$.

The number of cells that are traversed in horizontal and vertical direction is given by

$$n_s = 1 + \frac{dy(\varphi)}{s} \tag{C.1}$$

since the cell is completely traversed in horizontal, and only partially in vertical direction. For a given angle φ, the mean distance that needs to be travelled until a cell is entered in either horizontal or vertical direction is

$$\frac{d(\varphi)}{n_s} \tag{C.2}$$

We then determine the mean of such distances for all angles $0 \le \varphi \le \pi/4$:

$$\bar{l}_1 = \frac{4}{\pi} \int_{\varphi=0}^{\frac{\pi}{4}} \frac{d(\varphi)}{n_s} d\varphi = \frac{4}{\pi} \int_{\varphi=0}^{\frac{\pi}{4}} \frac{\frac{a}{\cos\varphi}}{1 + \frac{a \tan\varphi}{s}} d\varphi = \frac{4s}{\pi} \int_{\varphi=0}^{\frac{\pi}{4}} \frac{1}{\cos\varphi + \sin\varphi} d\varphi \tag{C.3}$$

In order to integrate this intermediate result, we make use of the following trigonometric identity about linear combination of the sine:

$$a \sin\alpha + b\cos\alpha = \sqrt{a^2 + b^2} \cdot \sin\left(\alpha + \arctan\frac{b}{a}\right) \tag{C.4}$$

Applying this identity yields

$$\bar{l}_1 = \frac{4s}{\pi} \int_{\varphi=0}^{\frac{\pi}{4}} \frac{1}{\sqrt{2}\sin\left(\varphi + \frac{\pi}{4}\right)} d\varphi \tag{C.5}$$

Because the argument of the sine contains only the variable of integration and a constant value, it can immediately be integrated using the following identity:

$$\int \csc x dx = \ln|\csc x - \cot x| + C \tag{C.6}$$

The result after integration is

$$\bar{l}_1 = \frac{2\sqrt{2}s}{\pi} \left[\ln \left| \frac{1}{\sin\left(\varphi + \frac{\pi}{4}\right)} - \frac{1}{\tan\left(\varphi + \frac{\pi}{4}\right)} \right| \right]_{\varphi=0}^{\frac{\pi}{4}} \tag{C.7}$$

Evaluating and simplifying to the end yields

$$\boxed{\bar{l}_1 = -\frac{2\sqrt{2}s}{\pi} \ln\left(\sqrt{2} - 1\right)} \tag{C.8}$$

The second result we require is the mean distance an object travels from the center of a square of length s to its border. Again, every direction is assumed equally probable. Symmetry properties of the square allow us to integrate over an angle of only $\varphi/4$. It holds that $d(\varphi) = (s/2)/\cos\varphi$ (Figure C.5.b). We can then set up the following integral and simplify:

$$\bar{l}_2 = \frac{4}{\pi} \int_{\varphi=0}^{\frac{\pi}{4}} d(\varphi) d\varphi = \frac{4}{\pi} \int_{\varphi=0}^{\frac{\pi}{4}} \frac{s}{2\cos\varphi} d\varphi = \frac{2s}{\pi} \int_{\varphi=0}^{\frac{\pi}{4}} \frac{1}{\cos\varphi} d\varphi \tag{C.9}$$

This time we can use the following integration rule:

$$\int \sec x dx = \ln|\sec x + \tan x| + C \tag{C.10}$$

Integrating by this rule yields

$$\bar{l}_2 = \frac{2s}{\pi}\left[\ln\left|\frac{1}{\cos\varphi} + \tan\varphi\right|\right]_{\varphi=0}^{\frac{\pi}{4}} \tag{C.11}$$

After simplifying we have the final result

$$\boxed{\bar{l}_2 = \frac{2s}{\pi}\ln(\sqrt{2}+1)} \tag{C.12}$$

In the case of a circle, shown in Figure C.5.c, the distance required from the center of the circle C to the border is always the circle's radius, independent of direction. Thus:

$$\boxed{\bar{l}_3 = R} \tag{C.13}$$

C.2 Derivation of the Location PDF

In this section we derive the closed form of the bivariate location probability density function (location pdf), which we use in (5.5) of Section 5.1.

We first restate the full equation in integral form, based on (5.1) and (5.2):

$$P_r = \frac{1}{2\pi\sqrt{|\Sigma|}} \iint_{(x-\mu_x)^2+(y-\mu_y)^2\leq r^2} e^{-\frac{1}{2}(X-M)^T\Sigma^{-1}(X-M)}dxdy \tag{C.14}$$

In the previous equation, we have used $M = (\mu_x, \mu_y)^T$ to denote the center of the circle, and Σ to denote the covariance matrix.

The random variables x and y are independent, and we assume that they are equal for both the x and y dimension, which is a reasonable assumption in case of a positioning system that is approximately isotropic with respect to position inaccuracy. Thus, we set $\sigma = \sigma_x = \sigma_y$, and the covariance matrix assumes the form $\sigma^2 E$ where E denotes the unit matrix. Thus, the inverse of Σ is $\Sigma^{-1} = E$ as well. Further, the determinant is $|\Sigma| = \sigma^4$. We can now simplify the argument of the exponential function:

$$\begin{aligned}
& -\frac{1}{2}(X-M)^T\Sigma^{-1}(X-M) && \text{(C.15)}\\
= \; & -\frac{1}{2\sigma^2}(x-\mu_x, y-\mu_y)\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\begin{pmatrix}x-\mu_x\\ y-\mu_y\end{pmatrix} && \text{(C.16)}\\
= \; & -\frac{1}{2\sigma^2}\left[(x-\mu_x)^2+(y-\mu_y)^2\right] && \text{(C.17)}
\end{aligned}$$

We plug this result and the determinant into (C.14) and we have

$$P_r = \frac{1}{2\pi\sigma^2} \iint_{(x-\mu_x)^2+(y-\mu_y)^2\leq r^2} e^{-\frac{1}{2\sigma^2}[(x-\mu_x)^2+(y-\mu_y)^2]}dxdy \tag{C.18}$$

Next, we substitute $x' = x - \mu_x$, which gives $dx' = dx$ and likewise, $y' = y - \mu_y$, which gives $dy' = dy$. Then

$$P_r = \frac{1}{2\pi\sigma^2} \iint_{x'^2+y'^2 \leq r} e^{-\frac{1}{2\sigma^2}(x'^2+y'^2)} dx'dy' \tag{C.19}$$

We now substitute polar coordinates, using the identity

$$\iint_C f(x,y)dxdy = \int_{\varphi'=0}^{\varphi'=2\pi} \int_{r'=0}^{r'=r} f(r\cos\varphi, r\sin\varphi) r dr d\varphi \tag{C.20}$$

Applying the substitution yields

$$P_r = \frac{1}{2\pi\sigma^2} \int_{\varphi'=0}^{\varphi'=2\pi} \int_{r'=0}^{r'=r} r' e^{-\frac{1}{2\sigma^2}r'^2} dr' d\varphi' \tag{C.21}$$

We now use the integral identity

$$\int x e^{ax^2} dx = \frac{1}{2a} e^{ax^2} + C \tag{C.22}$$

with $a = -\frac{1}{2\sigma^2}$ to solve the integral in (C.21) by first integrating via r':

$$\begin{aligned} P_r &= \frac{1}{2\pi\sigma^2} \int_{\varphi'=0}^{\varphi'=2\pi} \left[-\sigma^2 e^{-\frac{1}{2\sigma^2}r'^2} \right]_{r'=0}^{r'=r} d\varphi' && \text{(C.23)} \\ &= -\frac{1}{2\pi} \int_{\varphi'=0}^{\varphi'=2\pi} \left(e^{-\frac{1}{2\sigma^2}r^2} - 1 \right) d\varphi' && \text{(C.24)} \end{aligned}$$

Integration by φ' yields

$$\begin{aligned} P_r &= -\frac{1}{2\pi} \left[\varphi \left(e^{-\frac{1}{2\sigma^2}r^2} - 1 \right) \right]_{\varphi'=0}^{\varphi'=2\pi} && \text{(C.25)} \\ &= -\left(e^{-\frac{1}{\sigma^2}r^2} - 1 \right) && \text{(C.26)} \end{aligned}$$

Hence, the final result is

$$\boxed{P_r = 1 - e^{-\frac{1}{\sigma^2}r^2}} \tag{C.27}$$

Appendix D

Network Topology Exploration Region

This appendix provides the step-by-step derivation of the network topology exploration region (NTE region) that was introduced in Section 4.3.

The starting point are the Requirement (1) to (3), stated in Section 4.3. The following basic notations are assumed: By $\mathbf{A}_{\text{NTE}}$ and $A_{\text{NTE}} = |\mathbf{A}_{\text{NTE}}|$ we denote the NTE region and its corresponding area, respectively. The reference coordinate is denoted by $\mathbf{c}_r$. Let $\mathbf{r}_0$ denote the current position of data server u_0, and $d = |\mathbf{r}_0 - \mathbf{c}_r|$ the distance between $\mathbf{r}_0$ and $\mathbf{c}_r$.

D.1 Area Restriction on the NTE Region

The first objective is to satisfy Requirement (1), which dictates a circular region around position $\mathbf{r}_0$ of the current server u_0. This region must always be included in the NTE region, which must therefore satisfy the following minimum area:

$$A'_{\text{NTE}} = \pi \cdot R_0^2 \tag{D.1}$$

Parameter R_0 can be understood as a real multiple of the nominal transmission range r_{tx} (Section 3.1), that is, $R_0 = \nu \cdot r_{\text{tx}}$ for some $\nu \geq 0$.

The second objective is to satisfy Requirement (2), stating that the NTE region must obey an upper limit, independent of its shape. For that, we let us guide by the spatial domain of the migration recommendation policy (Section 4.2) and Requirement (3). First of all, the threshold distance d_{thresh} determines the outcome of the spatial predicate of the MDP (Equation (4.1)). Thus, nodes inside of the circle defined by d_{thresh} and $\mathbf{c}_r$ are potentially eligible and should be considered, independent of the server's current position $\mathbf{r}_0$ (note that the overall migration recommendation predicate may be true, but the spatial predicate may not).

If the server is located further away than d_{thresh}, any node closer to $\mathbf{c}_r$ than the server is more eligible when only considering the geometric distance. In this case, $\mathbf{r}_0$ and $\mathbf{c}_r$ define the shape of the NTE region. However, when the server is located beyond the critical distance d_{crit}, region A_{NTE} must be restricted in order to prevent the forming of an increasingly large scope. This restriction shall be defined based on the circle that is formed around $\mathbf{c}_r$ with radius d_{crit}.

