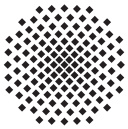


Universität Stuttgart

Institut für Parallele und Verteilte Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit No. 3638

Platzierung und Migration von CEP-Operatoren in MANet Szenarien

Tycho Bomancz

Studiengang:	Informatik
Prüfer/in:	Prof. Dr. Kurt Rothermel
Betreuer/in:	Dipl.-Inf. Beate Ottenwälder
Beginn am:	21. Oktober 2013
Beendet am:	22. April 2014
CR-Nummer:	C.1.4, C.2.1, C.2.2, C.2.4, C.4, D.4.4

Kurzfassung

Mobile Endgeräte haben in den letzten Jahren den Markt durchdrungen. Eine Vielzahl von Kommunikationsschnittstellen ermöglichen drahtlose Verbindungen mit anderen Endgeräten. Zumeist sind mobile Endgeräte zusätzlich mit einer Vielzahl von Sensoren ausgestattet. Systeme zur komplexen Ereignisverarbeitung ermöglichen es, Nutzer gezielt Informationen von Interesse zukommen zu lassen, ohne dass sie fortlaufend Abfragen ausführen müssen. Herkömmliche Systeme zur komplexen Ereignisverarbeitung führen Korrelationen in der Infrastruktur aus, durch steigende Prozessorleistungen ist aber auch ein vollständiges System in einem Ad-hoc-Netzwerk möglich, sodass dieses völlig autonom ohne eine vorhandene Infrastruktur arbeiten kann. Der daraus resultierende Vorteil ist die Unabhängigkeit von einer eventuell nicht vorhandenen Infrastruktur. Im Gegensatz zur Beständigkeit der Topologie in einer Infrastruktur bewegen sich mobile Knoten. Die Strecken, die ein Datenpaket durchläuft, können sich in Folge dieser Variation deutlich verlängern. Diese Arbeit soll Methoden einführen, die Operatoren in einem Ad-hoc-Szenario, in welchem keine Infrastruktur für eine Ereigniskorrelation zur Verfügung steht, mit dem Ziel zwischen mobilen Endgeräten migriert, den Energiebedarf zu optimieren und die Belegung des geteilten Kommunikationskanals und damit die zurückgelegte Strecke auf ein Minimum zu reduzieren.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Algorithmen

1 Einleitung

Viele Katastrophensituationen, beispielsweise das Tsunami-Unglück im Indischen Ozean 2004, das Tohoku-Erdbeben 2011 oder das Erdbeben in den Philippinen 2013 haben gezeigt, wie wichtig Kommunikation für Menschen ist. Sie ermöglicht es, gemeinsam Hilfe zu leisten oder gar weiteres Unheil vermeiden zu können. Katastrophenhelfer und Opfer sind in solchen Zeiten überfordert und müssen kontinuierlich über den Verlauf und die Auswirkungen von Folgekatastrophen informiert werden.

Komplexe Ereignisverarbeitung ist ein Paradigma, das in solchen Situation Hilfestellung und Unterstützung geben kann, um Menschen gezielt über Verläufe von Katastrophen zu informieren und ihnen so die Arbeit, Betroffene zu unterrichten, abnimmt. Die komplexe Ereignisverarbeitung erhält die von Sensoren erzeugten Ereignisse als Ereignisstrom, welcher von Operatoren direkt ausgewertet und den Konsumenten zugestellt wird. Ein Operator führt in einem System zur komplexen Ereignisverarbeitung Korrelationen aus. Im Falle von Erdbebenkatastrophen summiert ein Operator beispielsweise die Anzahl von mobilen Knoten, die Erschütterungen melden, um die Starke des Erdbebens abschätzen zu können. Naturereignisse, wie Nachbeben, können durch komplexe Ereignisverarbeitung erkannt werden, indem Operatoren Sensordaten von mobilen Knoten analysieren. In [LOBC12] wurden Mustererkennungen erörtert, indem Erschütterungen von einer großen Zahl von Knoten korreliert werden und die Menschen in den betreffenden Regionen schnellstmöglich Warnungen erhalten.