Based on the aforementioned reasoning, the NTE region's area must satisfy:

$$A''_{\mathrm{NTE}} = \begin{cases} \pi \cdot d_{\mathrm{thresh}}^2 & \text{for } d \leq d_{\mathrm{thresh}} \\ \pi \cdot d^2 & \text{for } d_{\mathrm{thresh}} < d < d_{\mathrm{crit}} \\ \pi \cdot d_{\mathrm{crit}}^2 & \text{for } d \geq d_{\mathrm{crit}} \end{cases} \tag{D.2}$$

In the following, we set $R = \min\{\max\{d, d_{\mathrm{thresh}}\}, d_{\mathrm{crit}}\}$, which implies $R \in [d_{\mathrm{thresh}}, d_{\mathrm{crit}}]$. Combining (D.1) and (D.2), the NTE region is fixed to the following magnitude:

$$A_{\mathrm{NTE}} = \max\{A'_{\mathrm{NTE}}, A''_{\mathrm{NTE}}\} \tag{D.3}$$

D.2 NTE Region Base Shapes

We first derive a number of base shapes that we require in Section D.3 to define the precise shapes of possible NTE regions. In the following discussions, we assume that a coordinate transformation was performed in a Cartesian coordinate system, such that $\mathbf{c}_r$ is located at the origin $(0, 0)$ and $\mathbf{r}_0$ is located at $(-d, 0)$.

D.2.1 Special Case

We make the following distinction based on the relative magnitudes of R_0 and R. If $R_0 \geq R$, the only design option is to take a circular shape around $\mathbf{r}_0$ with radius R_0 (special case, Figure D.1.a). In all other cases where $R_0 < R$, either a circular shape (circular case, $d \leq R - R_0$, Figure D.1.b) or a more complex shape involving the intersection of circle and ellipse (elliptical case, $d > R - R_0$, Figure D.1.c) is to be determined.

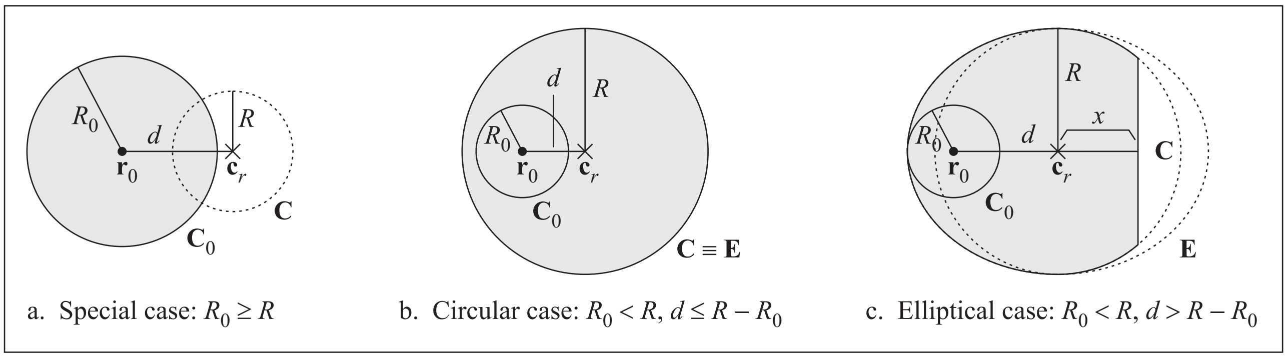


Figure D.1: Base shapes of the network topology exploration region.

D.2.2 Circular Case

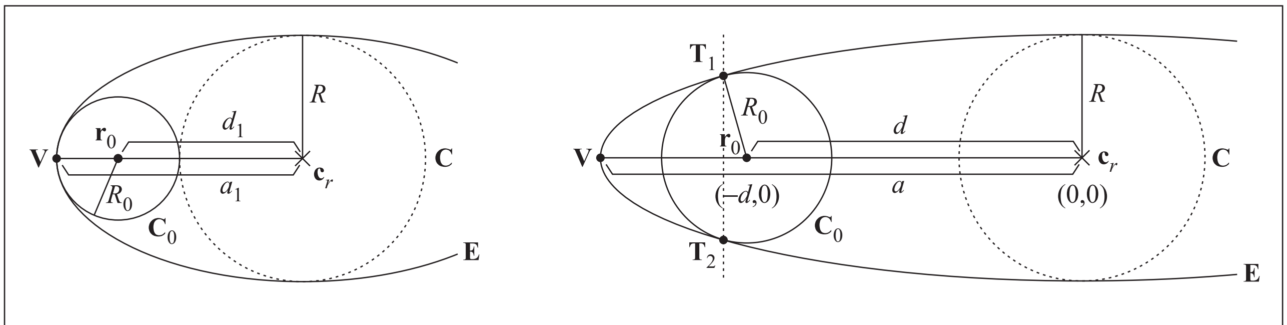
The circular shape in Figure D.1.b is defined by the condition that $d \leq d_0 = R - R_0$. In that case, the radius of the NTE region is R, and the region is centered at $\mathbf{c}_r$. Thus, the semimajor axis a of ellipse $\mathbf{E}$ (which in the circular case simply takes the form of a circle and is equivalent to $\mathbf{C}$) in the circular case is given by

$$\boxed{a = R \quad \text{for} \quad d \leq d_0} \tag{D.4}$$

D.2.3 Elliptical Case 1: Curvature Subcase

When the distance d between $\mathbf{r}_0$ and $\mathbf{c}_r$ reaches d_0, circle $\mathbf{C}_0$ around $\mathbf{r}_0$ touches circle $\mathbf{C}$ around $\mathbf{c}_r$. To guarantee that $\mathbf{r}_0$ maintains at least a distance d to any point on the NTE region, circle $\mathbf{C}$ is stretched into an ellipse $\mathbf{E}$ with increasing distance $d > d_0$. An ellipse is suitable because it allows the smooth stretching with increasing distances and provides both, good directionality and propagation characteristics (Requirement (1) in Section 4.3).

The ellipse $\mathbf{E}$ may first stretch until the curvature of the ellipse in its vertex $\mathbf{V}$ is equal to the curvature of the circle $\mathbf{C}_0$. This extremal case is shown in Figure D.2.a.



a. Elliptical case 1: curvature subcase. b. Elliptical case 2: tangent subcase.

Figure D.2: Elliptical case: curvature and tangent subcases.

While the curvature of $\mathbf{E}$ is greater than the curvature of $\mathbf{C}_0$, there is always a single point of intersection (tangent point) at $\mathbf{V}$ between $\mathbf{E}$ and $\mathbf{C}_0$. Thus, the second critical distance d_1 is defined as the situation where $\mathbf{E}$ and $\mathbf{C}_0$ have equal curvature. The curvature radius of $\mathbf{E}$ in vertex $\mathbf{V}$ of the major axis is defined as follows:

$$r_{\mathbf{V}} = \frac{b^2}{a} \tag{D.5}$$

The curvature radius of circle $\mathbf{C}$ is simply its radius R_0. The problem is to find the ellipse with length of the semiminor axis $b = R$ and $r_{\mathbf{V}} = R_0$. Thus, we solve (D.5) for extremal a_1:

$$R_0 = \frac{R^2}{a_1} \tag{D.6}$$

$$a_1 = \frac{R^2}{R_0} \tag{D.7}$$

We obtain the critical distance d_1 as follows:

$$d_1 = a_1 - R_0 \tag{D.8}$$

$$d_1 = \frac{R^2}{R_0} - R_0 \tag{D.9}$$

$$d_1 = \frac{1}{R_0}\left(R^2 - R_0^2\right) \tag{D.10}$$

Because (D.8) always holds (Figure D.2.a) it can be used to calculate a in general:

$$\boxed{a = R_0 + d \quad \text{for} \quad d_0 < d \leq d_1} \tag{D.11}$$

Observe that for $d = d_0 = R - R_0$, (D.4) and (D.11) take the same form.

D.2.4 Elliptical Case 2: Tangent Subcase

If ellipse $\mathbf{E}$ is stretched further, that is, distance d exceeds d_1, the curvature of $\mathbf{E}$ becomes smaller than that of $\mathbf{C}_0$. To maintain radius R_0 (Requirement (1) in Section 4.3), $\mathbf{E}$ must be placed tangent to $\mathbf{C}_0$ in two tangent points $\mathbf{T}_1$ and $\mathbf{T}_2$, as illustrated in Figure D.2.b. This case is applicable for all $d > d_1$.

The procedure to determine a as a function of d in the tangent subcase is to solve the equations for $\mathbf{E}$ and $\mathbf{C}_0$ under the constraints that i) there exists exactly two points of tangency $\mathbf{T}_1, \mathbf{T}_2$ between $\mathbf{E}$ and $\mathbf{C}_0$ and ii) that the length of the semiminor axis of $\mathbf{E}$ is $b = R$.

The general equation of a circle is

$$(x - x_m)^2 + (y - y_m)^2 = r^2 \tag{D.12}$$

where (x_m, y_m) denotes the center of the circle. In our particular case, the special equation for the circle located at $(x_m, y_m) = (-d, 0)$ with radius $r = R_0$ is

$$\mathbf{C}_0 : \ (x + d)^2 + y^2 = R_0^2 \tag{D.13}$$

The general equation of an ellipse is

$$\frac{(x - x_0)^2}{a^2} + \frac{(y - y_0)^2}{b^2} = 1 \tag{D.14}$$

where (x_0, y_0) denotes the center of the ellipse. In our case, we have $(x_0, y_0) = (0, 0)$ and $b = R$. Thus, the special form of the ellipse equation takes the form

$$\mathbf{E} : \ \frac{x^2}{a^2} + \frac{y^2}{R^2} = 1 \tag{D.15}$$

The problem is to solve the set of equations (D.13) and (D.15) for a under the constraint that $\mathbf{C}_0$ and $\mathbf{E}$ intersect at tangent points $\mathbf{T}_1, \mathbf{T}_2$ for $d > d_2$. We first transform (D.13) and get

$$y^2 = R_0^2 - (x + d)^2 \tag{D.16}$$

Then, we substitute y^2 in (D.15) with (D.16) and solve for x:

$$1 = \frac{x^2}{a^2} + \frac{R_0^2 - (x+d)^2}{R^2} \tag{D.17}$$

$$1 = \frac{x^2}{a^2} + \frac{R_0^2 - (x^2 + 2dx + d^2)}{R^2} \tag{D.18}$$

$$1 = \frac{x^2}{a^2} + \frac{R_0^2 - x^2 - 2dx - d^2}{R^2} \tag{D.19}$$

$$0 = \frac{x^2}{a^2} - \frac{x^2}{R^2} - \frac{2dx}{R^2} + \frac{R_0^2 - d^2}{R^2} - 1 \tag{D.20}$$

$$0 = \left(\frac{1}{a^2} - \frac{1}{R^2}\right) x^2 - \frac{2d}{R^2} x + \frac{R_0^2 - d^2}{R^2} - 1 \tag{D.21}$$

We can state the solutions of the quadratic equation in general form as follows:

$$x_{1,2} = \frac{-\beta \pm \sqrt{\beta^2 - 4\alpha\gamma}}{2\alpha} \tag{D.22}$$

where

$$\alpha = \frac{1}{a^2} - \frac{1}{R^2} \tag{D.23}$$

$$\beta = -\frac{2d}{R^2} \tag{D.24}$$

$$\gamma = \frac{R_0^2 - d^2}{R^2} - 1 \tag{D.25}$$

We are interested in the solutions where $x_1 = x_2$, which mark the tangent points $\mathbf{T}_1, \mathbf{T}_2$. Thus, we solve the discriminant such that

$$\beta^2 - 4\alpha\gamma = 0 \tag{D.26}$$

Plugging (D.23), (D.24), and (D.25) into (D.26) and solving for a yields:

$$0 = \left(-\frac{2d}{R^2}\right)^2 - 4\left(\frac{1}{a^2} - \frac{1}{R^2}\right)\left(\frac{R_0^2 - d^2}{R^2} - 1\right) \tag{D.27}$$

$$0 = \frac{4d^2}{R^4} - 4\left(\frac{1}{a^2} - \frac{1}{R^2}\right)\left(\frac{R_0^2 - d^2}{R^2} - 1\right) \tag{D.28}$$

$$\frac{d^2}{R^4} = \left(\frac{1}{a^2} - \frac{1}{R^2}\right)\left(\frac{R_0^2 - d^2}{R^2} - 1\right) \tag{D.29}$$

$$\frac{d^2}{R^4} = \left(\frac{1}{a^2} - \frac{1}{R^2}\right)\left(\frac{R_0^2 - d^2 - R^2}{R^2}\right) \tag{D.30}$$

$$\frac{1}{a^2} - \frac{1}{R^2} = \frac{d^2 R^2}{R^4 (R_0^2 - d^2 - R^2)} \tag{D.31}$$

$$\frac{1}{a^2} = \frac{d^2}{R^2 (R_0^2 - d^2 - R^2)} + \frac{1}{R^2} \tag{D.32}$$

$$\frac{1}{a^2} = \frac{d^2 + R_0^2 - d^2 - R^2}{R^2(R_0^2 - d^2 - R^2)} \tag{D.33}$$

$$a^2 = \frac{R^2(R_0^2 - d^2 - R^2)}{R_0^2 - R^2} \tag{D.34}$$

$$a^2 = \frac{R^2(R_0^2 - R^2) - R^2 d^2}{R_0^2 - R^2} \tag{D.35}$$

$$a^2 = R^2 - \frac{R^2 d^2}{R_0^2 - R^2} \tag{D.36}$$

$$a^2 = R^2 \left(1 - \frac{d^2}{R_0^2 - R^2}\right) \tag{D.37}$$

From the two solutions for a, only the positive value is relevant, thus we have:

$$\boxed{a = R\sqrt{1 + \frac{d^2}{R^2 - R_0^2}} \quad \text{for} \quad d > d_1} \tag{D.38}$$

Observe that for $d = d_1$, equations (D.38) and (D.11) yield the same value for a.

D.2.5 Summary of Cases

In summary, for given $d = |\mathbf{r}_0 - \mathbf{c}_r|$ we are able to determine the semimajor axis a of the ellipse $\mathbf{E}$ as a function of d. Taking the aforementioned three cases for the shapes the NTE region may take, from equations (D.4), (D.11), and (D.38), we have:

$$a(d) = \begin{cases} R & \text{for} \quad d \leq d_0 \\ R_0 + d & \text{for} \quad d_0 < d \leq d_1 \\ R\sqrt{1 + \frac{d^2}{R^2 - R_0^2}} & \text{for} \quad d > d_1 \end{cases} \tag{D.39}$$

with extremal values $d_0 = R - R_0$ and $d_1 = \frac{1}{R_0}(R^2 - R_0^2)$ taken from Section D.2.2 and (D.10), respectively. Note that $R > R_0$ is always assumed, and that $d_1 \geq d_0$ always holds.

D.3 NTE Region Specification

Based on ellipse $\mathbf{E}$ and circle $\mathbf{C}_0$ from Section D.2 we now define the NTE region $\mathbf{A}_{\text{NTE}}$ under the area constraint A_{NTE} and different configurations of d and a according to (D.39).

D.3.1 Special Case

We first consider the special case $R_0 \geq R$, where the NTE region is defined solely by the circle $\mathbf{C}_0$ around $\mathbf{r}_0$ (Figure D.1.a). Let $\mathbf{r}_i = (x_i, y_i)^T$ denote any position in the transformed coordinate system where $\mathbf{c}_r$ marks the origin and the x-axis extends on the straight line through

$\mathbf{r}_0$ and $\mathbf{c}_r$. Then, $\mathbf{r}_0 = (-d, 0)^T$, and $\mathbf{C}_0 : (x + d)^2 + y^2 \leq R_0^2$ denotes the circle around $\mathbf{r}_0$. The NTE region is defined as follows in the transformed coordinate system:

$$\boxed{\mathbf{A}_{\text{NTE}} \quad := \quad \{\mathbf{r}_i \in \mathbf{C}_0\} \quad \text{for} \quad R_0 \geq R} \tag{D.40}$$

D.3.2 Circular Case

In the circular case, where $R_0 < R$, the NTE region is the disk that is formed by circle $\mathbf{C}$ (Figure D.1.b) and becomes $\mathbf{C} : x^2 + y^2 \leq R^2$ in the transformed coordinate system. The definition of the NTE region can be stated immediately:

$$\boxed{\mathbf{A}_{\text{NTE}} \quad := \quad \{\mathbf{r}_i \in \mathbf{C}\} \quad \text{for} \quad R_0 < R \quad \wedge \quad d \leq d_0} \tag{D.41}$$

D.3.3 Elliptical Case 1: Curvature Subcase

When $d > d_1$ the ellipse $\mathbf{E}$ forms and the elliptical case occurs. For the curvature subcase, a distinction in two different subcase variants is necessary, shown in Figure D.3.

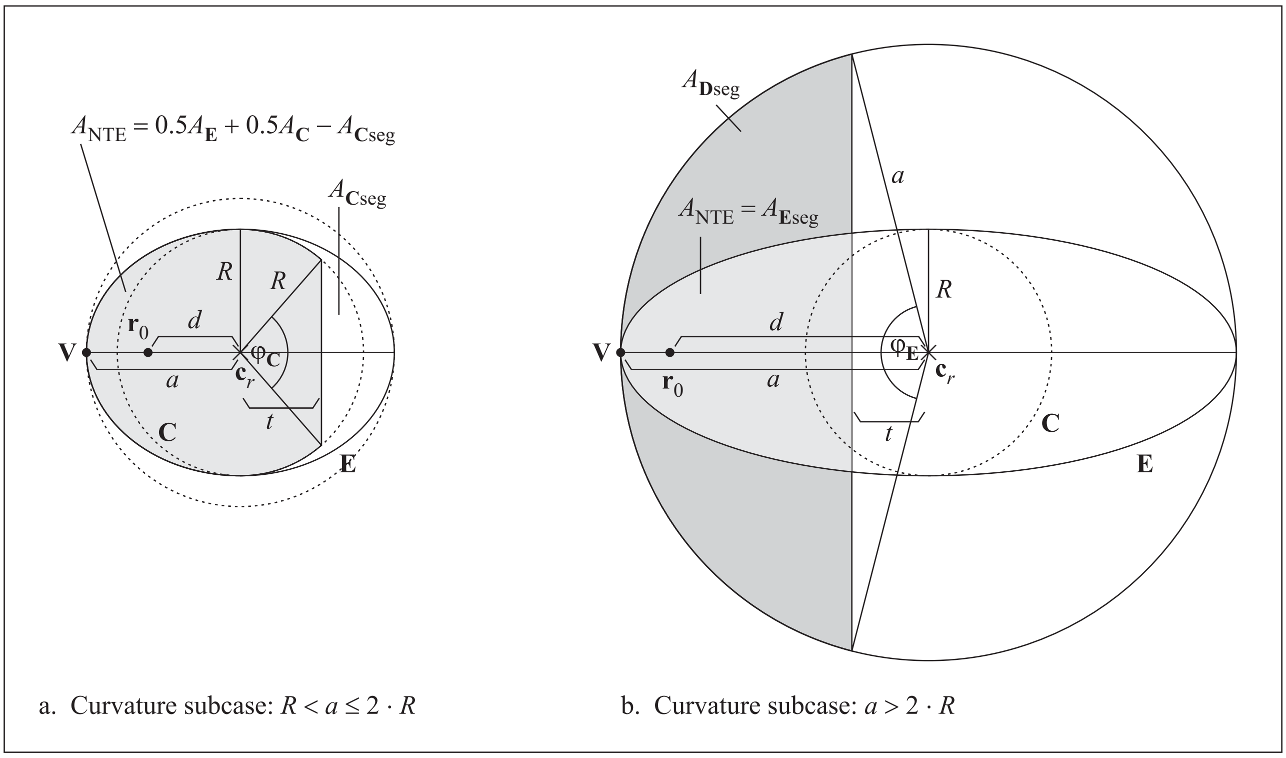


Figure D.3: Elliptical case 1: curvature subcase variants.

The distinction is made based on the condition that the NTE region is either completely located on the left or right side of the minor axis b. First, the area of the trivial ellipse (circle) is $A_{\mathbf{E}}^{a=b} = \pi b^2$. If $a = 2b$, then $A_{\mathbf{E}}^{a=2b} = \pi ab = \pi 2bb = 2\pi b^2 = 2A_{\mathbf{E}}^{a=b}$. Thus, for $b < a \leq 2b$, the area of the ellipse is extending beyond the minor axis, where for $a > 2b$ the area is located completely on the left side of the minor axis b. Note that this distinction is independent of the previously determined limit d_1 and that the possible cases depend on the proportion R/R_0.

First, (D.7) defines limit $a_1 = R^2/R_0$. Second, the critical distance for the area switch is $a' = 2b = 2R$. Thus, if the area switch' critical distance a' is beyond a_1, it will never be relevant, that is, when $a' > a_1 \Leftrightarrow 2R > R^2R_0 \Leftrightarrow 2 > R/R_0 \Leftrightarrow R > 2R_0$. In other words, if R is at least two times R_0, there will never be the case where the area is located only on the left side. For example, if $R = 1.5R_0$, then $a = 2b = 2R$ and according to (D.7), $a_1 = R^2/R_0 = R^2 \cdot 1.5/R = 1.5R$, in which case both possibilities exist, i.e., the NTE may be located on the left side or on both sides of the semiminor axis.

Case $R < a \leq 2 \cdot R$ (Figure D.3.a)

In this case, area A_{NTE} consists of one half of the area $A_{\mathbf{E}}$, plus half the area of circle $\mathbf{C}$, which is denoted $A_{\mathbf{C}}$, minus the circular segment $A_{\mathbf{C}\text{seg}}$:

$$A_{\text{NTE}} = \frac{1}{2}A_{\mathbf{E}} + \frac{1}{2}A_{\mathbf{C}} - A_{\mathbf{C}\text{seg}} \tag{D.42}$$

Applying the formula for the area of ellipse, circle, and circular segment, (D.42) becomes:

$$A_{\text{NTE}} = \frac{\pi a R}{2} + \frac{\pi R^2}{2} - \frac{R^2}{2}(\varphi_{\mathbf{C}} - \sin\varphi_{\mathbf{C}}) \tag{D.43}$$

Solving (D.43) for $\varphi_{\mathbf{C}}$ requires numerical methods because the angle appears both by itself and as the argument of the sine. In our situation, an approximation is sufficient because the NTE region is used for flooding topology requests. Thus, we achieve efficient approximation by specifying some $\Delta A_{\text{err}} \ll A_{\text{NTE}}$, the maximum areal error that we will tolerate. This value is at the same time the exit condition of the following iterative algorithm to determine $\varphi_{\mathbf{C}}$. Observe that $\varphi_{\mathbf{C}}$ may run from > 0 up to π and that the area increases strongly monotonic with decreasing $\varphi_{\mathbf{C}}$ (Figure D.3.a).