Herkömmliche komplexe Ereignisverarbeitung wird in einer dedizierten Infrastruktur ausgeführt. Eine Infrastruktur kann während einer Naturkatastrophe beschädigt werden oder ausfallen, das Ausbleiben von Warnungen für die betreffenden Regionen ist die Folge. Mobile Knoten haben die Möglichkeit, über drahtlose Kommunikationskanäle mit anderen mobilen Knoten Verbindungen herzustellen. Sie werden häufig durch Smartphones und in naher Zukunft auch durch Infotainmentsysteme von Automobilen repräsentiert. Diese Verbindungen sind nicht von einer Infrastruktur abhängig, sodass auch in einem Katastrophengebiet die Kommunikation aufrecht erhalten werden kann. Arbeiten auf dem Gebiet der komplexen Ereignisverarbeitung haben bereits gezeigt, dass mobile Knoten nicht auf die Ausführung von Sensoren oder Konsumenten beschränkt sind, sondern durch die steigende Prozessorleistung solcher Knoten auch die Ausführung von Operatoren sinnvoll möglich ist [Vet12]. Dies eröffnet die Möglichkeit, komplexe Ereignisverarbeitung vollständig in einem mobilen Ad-hoc-Netzwerk auszuführen, in welchem die teilnehmenden Knoten direkt miteinander kommunizieren und das System auf ihnen ausgeführt wird, ohne auf eine Infrastruktur angewiesen zu sein. Mit steigender Knotendichte weist ein so ausgeführtes System eine hohe Zusammenhangskomponente auf. Diese Eigenschaft ist für Krisengebiete von hoher Bedeutung.

Die Nutzung eines mobilen Ad-hoc-Netzwerkes und mobiler Knoten stellt neue Anforderungen an Systeme der komplexen Ereignisverarbeitung. Während in Infrastrukturnetzwerken das Routing von der Infrastruktur übernommen wird, sind in Ad-hoc-Netzwerken alle Knoten mit gleichen Pflichten

am Routing beteiligt. Mobile Knoten werden üblicherweise durch eine ortsunabhängige Energiequelle gespeist. Zudem muss die Netzwerkkapazität in Ad-hoc-Netzwerken von allen teilnehmenden mobilen Knoten geteilt werden und steht somit deutlich begrenzter [CCG00] zur Verfügung als in kabelgebundenen Netzwerken. Häufige Topologieänderungen durch sich bewegende Knoten stellen hohe Optimierungsanforderungen sowohl an Routingalgorithmen als auch an den Algorithmus zur Operatorplatzierung, damit mit Energie sinnvoll umgegangen wird und so zu einer langen Laufzeit von mobilen Knoten beigetragen wird.

Ein Algorithmus muss die Platzierung von CEP-Komponenten so vornehmen, dass möglichst wenig mobile Knoten bei der Vermittlung der Ereignisse benötigt werden und das geteilte Medium dadurch nur in einem solchen Umfang genutzt wird, wie es erforderlich ist. In dieser Arbeit werden Methoden aufgezeigt, die durch gezielte Neuevaluierung der aktuellen Operatorplatzierung prüfen, ob eine Migration auf einen anderen Knoten aufgrund der geänderten Topologie erforderlich geworden ist. Um das Datenvolumen, welches über den drahtlosen Kommunikationskanal transferiert wird, zu reduzieren, arbeiten diese Methoden über lokale Entscheidungen und ohne globales Wissen. Die Entscheidungen werden dabei auf Basis von Daten getroffen, die durch das im Ad-hoc-Netzwerk notwendige Routing bereits verfügbar sind.

Gliederung

Diese Arbeit ist in folgende Kapitel eingeteilt:

Das Kapitel 2 beschreibt die Grundlagen zu den Themen komplexe Ereignisverarbeitung, Ad-hoc-Netzwerken, geht auf den Energiebedarf von WiFi (802.11) ein und erklärt die verschiedenen Ansätze für Routing in Ad-hoc-Netzwerken. Kapitel 3 definiert die Entitäten, die in den Systemen, die in dieser Arbeit behandelt werden, enthalten sind. Außerdem wird das Modell des mobilen Ad-hoc-Netzwerks und das Energiemodell beschrieben, welches in dieser Arbeit angewandt wird. Kapitel 4 führt die Problemstellung ein, die bei der Nutzung von komplexer Ereignisverarbeitung in einem mobilen Ad-hoc-Netzwerk entsteht. Kapitel 5 gibt einen Überblick über die Architektur und die Kommunikation des eingeführten Frameworks mit dem umliegenden System. In Kapitel 6 werden die Methodiken eingeführt, die im Rahmen dieser Arbeit zur Lösung der Problemstellung entwickelt wurden. Kapitel 7 zeigt Implementierungsdetails. In Kapitel 8 werden die Simulationsergebnisse gezeigt und diskutiert. Auf eine Auswahl verwandter Arbeiten wird in Kapitel 9 eingegangen. Eine Zusammenfassung und ein Ausblick finden sich in Kapitel 10.

2 Grundlagen

In diesem Kapitel wird das Paradigma der komplexen Ereignisverarbeitung vorgestellt, danach wird auf mobile Ad-hoc-Netzwerke und ihre Unterschiede zu Infrastruktur-Netzwerken eingegangen.

2.1 Komplexe Ereignisverarbeitung

Herkömmliche Systeme, wie *relationale Datenbankmanagementsysteme (folgend kurz RDBMS genannt)*, sind konzipiert worden, um dem Benutzer auf eine einmal gestellte Frage zu antworten. Solche Systeme sind interessant für Anwendungen wie Kunden- und Inventarstämme. Dabei werden in der Vergangenheit abgespeicherte Zustände für die Beantwortung der Anfrage herangezogen. Vergleiche erfordern das Laden und Durchsuchen von alten Datensätzen, weswegen die Erkennung von Mustern zwischen alten und aktuellen Datenbeständen Zeit kostet. Das Konzept der *komplexen Ereignisverarbeitung (folgend kurz CEP genannt - Englisch für complex event processing)* setzt an dieser Stelle an. CEP arbeitet nicht wie ein herkömmliches RDBMS auf Anfragen des Nutzers, sondern generiert selbstständig Ereignisse, sobald etwas definiertes passiert. Die generierten Ereignisse werden direkt nach der Erzeugung durch das System verarbeitet und einem Konsumenten zur Verfügung gestellt. Dies erlaubt eine effiziente Beobachtung und Auswertung von Echtzeitdaten, wie es mit einem RDBMS nicht möglich ist [See].

In Abbildung 2.1 wird gezeigt, wie komplexe Ereignisverarbeitung verteilt ausgeführt werden kann. Teilnehmer, die innerhalb des CEP-Systems als Sensoren betrieben werden, erstellen Daten, die von anderen Teilnehmern gesammelt und nach einer nach bestimmten Mustern stattfindenden Verarbeitung an Konsumenten gesendet werden, die an den entstandenen Daten Interesse haben.