Initially, we set $t = \Delta t = R/2$. Note that $t \in [0, R]$. Observe that the maximum deviation from the actual area A_{NTE} is upper bound by $\Delta t \cdot R$. We execute the following procedure:

(1) compute angle $\varphi' = 2 \cdot \arccos(t/R)$

(2) compute A'_{NTE} using (D.43) with $\varphi_{\mathbf{C}} = \varphi'$

(3) if $\Delta t \cdot R < \Delta A_{\text{err}}$ or $A'_{\text{NTE}} = A_{\text{NTE}}$ then exit

(4) set $\Delta t := \Delta t/2$

(5) if $A'_{\text{NTE}} > A_{\text{NTE}}$ then set $t := t - \Delta t$ and goto (1)

(6) if $A'_{\text{NTE}} < A_{\text{NTE}}$ then set $t := t + \Delta t$ and goto (1)

The result A'_{NTE} is an approximation that is within the bounds specified by ΔA_{err}. At the same time, the algorithm outputs value t that fixes the extension of $\mathbf{A}_{\text{NTE}}$ to the right-side.

The full specification of the NTE region is as follows. Let $\mathbf{r}_i = (x_i, y_i)^T$. Then, $\mathbf{E} : \frac{x^2}{a^2} + \frac{y^2}{R^2} \leq 1$. The NTE region is defined as follows in the transformed coordinate system:

$$\boxed{\begin{array}{llllllll} \mathbf{A}_{\text{NTE}} & := & \mathbf{r}_i \in \mathbf{E} & \wedge & x_i \leq 0 & \vee & \mathbf{r}_i \in \mathbf{C} & \wedge \quad x_i \leq t \\ & \text{for} & R_0 < R & \wedge & d_0 < d \leq d_2 & \wedge & R < a \leq 2 \cdot R & \end{array}} \tag{D.44}$$

Case $a > 2 \cdot R$ (Figure D.3.b)

In the case where the NTE region does not extend beyond the semiminor axis R of ellipse $\mathbf{E}$, area A_{NTE} consists only of a single elliptical segment:

$$A_{\text{NTE}} = A_{\mathbf{E}\text{seg}} = \frac{b}{a} A_{\mathbf{D}\text{seg}} \tag{D.45}$$

Note that the identity used in (D.45) between ellipse and circle holds, because if we write the y coordinate as a function of x in the general ellipse equation, $y_e = b\sqrt{1 - \frac{x^2}{a^2}}$, and that of a circle for which $r = a$, $y_c = a\sqrt{1 - \frac{x^2}{a^2}}$, then $y_e/y_c = b/a$.

Applying the formula for the area of a circular segment, (D.45) becomes:

$$A_{\text{NTE}} = \frac{b}{a} \frac{a^2}{2} (\varphi_{\mathbf{E}} - \sin \varphi_{\mathbf{E}}) = \frac{ab}{2} (\varphi_{\mathbf{E}} - \sin \varphi_{\mathbf{E}}) \tag{D.46}$$

We use again numerical approximation with an error bound $\Delta A_{\text{err}} \ll A_{\text{NTE}}$. Observe that $\varphi_{\mathbf{E}}$ may run from > 0 up to $< \pi$. This time, A_{NTE} increases strongly monotonic with $\varphi_{\mathbf{E}}$ (Figure D.3.b). Setting $t = \Delta t = a/2$ with $t \in [0, a]$, the following procedure is executed:

```
(1) Compute angle $\varphi' = 2 \cdot \arccos(t/a)$
(2) Compute $A_{\text{NTE}}$ using (D.46) with $\varphi_{\mathbf{E}} = \varphi'$
(3) if $\Delta t \cdot R < \Delta A_{\text{err}}$ or $A'_{\text{NTE}} = A_{\text{NTE}}$ then exit
(4) set $\Delta t := \Delta t/2$
(5) If $A'_{\text{NTE}} > A_{\text{NTE}}$ then set $t := t + \Delta t$ and goto (1)
(6) if $A'_{\text{NTE}} < A_{\text{NTE}}$ then set $t := t - \Delta t$ and goto (1)
```

Again the algorithm outputs value t that determines the dimensions of $\mathbf{A}_{\text{NTE}}$ on the x-axis.

Let $\mathbf{r}_i = (x_i, y_i)^T$, $\mathbf{E}$ and $\mathbf{C}$ as before. With a modified restriction in x_i, the NTE region is defined as follows in the transformed coordinate system:

$$\boxed{\begin{array}{llllll} \mathbf{A}_{\text{NTE}} & := & \mathbf{r}_i \in \mathbf{E} & \wedge & x_i \leq -t & \\ & \text{for} & R_0 < R & \wedge & d_0 < d \leq d_2 & \wedge \quad a > 2 \cdot R \end{array}} \tag{D.47}$$

D.3.4 Elliptical Case 2: Tangent Subcase

In the tangent subcase, the NTE region is composed of elliptical and circular segments, illustrated in Figure D.4. Depending on distance d and values R, R_0, area A_{NTE} may or may not extend beyond the semiminor axis R of ellipse $\mathbf{E}$. However, note that it is not possible that A_{NTE} extends only up to the line connecting $\mathbf{T}_1$ and $\mathbf{T}_2$, because $A_{\mathbf{C}0} < A_{\mathbf{C}}$ always holds due to $R > R_0$. Also, it is not possible that the NTE region extends beyond circle $\mathbf{C}$, because $A_{\text{NTE}} = A_{\mathbf{C}}$ in the ideal case without approximation.

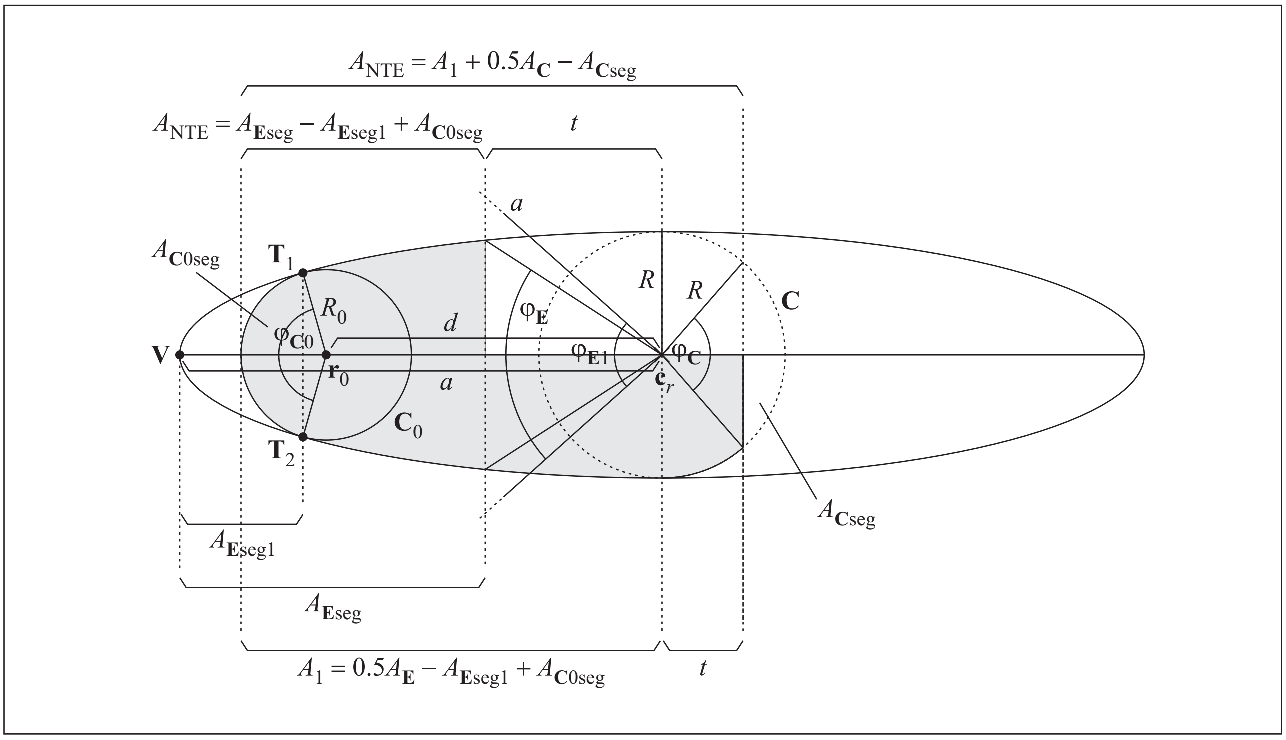

$A_{\text{NTE}} = A_1 + 0.5A_{\mathbf{C}} - A_{\mathbf{C}\text{seg}}$
$A_{\text{NTE}} = A_{\mathbf{E}\text{seg}} - A_{\mathbf{E}\text{seg1}} + A_{\mathbf{C}0\text{seg}}$
t
a
$A_{\mathbf{C}0\text{seg}}$
$\mathbf{T}_1$
R_0
$\varphi_{\mathbf{C}0}$
d
$\varphi_{\mathbf{E}}$
R
R
$\mathbf{C}$
$\mathbf{V}$
$\mathbf{r}_0$
a
$\varphi_{\mathbf{E}1}$
$\varphi_{\mathbf{C}}$
$\mathbf{c}_r$
$\mathbf{C}_0$
$\mathbf{T}_2$
$A_{\mathbf{C}\text{seg}}$
$A_{\mathbf{E}\text{seg1}}$
$A_{\mathbf{E}\text{seg}}$
$A_1 = 0.5A_{\mathbf{E}} - A_{\mathbf{E}\text{seg1}} + A_{\mathbf{C}0\text{seg}}$
t

Figure D.4: Elliptical case 2: tangent subcase variants.