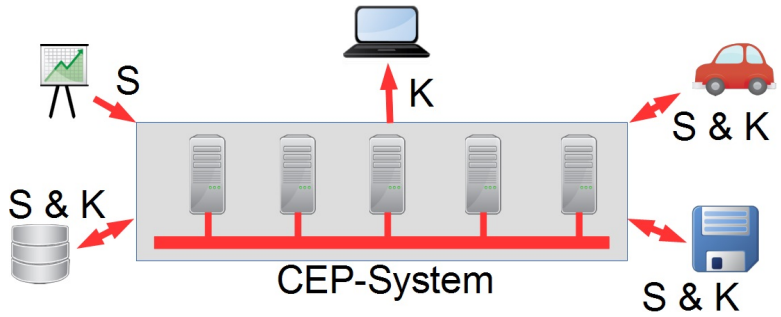


Abbildung 2.1: Schema eines CEP-Systems

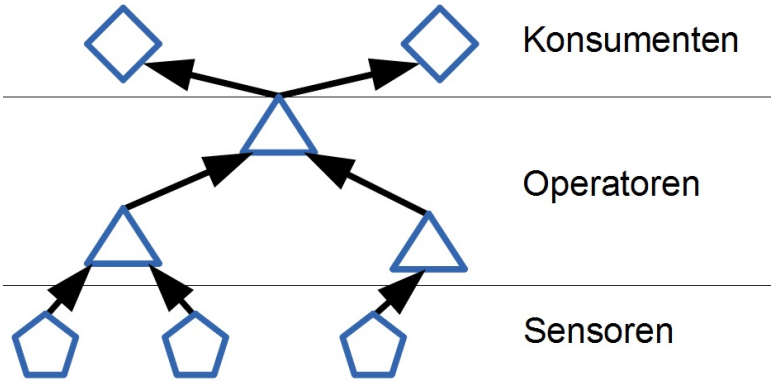


Abbildung 2.2: Ein Operatorgraph nach [OKRR13]

Durch Staffelung von Operatoren innerhalb des CEP-Systems können Daten, die von einer Vielzahl von Sensoren gesammelt werden, stufenweise in das gewünschte Endprodukt überführt werden, indem sie nicht nur einen, sondern mehrere Operatoren durchlaufen, bevor sie den Konsumenten zur Verfügung gestellt werden. Teilnehmer sind nicht auf einen Typen innerhalb des CEP-Systems beschränkt, sondern können, wie beispielhaft in Abbildung 2.1 gezeigt wird, auch mehrere Typen gleichzeitig repräsentieren. S steht in Abbildung 2.1 für Sensor, K für Konsument.

2.1.1 Operatorgraphen

In viele Arbeiten, wie in [OKRR13] oder in [KKR10], wird das Konzept des Operatorgraphs verwendet. Ein solcher Operatorgraph ist ein azyklischer Graph, der aus Sensoren, Operatoren, Konsumenten und den sie verbindenden Ereignisströmen besteht. Er zeigt den logischen (das heißt nicht physikalischen) Weg, den ein durch ein Sensor erzeugtes Ereignis innerhalb des CEP-Systems durch einen oder mehrere Operatoren nimmt, bevor es einem oder mehreren Konsumenten zugestellt wird.

Ein Operatorgraph lässt sich, wie in Abbildung 2.2, in mehrere Schichten aufteilen. In der ersten Schicht finden sich Sensoren, die Ereignisse produzieren. Die folgende Schicht besteht aus Operatoren, die von Sensoren oder anderen Operatoren Ereignisse erhalten und diese verarbeiten. In der letzten Schicht finden sich die Konsumenten, welche Ereignisse von Operatoren erhalten.

2.1.2 Ereignisse und Ereignisströme

In einem CEP-System treten zwei Typen von Ereignissen auf:

- **einfache** Ereignisse werden von Sensoren produziert und an Operatoren versendet.
- **komplexe** Ereignisse entstehen durch Verarbeitung anderer Ereignisse. Solche Ereignisse sind bereits mindestens einmal von einem Operator verarbeitet worden.

Eine Folge von Ereignissen wird Ereignisstrom genannt. Ein Ereignisstrom hat immer einen definierten Erzeuger und einen definierten Empfänger. Befindet sich am Ende eines Ereignisstroms ein Operator, ist es möglich, Ereignisse zu puffern. Ereignisse können von Operatoren nach einem vom Administrator des CEP-Systems frei definierbaren Muster, wie zum Beispiel first-in-first-out, bearbeitet werden. Die Pufferung von Ereignissen kann beispielsweise genutzt werden, um bei einer Korrelation die jeweils letzten drei Ereignisse heranziehen zu können. [CM12]

2.1.3 Ereignisproduzenten und -konsumenten

Komponenten eines CEP-Systems können in eine oder beide der folgenden Kategorien eingeteilt werden:

- **Ereignisproduzenten** produzieren Ereignisse, die für andere CEP-Komponenten bestimmt sind.
- **Ereigniskonsumenten** empfangen Ereignisse von anderen CEP-Komponenten.