We begin by determining the critical area A_1 that extends up to the semiminor axis R of $\mathbf{E}$:

$$\begin{aligned}
A_1 &= \frac{1}{2}A_{\mathbf{E}} - A_{\mathbf{E}\text{seg1}} + A_{\mathbf{C}0\text{seg}} && \text{(D.48)}\\
&= \frac{\pi}{2}ab - \frac{b}{a}A_{\mathbf{C}\text{seg1}} + A_{\mathbf{C}0\text{seg}} && \text{(D.49)}\\
&= \frac{\pi}{2}ab - \frac{b}{a}\frac{a^2}{2}(\varphi_{\mathbf{E}1} - \sin\varphi_{\mathbf{E}1}) + \frac{R_0^2}{2}(\varphi_{\mathbf{C}0} - \sin\varphi_{\mathbf{C}0}) && \text{(D.50)}\\
&= \frac{\pi}{2}ab - \frac{ab}{2}(\varphi_{\mathbf{E}1} - \sin\varphi_{\mathbf{E}1}) + \frac{R_0^2}{2}(\varphi_{\mathbf{C}0} - \sin\varphi_{\mathbf{C}0}) && \text{(D.51)}
\end{aligned}$$

Area A_1 can be calculated for known $\varphi_{\mathbf{E}1}, \varphi_{\mathbf{C}0}$, which are given by the following identities:

$$\cos\left(\frac{\varphi_{\mathbf{E}1}}{2}\right) = \frac{|x_{\mathbf{T}_1}|}{a} \quad \text{and} \quad \cos\left(\frac{\varphi_{\mathbf{C}0}}{2}\right) = \frac{|x_{\mathbf{T}_1}| - d}{R_0} \tag{D.52}$$

Value $|x_{\mathbf{T}_1}|$ can be obtained from (D.22) under the assumption that $\beta^2 - 4\alpha\gamma = 0$, thus:

$$|x_{\mathbf{T}_1}| = \left|\frac{-\beta}{2\alpha}\right| \tag{D.53}$$

Plugging in α and β from equations (D.23) and (D.24), respectively, we obtain:

$$|x_{\mathbf{T}_1}| = \left|\frac{-\left(-\frac{2d}{R^2}\right)}{2\left(\frac{1}{a^2}-\frac{1}{R^2}\right)}\right| = \left|\frac{d}{R^2\left(\frac{1}{a^2}-\frac{1}{R^2}\right)}\right| = \left|\frac{d}{\frac{R^2}{a^2}-1}\right| = \left|\frac{da^2}{R^2-a^2}\right| \tag{D.54}$$

Based on the critical area A_1, we make the following distinction to determine $\mathbf{A}_{\text{NTE}}$.

Case $A_{\text{NTE}} \le A_1$ (Figure D.4, Upper Shaded Region)

In this case, we obtain from Figure D.4 the following equation:

$$A_{\text{NTE}} = A_{\mathbf{E}\text{seg}} - A_{\mathbf{E}\text{seg1}} + A_{\mathbf{C}0\text{seg}} \tag{D.55}$$

We use the following intermediary result that is a transformation of (D.48):

$$-A_{\mathbf{E}\text{seg1}} + A_{\mathbf{C}0\text{seg}} = A_1 - \frac{1}{2}A_{\mathbf{E}} \tag{D.56}$$

We can substitute the right side of (D.56) for the rightmost two terms in (D.55) and obtain

$$A_{\text{NTE}} = A_{\mathbf{E}\text{seg}} + A_1 - \frac{1}{2}A_{\mathbf{E}} = \frac{ab}{2}(\varphi_{\mathbf{E}} - \sin\varphi_{\mathbf{E}}) + A_1 - \frac{\pi}{2}ab \tag{D.57}$$

which leaves only $\varphi_{\mathbf{E}}$ variable. Note that the first term is obtained from (D.46).

Like in the previous calculations we use numerical approximation with an error bound $\Delta A_{\text{err}} \ll A_{\text{NTE}}$. (D.57) is applicable for $\varphi_{\mathbf{E}}$ between $\varphi_{\mathbf{E}1}$ to π. Initially, we set $t = \Delta t = \frac{1}{2}|x_{\mathbf{T}_1}|$. We then execute the following iterative algorithm:

```
(1) Compute angle $\varphi' = 2 \cdot \arccos(t/a)$
(2) Compute $A'_{\text{NTE}}$ using (D.57), with $\varphi_{\mathbf{E}} = \varphi'$
(3) if $\Delta t \cdot R < \Delta A_{\text{err}}$ or $A'_{\text{NTE}} = A_{\text{NTE}}$ then exit
(4) set $\Delta t := \Delta t/2$
(5) if $A'_{\text{NTE}} > A_{\text{NTE}}$ then set $t := t + \Delta t$ and goto (1)
(6) if $A'_{\text{NTE}} < A_{\text{NTE}}$ then set $t := t - \Delta t$ and goto (1)
```

The algorithm outputs A'_{NTE} within bounds ΔA_{err} and t that fixes the dimensions of $\mathbf{A}_{\text{NTE}}$.

Let $\mathbf{r}_i = (x_i, y_i)^T$ and $\mathbf{r}_0 = (-d, 0)^T$ in the transformed coordinate system. Then, $\mathbf{C}_0 : (x+d)^2 + y^2 \le R_0^2$ and $\mathbf{E} : \frac{x^2}{a^2} + \frac{y^2}{R^2} \le 1$ with $a = R\sqrt{1 + \frac{d^2}{R^2+R_0^2}}$ according to (D.38). The NTE region is defined as follows for $A \le A_1$:

$$\boxed{\begin{array}{llllll} \mathbf{A}_{\text{NTE}} & := & \mathbf{r}_i \in \mathbf{C}_0 & \wedge & x_i \le x_{\mathbf{T}_1} & \\ & \vee & \mathbf{r}_i \in \mathbf{E} & \wedge & x_{\mathbf{T}_1} < x_i \le -t & \\ & \text{for} & R_0 < R & \wedge & d > d_1 & \wedge \quad A_{\text{NTE}} \le A_1 \end{array}} \tag{D.58}$$

Case $A_{\text{NTE}} > A_1$ (Figure D.4, Lower Shaded Region)

In this case, Figure D.4 yields the following equation:

$$A_{\text{NTE}} = A_1 + \frac{1}{2}A_{\mathbf{C}} - A_{\mathbf{C}\text{seg}} = A_1 + \frac{\pi}{2}R^2 - \frac{R^2}{2}(\varphi_{\mathbf{C}} - \sin\varphi_{\mathbf{C}}) \tag{D.59}$$

The following algorithm yields t, where $\varphi_{\mathbf{C}}$ runs from π to > 0. Initially, $t = \Delta t = R/2$.

(1) Compute angle $\varphi' = 2 \cdot \arccos(t/R)$

(2) Compute A'_{NTE} using (D.59) with $\varphi_{\mathbf{C}} = \varphi'$.

(3) if $\Delta t \cdot R < \Delta A_{\text{err}}$ or $A'_{\text{NTE}} = A_{\text{NTE}}$ then exit

(4) set $\Delta t := \Delta t/2$

(5) if $A'_{\text{NTE}} > A_{\text{NTE}}$ then set $t := t - \Delta t$ and goto (1)

(6) if $A'_{\text{NTE}} < A_{\text{NTE}}$ then set $t := t + \Delta t$ and goto (1)

Let $\mathbf{r}_i = (x_i, y_i)^T$ and $\mathbf{C}_0$, $\mathbf{E}$, and a be defined as above, and let $\mathbf{C} : x^2 + y^2 \leq R^2$ in the transformed coordinate system. Then, for $A_{\text{NTE}} > A_1$, the NTE region is defined as follows:

$$\boxed{\begin{array}{llllll} \mathbf{A}_{\text{NTE}} & : & \mathbf{r}_i \in \mathbf{C}_0 & \wedge & x_i \leq x_{\mathbf{T}_1} & \\ & \vee & \mathbf{r}_i \in \mathbf{E} & \wedge & x_{\mathbf{T}_1} < x_i \leq 0 & \\ & \vee & \mathbf{r}_i \in \mathbf{C} & \wedge & 0 < x_i \leq t & \\ & \text{for} & R_0 < R & \wedge & d > d_1 & \wedge \quad A_{\text{NTE}} > A_1 \end{array}} \tag{D.60}$$

D.3.5 Summary of Cases

The set of equations, (D.40), (D.41), (D.44), (D.47), (D.58), and (D.60), defines all possible shapes and thus completely the NTE region. The full set of definitions is implemented by the prototype in the evaluation of data migration in Section 4.6.

Refereed Publications

[DWM08] Dominique Dudkowski, Harald Weinschrott, and Pedro José Marrón. Design and Implementation of a Reference Model for Context Management in Mobile Ad-Hoc Networks. In *Proceedings of the 22nd International Conference on Advanced Information Networking and Applications - Workshops (AINAW 2008)*, pages 832–837, Gino-wan, Okinawa, Japan, March, 2008. IEEE Computer Society.

[DMR07] Dominique Dudkowski, Pedro José Marrón, and Kurt Rothermel. Migration Policies for Location-Centric Storage in Mobile Ad-Hoc Networks. In *Proceedings of the 3rd International Conference on Mobile Ad-hoc and Sensor Networks (MSN 2007)*, pages 197–208, Beijing, China, December 2007. Lecture Notes in Computer Science, Volume 4864/2007, Springer Berlin / Heidelberg.

[HDM+07] Jörg Hähner, Dominique Dudkowski, Pedro José Marrón, and Kurt Rothermel. Quantifying Network Partitioning in Mobile Ad Hoc Networks. In *Proceedings of the 8th International Conference on Mobile Data Management (MDM 2007)*, pages 174–181, Mannheim, Germany, May 2007. IEEE.

[DMR06b] Dominique Dudkowski, Pedro José Marrón, and Kurt Rothermel. Efficient Algorithms for Probabilistic Spatial Queries in Mobile Ad Hoc Networks. In *Proceedings of the First International Conference on Communication System Software and Middleware (COMSWARE 2006)*, New Delhi, India, January 2006. IEEE.

[DMR06a] Dominique Dudkowski, Pedro José Marrón, and Kurt Rothermel. An Efficient Resilience Mechanism for Data Centric Storage in Mobile Ad Hoc Networks. In *Proceedings of the 7th International Conference on Mobile Data Management (MDM 2006)*, Nara, Japan, May 2006. IEEE Computer Society.

[DDM05] Dominique Dudkowski, Tobias Drosdol, and Pedro José Marrón. Towards Scalable and Efficient Processing of Probabilistic Spatial Queries in Mobile Ad Hoc and Sensor Networks. In *Mobile Datenbanken: heute, morgen und in 20 Jahren. 8. Workshop des GI-Arbeitskreises "Mobile Datenbanken und Informationssysteme"*, Karlsruhe, February/March, 2005.

[HDM+05] Jörg Hähner, Dominique Dudkowski, Pedro José Marrón, and Kurt Rothermel. A Quantitative Analysis of Partitioning in Mobile Ad Hoc Networks. *Extended Abstract in Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS 2004/Performance 2004)*, pages 400–401, New York, NY, USA, June 2004. ACM Press.