2.1.4 Sensoren in einem CEP-System

Sensoren produzieren in einem CEP-System einfache Ereignisse und leiten diese an Operatoren (oder Konsumenten) weiter. Hierbei können alle möglichen Sensoren oder Stati eines Geräts oder seiner Umgebung als Datenquellen genutzt werden, wie die GPS-Koordinaten, die Geschwindigkeit oder den Bremspedal- und Drosselklappenstand eines Kraftfahrzeugs.

Sensoren produzieren **einfache** Ereignisse und werden ausschließlich den **Ereignisproduzenten** zugeordnet, da Sensoren keine Ereignisse empfangen. In Abbildung 2.2 befinden sich die Sensoren in der unteren Schicht.

2.1.5 Operatoren in einem CEP-System

Operatoren sind Komponenten im CEP-System, die eine für jeden Operator individuelle Anzahl an Ereignisströmen empfangen und selbst produzieren. Die eingehenden Ereignisströme stammen aus unterschiedlichen Quellen und beinhalten unterschiedliche Ereignisse, wohingegen ausgehende Ereignisströme an unterschiedliche Empfänger gerichtet sind, aber alle den gleichen Inhalt aufweisen. Operatoren generieren ausgehende Ereignisse durch Korrelation, Aggregation, Filterung oder einer anderen, vorher festgelegten, Funktion. Sie generieren definitionsgemäß **komplexe** Ereignisse. Ereignisse, die durch einen Operator empfangen wurden, können dabei selbst **einfache** oder **komplexe** Ereignisse sein, also vorher bereits andere Operatoren durchlaufen haben.

In frühen CEP-Systemen wurden Operatoren ausschließlich in der Infrastruktur, also nicht auf mobilen Endgeräten, ausgeführt. Vorangegangene Arbeiten [Vet12] haben gezeigt, dass durch die voranschreitende Entwicklung von Prozessoren auch auf solchen Geräten eine Ausführung sinnvoll möglich ist.

Operatoren produzieren und empfangen Ereignisse und werden somit den **Ereignisproduzenten** und den **Ereigniskonsumenten** zugeordnet. Sie stehen, wie es in Abbildung 2.2 ersichtlich ist, in der Mitte der Verarbeitungskette.

2.1.6 Konsumenten in einem CEP-System

Konsumenten sind Endpunkte, an welchen die Verarbeitung der gelieferten Information aus der Ereignisverarbeitung erfolgt. Dies kann durch einfache Darstellung auf einem Display, dem Auslösen einer bestimmten Reaktion (wie eine Abschaltung des Tempomaten bei vorausgehendem Stau) oder Speicherung auf einem persistenten Speicher geschehen.

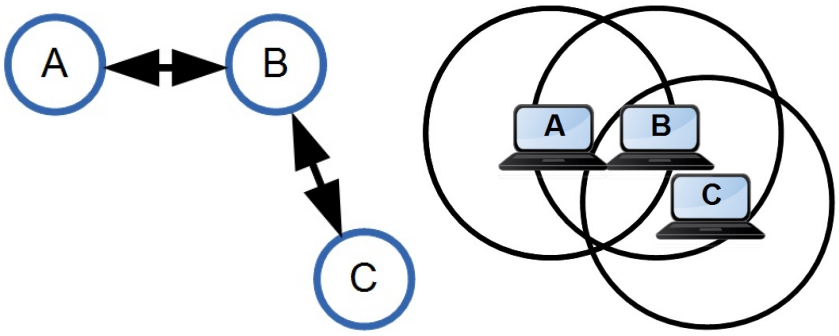
Konsumenten empfangen normalerweise **komplexe** Ereignisse, die mittels Operatoren aufgearbeitet wurden. Prinzipiell können sie aber auch **einfache** Ereignisse empfangen. Sie produzieren selbst keine Ereignisse und werden daher ausschließlich den **Ereigniskonsumenten** zugeordnet. Folglich sind sie das Ende eines Operatorgraphen, wie auch in Abbildung 2.2 ersichtlich ist.