Bibliography

[ACNP07] Michele Albano, Stefano Chessa, Francesco Nidito, and Susanna Pelagatti. Q-NiGHT: Adding QoS to data centric storage in non-uniform sensor networks. In *Proceedings of the 8th International Conference on Mobile Data Management (MDM'07)*, pages 100–107, Mannheim, Germany, May 2007. 10, 84, 85, 86, 93

[ADM04] Ittai Abraham, Danny Dolev, and Dahlia Malkhi. LLS: a locality aware location service for mobile ad hoc networks. In *Proceedings of the 2004 Joint Workshop on Foundations of Mobile Computing*, pages 75–84, Philadelphia, Pennsylvania, USA, October 2004. 86, 87

[ARK$^+$05] Filipe Araújo, Luís Rodrigues, Jörg Kaiser, Changling Liu, and Carlos Mitidieri. CHR: a distributed hash table for wireless ad hoc networks. In *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW'05)*, pages 407–413, Columbus, Ohio, USA, June 2005. 83, 84, 85, 86, 100

[AWW05] Ian F. Akyildiz, Xudong Wang, and Weilin Wang. Wireless mesh networks: A survey. *Computer Networks*, 47(4):445–487, March 2005. 46

[BA06] Kenneth C. Barr and Krste Asanović. Energy-aware lossless data compression. *ACM Transactions on Computer Systems (TOCS)*, 24(3):250–291, August 2006. 48

[Bar99] Daniel Barbará. Mobile computing and databases - a survey. *IEEE Transactions on Knowledge and Data Engineering*, 11(1):108–117, January 1999. 72

[BCQ$^+$07] Cristiana Bolchini, Carlo A. Curino, Elisa Quintarelli, Fabio A. Schreiber, and Letizia Tanca. A data-oriented survey of context models. *ACM SIGMOD Record*, 36(4):19–26, December 2007. 40

[BDR07] Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2(4):263–277, 2007. 78

[Bet02] Christian Bettstetter. On the minimum node degree and connectivity of a wireless multihop network. In *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'02)*, pages 80–91, Lausanne, Switzerland, June 2002. 50

[Bet03] Christian Bettstetter. Topology properties of ad hoc networks with random waypoint mobility model. *ACM SIGMOBILE Mobile Computing and Communications Review*, 7(3):50–52, July 2003. 50

[Bha03] Sangeeta Bhattacharya. Randomized location service in mobile ad hoc networks. In *Proceedings of the 6th International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM'03)*, pages 66–73, San Diego, California, USA, September 2003. 73, 86, 87

[BJKS02] Rimantas Benetis, Christian S. Jensen, Gytis Karciauskas, and Simonas Saltenis. Nearest neighbor and reverse nearest neighbor queries for moving objects. In *Proceedings of the International Database Engineering and Applications Symposium (IDEAS'02)*, pages 44–53, Edmonton, Canada, July 2002. 86, 87

[BMJ$^+$98] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pages 85–97, Dallas, Texas, USA, October 1998. 58, 123, 124, 168

[Bre00] Eric A. Brewer. Towards robust distributed systems (invited talk). In *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing*, page 7, Portland, Oregon, USA, July 2000. ACM New York, NY, USA. 92

[CAMCM05] Shiva Chetan, Jalal Al-Muhtadi, Roy Campbell, and M. Dennis Mickunas. Mobile Gaia: A middleware for ad-hoc pervasive computing. In *Second IEEE Consumer Communications and Networking Conference (CCNC'05)*, pages 223–228, Las Vegas, Nevada, USA, January 2005. 79

[CC07] Jinchuan Chen and Reynold Cheng. Efficient evaluation of imprecise location-dependent queries. In *Proceedings of the IEEE 23rd International Conference on Data Engineering (ICDE'07)*, pages 586–595, Istanbul, Turkey, April 2007. 89

[CCMC08] Reynold Cheng, Jinchuan Chen, Mohamed Mokbel, and Chi-Yin Chow. Probabilistic verifiers: Evaluating constrained nearest-neighbor queries over uncertain data. In *Proceedings of the IEEE 24th International Conference on Data Engineering (ICDE'08)*, pages 973–982, Cancun, Mexico, April 2008. 89

[CGP04] Jian Chen, Yong Guan, and U. Pooch. Customizing GPSR for wireless sensor networks. In *Proceedings of the 2004 IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, pages 549–551, Fort Lauderdale, Florida, USA, October 2004. 82

[CKP03] Reynold Cheng, Dmitri V. Kalashnikov, and Sunil Prabhakar. Evaluating probabilistic queries over imprecise data. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 551–562, San Diego, California, USA, June 2003. 89

[CKP04] Reynold Cheng, Dmitri V. Kalashnikov, and Sunil Prabhakar. Querying imprecise data in moving object environments. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1112–1127, September 2004. 89

[CNS04] Alexandru Coman, Mario A. Nascimento, and Jörg Sander. A framework for spatio-temporal query processing over wireless sensor networks. In *Proceedings of the First Workshop on Data Management for Sensor Networks (DMSN'04)*, Toronto, Canada, August 2004. 86, 87

[CP03] Reynold Cheng and Sunil Prabhakar. Managing uncertainty in sensor databases. *ACM SIGMOD Record. Special Issue: Special Section on Sensor Network Technology and Sensor Data Managment*, 32(4):41–46, December 2003. 89

[CPK03] Reynold Cheng, Sunil Prabhakar, and Dmitri V. Kalashnikov. Querying imprecise data in moving object environments. In *Proceedings of the 19th International Conference on Data Engineering (ICDE'03)*, pages 723–725, Bangalore, India, March 2003. 89

[CXSN04] Kai Chen, Yuan Xue, Samarth H. Shah, and Klara Nahrstedt. Understanding bandwidth-delay product in mobile ad hoc networks. *Computer Communications*, 27(10):923–934, June 2004. 158

[CYC07] Tzung-Shi Chen, Gwo-Jong Yu, and Hsin-Ju Chen. A framework of mobile context management for supporting context-aware environments in mobile ad hoc networks. In *International Wireless Communications and Mobile Computing Conference 2007 (IWCMC'07)*, pages 647–652, Honolulu, Hawaii, USA, August 2007. 81

[DA99] Anind K. Dey and Gregory D. Abowd. Towards a better understanding of context and context-awareness. Technical Report GIT-GVU-99-22, Georgia Institute of Technology, Atlanta, Georgia, USA, 1999. 39

[DF03] Murat Demirbas and Hakan Ferhatosmanoglu. Peer-to-peer spatial queries in sensor networks. In *Proceedings of the Third International Conference on Peer-to-Peer Computing (P2P'03)*, pages 32–39, Linköping, Sweden, September 2003. 86, 87

[DGM$^+$04] Amol Deshpande, Carlos Guestrin, Samuel R. Madden, Joseph M. Hellerstein, and Wei Hong. Model-driven data acquisition in sensor networks. In *Proceedings of the 30th VLDB Conference*, pages 588–599, Toronto, Canada, August 2004. 190

[DHH07] Anusuriya Devaraju, Simon Hoh, and Michael Hartley. A context gathering framework for context-aware mobile solutions. In *Proceedings of the 4th International Conference on Mobile Technology, Applications, and Systems and the 1st International Symposium on Computer Human Interaction in Mobile Technology*, pages 39–46, Singapore, September 2007. 78, 79

[DLR07] Tobias Drosdol, Ralph Lange, and Kurt Rothermel. Energy-efficient tracking of mobile objects with early distance-based reporting. In *Proceedings of the Fourth Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'07)*, Philadelphia, PA, USA, August 2007. 48

[Dun97] Steven R. Dunbar. The average distance between points in geometric figures. *The College Mathematics Journal*, 28(3):187–197, May 1997. 232

[EM99] P. Enge and P. Misra. Scanning the issue/technology - special issue on global positioning system. *Proceedings of the IEEE*, 87(1):3–15, 1999. 39, 91

[ERS06] Cheng Tien Ee, Sylvia Ratnasamy, and Scott Shenker. Practical data-centric storage. In *Proceedings of the 3rd Symposium on Networked Systems Design and Implementation (NSDI'06)*, pages 325–338, San Jose, California, USA, May 2006. 84, 85, 86

[FC06] Patrick Fahy and Siobhán Clarke. CASS: Middleware for mobile, context-aware applications. In *MobiSys 2004 Workshop on Context Awareness*, Boston, Massachusetts, USA, June 2006. 79

[FPL07] Tao-Young Fu, Wen-Chih Peng, and Wang-Chien Lee. Optimizing parallel itineraries for KNN query processing in wireless sensor networks. In *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management*, pages 391–400, Lisboa, Portugal, November 2007. 86, 87, 88

[FRWZ07] Elena Fasolo, Michele Rossi, Jörg Widmer, and Michele Zorzi. In-network aggregation techniques for wireless sensor networks: A survey. *IEEE Wireless Communications*, 14(2):70–87, April 2007. 145

[FSAA01] Hakan Ferhatosmanoglu, Ioanna Stanoi, Divyakant Agrawal, and Amr El Abbadi. Constrained nearest neighbor queries. In *Proceedings of the 7th International Symposium on Advances in Spatial and Temporal Databases (SSTD'01)*, pages 257–278, Redondo Beach, California, USA, January 2001. 86, 87

[FV03] Dieter Fritsch and Steffen Volz. Nexus - the mobile GIS-environment. In *Proceedings of the Joint First Workshop on Mobile Future and Symposium on Trends in Communications (SympoTIC'03)*, pages 1–4, Bratislava, Slovak Republic, October 2003. 40

[FW06] Roland Flury and Roger Wattenhofer. MLS: An efficient location service for mobile ad hoc networks. In *Proceedings of the Seventh ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'06)*, pages 226–237, Florence, Italy, May 2006. 86, 87

[GA02] Sanny Gustavsson and Sten F. Andler. Self-stabilization and eventual consistency in replicated real-time databases. In *Proceedings of the First Workshop on Self-healing Systems (WOSS'02)*, pages 105–107, Charleston, South Carolina, USA, November 2002. 92

[Gar09] Garmin GPSMAP 76CSx. http://www.garmin.com/, September 2009. 182

[GBH$^+$05] Matthias Grossmann, Martin Bauer, Nicola Hönle, Uwe-Philipp Käppeler, Daniela Nicklas, and Thomas Schwarz. Efficiently managing context information for large-scale scenarios. In *Proceedings of the 3rd IEEE International Conference on Pervasive Computing and Communications (PerCom'05)*, pages 331–340, Kauai, Hawaii, USA, March 2005. 40, 78

[GdWFM02] Michael Gerharz, Christian de Waal, Matthias Frank, and Peter Martini. Link stability in mobile wireless ad hoc networks. In *Proceedings of the 27th Annual IEEE Conference on Local Computer Networks (LCN'02)*, pages 30–39, Tampa, Florida, USA, November 2002. 50, 151

[GEG$^+$03] Benjamin Greenstein, Deborah Estrin, Ramesh Govindan, Sylvia Ratnasamy, and Scott Shenker. DIFS: A distributed index for features in sensor networks. In *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications (SNPA'03)*, pages 163–173, Anchorage, Alaska, USA, May 2003. 86, 87

[GGC03] Abhishek Ghose, Jens Grossklags, and John Chuang. Resilient data-centric storage in wireless ad-hoc sensor networks. In *Proceedings of the 4th International Conference on Mobile Data Management (MDM'03)*, pages 45–62, Melbourne, Australia, January 2003. 11, 81, 85, 86

[GL02] Seth Gilbert and Nancy Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *ACM SIGACT News*, 33(2):51–59, June 2002. 92

[GPZ05] Tao Gu, Hung Keng Pung, and Da Qing Zhang. A service-oriented middleware for building context-aware services. *Journal of Network and Computer Applications*, 28:1–18, 2005. 79

[GRS$^+$03] Benjamin Greenstein, Sylvia Ratnasamy, Scott Shenker, Ramesh Govindan, and Deborah Estrin. DIFS: A distributed index for features in sensor networks. *Ad Hoc Networks*, 1(2-3):333–349, September 2003. 86, 87

[GS02] Sandeep K. S. Gupta and Pradip K. Srimani. *Handbook of Wireless Networks and Mobile Computing*, chapter Data Management in Wireless Mobile Environments, pages 553–579. Wiley & Sons, February 2002. 72

[HBLR05] Qingfeng Huand, Sangeeta Bhattacharya, Chenyang Lu, and Gruia-Catalin Roman. FAR: Face-aware routing for mobicast in large-scale sensor networks. *ACM Transactions on Sensor Networks*, 1(2):240–271, November 2005. 82

[HBR04] Jörg Hähner, Christian Becker, and Kurt Rothermel. Update-linearizability: A consistency concept for the chronological ordering of events in MANETs. In *Proceedings of the 2004 IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, pages 1–10, Fort Lauderdale, Florida, USA, October 2004. 100

[HHSW05] Jessica Heesen, Christoph Hubig, Oliver Siemoneit, and Klaus Wiegerling. Leben in einer vernetzten und informatisierten Welt. Context-Awareness im Schnittfeld von Mobile und Ubiquitous Computing. SFB 627 Bericht 2005/05, Center of Excellence 627, Universität Stuttgart, March 2005. 38, 39

[HKL+99] Fritz Hohl, Uwe Kubach, Alexander Leonhardi, Kurt Rothermel, and Markus Schwehm. Next century challenges: Nexus - an open global infrastructure for spatial-aware applications. In *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99)*, pages 249–255, Seattle, Washington, USA, August 1999. 40

[HLR04] Qingfeng Huang, Chenyang Lu, and Gruia-Catalin Roman. Reliable mobicast via face-aware routing. In *Proceedings of the Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'04)*, volume 3, pages 2108–2118, Hong Kong, China, March 2004. 82

[HS01] Hung-Yun Hsieh and Raghupathy Sivakumar. Performance comparison of cellular and multi-hop wireless networks: A quantitative study. In *Proceedings of the 2001 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'01)*, pages 113–122, Cambridge, Massachusetts, USA, June 2001. 51

[HSC03] Michail Hauspie, David Simplot, and Jean Carle. Partition detection in mobile ad-hoc networks using multiple disjoint paths set. In *Proceedings of the 1st International Workshop on Objects Models and Multimedia Technologies*, Geneva, Switzerland, September 2003. 151

[IB93] Tomasz Imielinski and B. R. Badrinath. Data management for mobile computing. *ACM SIGMOD Record*, 22(1):34–39, March 1993. 37, 72

[IB94] Tomasz Imielinski and B. R. Badrinath. Mobile wireless computing: Challenges in data management. *Communications of the ACM*, 37(10):18–28, October 1994. 72

[JS03] Glenn Judd and Peter Steenkiste. Providing contextual information to pervasive computing applications. In *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom'03)*, page 133, Dallas-Fort Worth, Texas, USA, March 2003. 79

[KD04] Vijay Kumar and Samir R. Das. Performance of dead reckoning-based location service for mobile ad hoc networks. *Wireless Communications and Mobile Computing*, 4(2):189–202, 2004. 86, 87

[KFWM04] Wolfgang Kieß, Holger Füßler, Jörg Widmer, and Martin Mauve. Hierarchical location service for mobile ad-hoc networks. *Mobile Computing and Communications Review*, 1(2):47–58, October 2004. 73, 85, 86, 87, 184

[KGKS05] Young-Jin Kim, Ramesh Govindan, Brad Karp, and Scott Shenker. Geographic routing made practical. In *Proceedings of the 2nd Symposium on Networked*

Systems Design & Implementation (NSDI'05), pages 217–230, Boston, Massachusetts, USA, May 2005. 82

[Khe06] Kavi K. Khedo. Context-aware systems for mobile and ubiquitous networks. In *Proceedings of the International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies 2006 (ICN/ICONS/MCL'06)*, page 123, Morne, Mauritius, April 2006. 78

[Kja07] Kristian E. Kjaer. A survey of context-aware middleware. In *Proceedings of IASTED Software Engineering Conference 2007*, August 2007. 78

[KK00] Brad Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MobiCom'00)*, pages 243–254, Boston, Massachusetts, USA, August 2000. 10, 81, 84, 128

[KK02] Mohamed Khedr and Ahmed Karmouch. ACAN - ad hoc context aware network. In *Proceedings of the Canadian Conference on Electrical & Computer Engineering (CCECE'02)*, pages 1342–1346, Winnipeg, Canada, May 2002. 80

[KKR07] Hans-Peter Kriegel, Peter Kunath, and Matthias Renz. Probabilistic nearest-neighbor query on uncertain objects. In *Proceedings of the 12th International Conference on Database Systems for Advanced Applications (DASFAA'07)*, pages 337–348, Bangkok, Thailand, April 2007. Springer Berlin / Heidelberg. 89

[Kle95] Leonhard Kleinrock. Nomadic computing - an opportunity. *ACM SIGCOMM Computer Communication Review*, 25(1):36–40, January 1995. 37

[Kum06] Vijay Kumar. *Mobile Database Systems (Wiley Series on Parallel and Distributed Computing)*. Wiley-Interscience, June 2006. 72

[KWZ02] Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger. Asymptotically optimal geometric mobile ad-hoc routing. In *Proceedings of the 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIALM'02)*, pages 24–33, New York, NY, USA, September 2002. 82

[KWZ03] Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger. Worst-case optimal and average-case efficient geometric ad-hoc routing. In *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'03)*, pages 267–278, Annapolis, Maryland, USA, June 2003. 82

[KWZZ03] Fabian Kuhn, Roger Wattenhofer, Yan Zhang, and Aaron Zollinger. Geometric ad-hoc routing: Of theory and practice. In *Proceedings of the Twenty-Second Annual Symposium on Principles of Distributed Computing (PODC'03)*, pages 63–72, Boston, Massachusetts, USA, July 2003. 82

[LC04] Jennifer J.-N. Liu and Imrich Chlamtac. *Mobile Ad Hoc Networking*, chapter Mobile Ad-Hoc Networking with a View of 4G Wireless: Imperatives and Challenges, pages 1–46. Wiley-Interscience, 2004. 45

[LC08] Xiang Lian and Lei Chen. Probabilistic group nearest neighbor queries in uncertain databases. *IEEE Transactions on Knowledge and Data Engineering*, 20(6):809–824, June 2008. 89, 90

[LCN05] Xinwei Luo, Tracy Camp, and William Navidi. Predictive methods for location services in mobile ad hoc networks. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)*, page 6, Denver, Colorado, USA, April 2005. 86, 87

[LCP+05] Eng K. Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma, and Steven Lim. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys & Tutorials*, 7(2):72–93, 2005. 88

[LGW06] Olaf Landsiegel, Stefan Götz, and Klaus Wehrle. Towards scalable mobility in distributed hash tables. In *Proceedings of the Sixth IEEE International Conference on Peer-to-Peer Computing (P2P'06)*, pages 203–209, Cambridge, UK, September 2006. 85

[LJC+00] Jinyang Li, John Jannotti, Douglas S. J. De Couto, David R. Karger, and Robert Morris. A scalable location service for geographic ad hoc routing. In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCom'00)*, pages 120–130, Boston, Massachusetts, USA, August 2000. 73, 86, 87, 184

[LKGH03] Xin Li, Young-Jin Kim, Ramesh Govindan, and Wei Hong. Multi-dimensional range queries in sensor networks. In *Proceedings of the ACM Conference on Embedded Networked Sensor Systems (SenSys'03)*, Los Angeles, California, USA, November 2003. ACM Press. 86, 87

[LLM06] Ben Leong, Barbara Liskov, and Robert Morris. Geographic routing without planarization. In *Proceedings of the 3rd Symposium on Networked Systems Design and Implementation (NSDI'06)*, pages 339–352, San Jose, California, USA, May 2006. 82

[LSD+02] Hui Lei, Daby M. Sow, John S. Davis, Guruduth Banavar, and Maria R. Ebling. The design and applications of a context service. *ACM SIGMOBILE Mobile Computing and Communications Review*, 6(4):45–55, October 2002. 79

[LTLS00] Wen-Hwa Liao, Yu-Chee Tseng, Kuo-Lun Lo, and Jang-Ping Sheu. GeoGRID: a geocasting protocol for mobile ad hoc networks based on GRID. *Journal of Internet Technology*, 1(2):23–32, December 2000. 85

[Mai04] Christian Maihöfer. A survey of geocast routing protocols. *IEEE Communications Surveys and Tutorials*, 6(2):32–42, 2004. 145

[MLS05] Christian Maihöfer, Tim Leinmüller, and Elmar Schoch. Abiding geocast: Time-stable geocast for ad hoc networks. In *Proceedings of the 2nd ACM International Workshop on Vehicular Ad Hoc Networks (VANET'05)*, pages 20–29, Cologne, Germany, September 2005. 85

[MMM05] Bratislav Milic, Nikola Milanovic, and Miroslaw Malek. Prediction of partitioning in location-aware mobile ad hoc networks. In *Proceedings of the 38th Hawaii International Conference on System Sciences (HICSS'05)*, page 306c, Hilton Waikoloa Village, HI, USA, January 2005. 149, 151

[NS03] James Newsome and Dawn Song. GEM: Graph EMbedding for routing and data-centric storage in sensor networks without geographic information. In *Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys'03)*, pages 76–88, Redwood, California, USA, November 2003. 84

[NTCS99] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping Sheu. The broadcast storm problem in a mobile ad hoc network. In *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99)*, pages 151–162, Seattle, Washington, USA, August 1999. 49, 50, 54

[OT98] Katia Obraczka and Gene Tsudik. Multicast routing issues in ad hoc networks. In *Proceedings of the 1998 International Conference on Universal Personal Communications*, pages 751–756, Florence, Italy, October 1998. 54

[PJC06] Filip Perich, Anupam Joshi, and Rada Chirkova. *Enabling Technologies for Wireless E-Business*, chapter Data Management for Mobile Ad-Hoc Networks, pages 132–176. Springer-Verlag, 2006. 72, 78, 80

[PLK05] Nam-Shik Park, Kang-Woo Lee, and Hyun Kim. A middleware for supporting context-aware services in mobile and ubiquitous environment. In *Proceedings of the International Conference on Mobile Business (ICMB'05)*, pages 694–697, Sydney, Australia, July 2005. 79

[PPJ$^+$06] Anand Patwardhan, Filip Perich, Anupam Joshi, Tim Finin, and Yelena Yesha. Querying in packs: Trustworthy data management in ad hoc networks. *International Journal of Wireless Information Networks*, 13(4):263–274, October 2006. 80