2.2 Ad-hoc-Netzwerke

Durch Kombination von drahtlosen Kommunikationstechnologien mit mobilen Endgeräten entsteht die Möglichkeit, diese an jedem Ort miteinander zu vernetzen, ohne dass eine Infrastruktur erforderlich ist. Ein solches Netzwerk wird MANet genannt (mobile Ad-hoc Network) [BRG99].

In einem Infrastruktur-Netzwerk kommunizieren die Knoten über ein zentrales System über einen drahtgebundenen Kanal. Dieses System übernimmt neben dem Transport gleichzeitig auch die Routing-Entscheidungen für Pakete, die von einem teilnehmenden Endgerät zu einem anderen Endgerät versendet werden. Ein Infrastruktur-Netzwerk kann ebenfalls mit drahtlosen Komponenten betrieben werden, wie es im Infrastruktur-Modus von WiFi (802.11) realisiert ist. Die teilnehmenden Knoten kommunizieren dann drahtlos mit einem Zugangspunkt, der die Knoten mit dem drahtgebundenen System verbindet. Der Aufbau eines solchen Netzwerks erfordert eine gewisse Planung im Hinblick auf die Ausgestaltung der Infrastruktur und es entsteht eine Abhängigkeit bezüglich dieser. Fallen Teile oder das gesamte Infrastruktur-Netzwerk aus, ist für alle betroffenen Knoten keine Kommunikation mit anderen Knoten mehr möglich, die sich in anderen Partitionen des Infrastrukturnetzwerks befinden. Dieses Szenario wiederholt sich regelmäßig in Katastrophengebieten.

In einem MANet existiert keine ausfallgefährdete Infrastruktur, an welche ein teilnehmender Knoten angeschlossen wird. Die Knoten verbinden sich über einen Funkkanal - meist WiFi - mit ihrer Nachbarschaft. Die Nachbarschaft definiert sich über die Reichweite eines Funkkanals, die von den physikalischen Eigenschaften und der Umgebung, in welcher das Netzwerk sich befindet, abhängt. Die Umgebung kann vielerlei Auswirkungen auf den Funkkanal zeigen: Multipfad-Ausbreitung, Abschattung, Brechung, Reflexion und frequenzabhängige Freiraumdämpfung [Sch03]. Dadurch ist die Zuverlässigkeit des Kommunikationskanals nicht so hoch, wie bei Infrastrukturnetzen, die als kabelgebundene Netzwerke ausgeführt werden, sodass eine Mehrfachübertragung eines Datenpakets über einen Funkkanal häufiger vorkommt. Im Gegenzug können äußere Einflüsse, wie ein Erdbeben, die Kommunikation nicht durch Zerstörung von Landleitungen beeinflussen.



(a) Graphdarstellung **(b)** Darstellung mit Reichweite

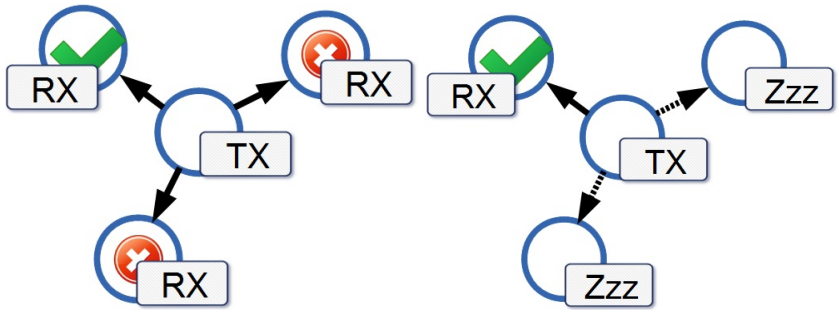
Abbildung 2.3: Mobile Knoten kommunizieren direkt oder indirekt in einem Ad-hoc-Netzwerk

Durch die Verteilung der Knoten und die verschiedenen Reichweiten resultiert ein Netzwerk, in welchem Knoten mit ihren Nachbarn kommunizieren können, wie es in Abbildung 2.3 gezeigt wird, wobei die einzelnen Knoten durch Kreise dargestellt werden. Knoten A und Knoten B können, genau wie Knoten B und Knoten C, über den drahtlosen Funkkanal kommunizieren. Anders, als in einem Infrastruktur-Netzwerk, können Knoten A und Knoten C nicht durch eine Infrastruktur miteinander in Kontakt treten. Durch Routingverfahren, die sogenannte Multihop-Verbindungen realisieren, können A und C durch andere Knoten kommunizieren, indem die zwischenliegenden Knoten die Datenpakete vermitteln. Im Gegensatz zu Infrastruktur-Netzwerken sind die Knoten eines MANet gleichberechtigt an Routingentscheidungen beteiligt. Routingalgorithmen, die in MANets eingesetzt werden, müssen mit der Dynamik, wie sie durch mobile Knoten möglich ist, umgehen können und schnell an die aktuelle Topologie konvergieren, um Pakete, die auf dem Weg sind, zustellen zu können. Ein Paket, das wegen nicht aktueller Routingtabellen nicht zugestellt werden kann, muss sonst erneut übermittelt werden. In MANets existieren von einem Knoten zu einem anderen Knoten in den meisten Fällen mehrere Wegmöglichkeiten, sodass eine hohe Redundanz gegeben ist, um mit Knotenausfällen wirksam umzugehen.

In den meisten Fällen werden mobile Endgeräte durch Akkumulatoren gespeichert. Akkumulatoren können nur in begrenzter Menge Energie aufnehmen, weshalb die Lebensdauer eines mobilen Knotens unmittelbar davon abhängt, wie er mit der Energie, die ihm zur Verfügung steht, umgeht. Um sinnvoll mit dieser umzugehen, sollten Algorithmen, die auf mobilen Endgeräten ausgeführt werden, so gestaltet sein, dass sie möglichst effizient mit den verfügbaren Mitteln arbeiten. Routingalgorithmen, die Multihop-Routing realisieren, sollten die Nachricht auf direktem Weg zum Empfänger leiten, damit so wenig Knoten wie möglich an der Weitergabe beteiligt sind.

2.3 Energiebedarf von WiFi (802.11)

In dieser Arbeit sollen Verbindungen innerhalb des CEP-Systems durch ein mobiles Ad-hoc-Netzwerk betrachtet werden. In dieser Arbeit wird das mobile Ad-hoc-Netzwerk durch WiFi modelliert, weswegen hier darauf eingegangen wird.



(a) Verwerfen des Datenpaketes durch nicht adressierte Nachbarn

(b) Ignorieren des Datenpaketes durch nicht adressierte Nachbarn

Abbildung 2.4: Verschiedene Verfahren zur Zustellung eines Datenpaketes an Nachbarn