[PS97] Evaggelia Pitoura and George Samaras. *Data Management for Mobile Computing (Advances in Database Systems)*. Springer, 1997. 72

[PSTM04] Dimitris Papadias, Qiongmao Shen, Yufei Tao, and Kyriakos Mouratidis. Group nearest neighbor queries. In *Proceedings of the 20th International Conference on Data Engineering (ICDE'04)*, pages 301–312, Boston, Massachusetts, USA, March 2004. 86, 87, 90

[RBB03] Kurt Rothermel, Martin Bauer, and Christian Becker. *Digitale Weltmodelle - Grundlage kontextbezogener Systeme*, chapter 5, pages 123–141. Springer-Verlag, Berlin, Heidelberg, New York, May 2003. German. 39, 40

[RD01] Antony I. T. Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms*, pages 329 – 350, Heidelberg, Germany, November 2001. Springer-Verlag. 88

[RDD$^+$03] Kurt Rothermel, Dominique Dudkowski, Frank Dürr, Martin Bauer, and Christian Becker. Ubiquitous computing - more than computing anytime anyplace? In Dieter Fritsch, editor, *Proceedings of Photogrammetric Week '03*, pages 1–9, Stuttgart, 2003. 39, 40

[REF$^+$06] Kurt Rothermel, Thomas Ertl, Dieter Fritsch, Paul J. Kühn, Bernhard Mitschang, Engelbert Westkämper, Christian Becker, Dominique Dudkowski, Andreas Gutscher, Christian Hauser, Lamine Jendoubi, Daniela Nicklas, Steffen Volz, and Matthias Wieland. SFB 627 - Umgebungsmodelle für mobile kontextbezogene Systeme. *Informatik Forschung und Entwicklung*, 21:105–113, June 2006. German. 40

[RJM$^+$06] Alexander Roßnagel, Silke Jandt, Jürgen Müller, Andreas Gutscher, and Jessica Heesen. *Datenschutzfragen mobiler kontextbezogener Systeme*. DuD-Fachbeiträge. Deutscher Universitäts-Verlag, Wiesbaden, October 2006. German. 41

[RKS$^+$03] Sylvia Ratnasamy, Brad Karp, Scott Shenker, Deborah Estrin, Ramesh Govindan, Li Yin, and Fang Yu. Data-centric storage in sensornets with GHT, a geographic hash table. *Mobile Networks and Applications*, 8(4):427–442, August 2003. 73, 81, 100

[RKV95] Nick Roussopoulos, Stephen Kelley, and Frédéric Vincent. Nearest neighbor queries. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data (SIGMOD'95)*, pages 71–79, San Jose, California, USA, May 1995. 86, 87

[RKY$^+$02] Sylvia Ratnasamy, Brad Karp, Li Yin, Fang Yu, Deborah Estrin, Ramesh Govindan, and Scott Shenker. GHT: a geographic hash table for data-centric storage. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, pages 78–87, Atlanta, Georgia, USA, September 2002. 10, 11, 13, 73, 81, 84, 85, 86, 93, 95, 99, 109, 110, 124, 125, 139

[RPSS03] Ananth Rao, Christos Papadimitriou, Scott Shenker, and Ion Stoica. Geographic routing without location information. In *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking*, pages 96–108, San Diego, California, USA, September 2003. ACM Press. 91

[SH03] Karim Seada and Ahmed Helmy. Rendezvous regions: A scalable architecture for service location and data-centric storage in large-scale wireless networks. Technical report, University of Southern California, Los Angeles, California, USA, July 2003. 83, 84, 99

[SH04] Karim Seada and Ahmed Helmy. Rendezvous regions: A scalable architecture for service location and data-centric storage in large-scale wireless networks. In

Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS'04), page 218a, Santa Fe, New Mexico, USA, April 2004. 83, 84, 85, 86, 99, 184

[SHB+03] Illya Stepanov, Jörg Hähner, Christian Becker, Jing Tian, and Kurt Rothermel. A meta-model and framework for user mobility in mobile networks. In *Proceedings of the 11th IEEE International Conference on Networks (ICON'03)*, pages 231–238, Sydney, Australia, September 2003. 58

[SIG+04] Thomas Schwarz, Markus Iofcea, Matthias Grossmann, Nicola Hönle, Daniela Nicklas, and Bernhard Mitschang. On efficiently processing nearest neighbor queries in a loosely coupled set of data sources. In *Proceedings of the 12th Annual ACM International Workshop on Geographic Information Systems (GIS'04)*, pages 184–193, Washington, DC, USA, November 2004. 86, 87

[SR01] Zhexuan Song and Nick Roussopoulos. K-nearest neighbor search for moving query point. In *Proceedings of the 7th International Symposium on Advances in Spatial and Temporal Databases (SSTD'01)*, pages 79–96, Redondo Beach, California, USA, July 2001. 86, 87

[SRK+03] Scott Shenker, Sylvia Ratnasamy, Brad Karp, Ramesh Govindan, and Deborah Estrin. Data-centric storage in sensornets. *Computer Communication Review*, 33(1):137–142, January 2003. 73, 81

[ST94] Bill N. Schilit and Marvin M. Theimer. Disseminating active map information to mobile hosts. *IEEE Network*, 8(5):22–32, 1994. 39

[SWS+04] Carl-Fredrik Sørensen, Maomao Wu, Thirunavukkarasu Sivaharan, Gordon S. Blair, Paul Okanda, Adrian Friday, and Hector Duran-Limon. A context-aware middleware for applications in mobile ad hoc environments. In *Proceedings of the 2nd Workshop on Middleware for Pervasive and Ad-Hoc Computing (MPAC'04)*, pages 107–110, Toronto, Canada, October 2004. 80

[TDC01] Damla Turgut, Sajal K. Das, and Mainak Chatterjee. Longevity of routes in mobile ad hoc networks. In *Proceedings of the IEEE Semiannual Vehicular Technology Conference (VTC Spring'01)*, volume 4, pages 2833–2837, Rhodes, Greece, May 2001. 149, 151

[TDP+94] Douglas B. Terry, Alan J. Demers, Karin Petersen, Mike J. Spreitzer, Marvin M. Theimer, and Brent B. Welch. Session guarantees for weakly consistent replicated data. In *Proceedings of the Third International Conference on Parallel and Distributed Information Systems (PDIS'94)*, pages 140–149, Austin, Texas, USA, September 1994. 92

[THB+02] Jing Tian, Jörg Hähner, Christian Becker, Illya Stepanov, and Kurt Rothermel. Graph-based mobility model for mobile ad hoc network simulation. In *Proceedings of the 35th Annual Simulation Symposium (ANSS'02)*, pages 337–344, San Diego, California, USA, April 2002. 58

[TNK04] Ravinder Tamishetty, Lek Heng Ngoh, and Pung Hung Keng. An efficient resiliency scheme for data centric storage in wireless sensor networks. In *Proceedings of the IEEE 60th Vehicular Technology Conference (VTC Fall'04)*, volume 4, pages 2936–2940, Los Angeles, California, USA, September 2004. 83, 99

[TS06] Andrew S. Tanenbaum and Maarten Van Steen. *Distributed Systems Principles and Paradigms*. Prentice Hall, second edition, October 2006. 92

[TV04] Jivodar B. Tchakarov and Nitin H. Vaidya. Efficient content location in wireless ad hoc networks. In *Proceedings of the 2004 IEEE International Conference on Mobile Data Management (MDM'04)*, pages 74–85, Berkeley, California, USA, January 2004. 86, 87

[WCCC07] Shan-Hung Wu, Kun-Ta Chuang, Chung-Min Chen, and Ming-Syan Chen. DIKNN: An itinerary-based KNN query processing algorithm for mobile sensor networks. In *IEEE 23rd International Conference on Data Engineering (ICDE'07)*, pages 456–465, Istanbul, Turkey, April 2007. 86, 88

[Wei91] Mark Weiser. The computer for the twenty-first century. *Scientific American Special Issue on Communications, Computers, and Networks*, pages 94–104, September 1991. 38

[WL04] Julian Winter and Wang-Chien Lee. KPT: A dynamic KNN query processing algorithm for location-aware sensor networks. In *Proceedings of the First Workshop on Data Management for Sensor Networks (DMSN'04)*, Toronto, Canada, August 2004. 86, 87

[WSCY99] Ouri Wolfson, A. Prasad Sistla, Sam Chamberlain, and Yelena Yesha. Updating and querying databases that track mobile units. *Distributed and Parallel Databases*, 7(3):257–387, July 1999. 89

[Wu05] Xiaoxin Wu. VPDS: Virtual home region based distributed position service in mobile ad hoc networks. In *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*, pages 113–122, Columbus, Ohio, USA, June 2005. 85

[WXL05] Julian Winter, Yingqi Xu, and Wang-Chien Lee. Energy efficient processing of K nearest neighbor queries in location-aware sensor networks. In *Proceedings of the Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'05)*, pages 281–292, San Diego, California, USA, July 2005. 86, 87

[XFLW07] Yingqi Xu, Tao-Yang Fu, Wang-Chien Lee, and Julian Winter. Itinerary-based techniques for processing K nearest neighbor queries in location-aware sensor networks. *Signal Processing*, 87(12):2861–2881, December 2007. 86, 87, 88

[XL03] Yingqi Xu and Wang-Chien Lee. Window query processing in highly dynamic geo-sensor networks: Issues and solutions. In *Proceedings of NSF Workshop Geo Sensor Networks (GSN'03)*, Portland, Maine, USA, October 2003. 86, 88

[YLG03] Dan Yu, Hui Li, and Ingo Gruber. Path availability in ad hoc network. In *Proceedings of the 10th International Conference on Telecommunications (ICT'03)*, volume 1, pages 383–387, Papeete, French Polynesia, February 2003. 151

[YLN03] Jungkeun Yoon, Mingyan Liu, and Brian Noble. Random waypoint considered harmful. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'03)*, San Francisco, California, USA, March 2003. 58

[Zah06] Thomas Christian Zahn. *Structured Peer-to-Peer Services for Mobile Ad Hoc Networks*. Dissertation, Freie Universität Berlin, Berlin, Germany, July 2006. 88

[ZS05] Thomas Zahn and Jochen Schiller. MADPastry: A DHT substrate for practicably sized MANETs. In *Proceedings of the 5th Workshop on Applications and Services in Wireless Networks (ASWN'05)*, Paris, France, June 2005. 88

[ZSLS06] Changxi Zheng, Guobin Shen, Shipeng Li, and Scott Shenker. Distributed segment tree: Support of range query and cover query over DHT. In *Proceedings of the 5th International Workshop on Peer-to-Peer Systems (IPTPS'06)*, Santa Barbara, California, USA, February 2006. 88

[ZWS06] Thomas Zahn, Georg Wittenburg, and Jochen Schiller. Towards efficient range queries in mobile ad hoc networks using DHTs. In *Proceedings of the 1st International Workshop on Decentralized Resource Sharing in Mobile Computing and Networking*, pages 72–74, Los Angeles, California, USA, September 2006. 86, 88

Index