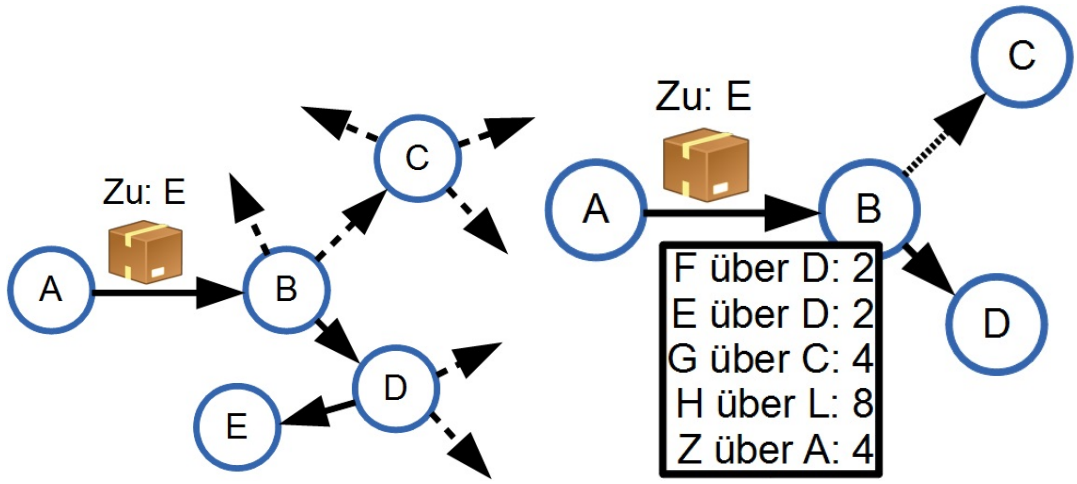
In mobilen Umgebungen werden Knoten im Normalfall von einer ortsunabhängigen, aber limitierten Energiequelle gespeist. Die Anwendung des drahtlosen Funkkanals entzieht dieser Quelle gespeicherte Energie, sodass die Zeit, in welcher der mobile Knoten aktiv ist, auch durch die Nutzung des Funkkanals mitbestimmt wird.

Messungen [FN01] zeigen, dass abhängig vom Betriebszustand unterschiedlicher Energiebedarf für das Netzwerkinterface besteht. Nicht nur das Senden, auch der Empfang kostet Energie, die der mobilen Quelle entzogen wird. Absolute Werte finden sich in Tabelle 2.1.

Das bedeutet, dass ein Knoten mit einer Nachbarschaft von n anderen Knoten neben dem eigenen Energiebedarf, der durch das Senden von Daten entsteht, zusätzlich einen Energiebedarf von n mal dem Empfangsmodus auf den Knoten in der Nachbarschaft auslöst, obwohl n - 1 Nachbarn nicht am Empfang dieser Daten interessiert sind, wie es in Abbildung 2.4a gezeigt wird. Zur Optimierung werden Knoten, die eine Sendung empfangen können, die jedoch nicht an sie adressiert ist, in einen Schlafmodus versetzt, der weniger Energie verbraucht als der Modus, in welchem das Datenpaket empfangen wird. Die dafür benötigte Zeit kann durch die Größe des zu übertragenden Datenpakets, welche am Anfang der Sendung angekündigt wird, abgeschätzt werden. Gleichzeitig wird auch der Knoten, an welchen das Datenpaket gerichtet ist, adressiert, sodass alle Knoten, die nicht angesprochen werden, diese Nachricht nicht empfangen werden. Dies wird in Abbildung 2.4b verdeutlicht. [CSAK98]

Datenrate	Modus	Energieverbrauch (gemessen)
2 Mbps	Schlafmodus	14 mA
2 Mbps	Empfangsmodus	200 mA
2 Mbps	Sendemodus	280 mA
11 Mbps	Schlafmodus	10 mA
11 Mbps	Empfangsmodus	190 mA
11 Mbps	Sendemodus	284 mA

Tabelle 2.1: Energiebedarf einer Lucent IEEE 802.11 DSSS PC Card, entnommen aus [FN01]



(a) Zustellung eines Paketes durch Fluten **(b)** Zustellung eines Paketes durch Routing

Abbildung 2.5: Verschiedene Verfahren zur Zustellung eines Datenpaketes in Netzwerken

2.4 Routing in Ad-hoc-Netzwerken

Um mit der Dynamik eines mobilen Ad-hoc-Netzwerkes umgehen zu können, sind Routingalgorithmen erforderlich, die mit der sich schnell ändernden Topologie umgehen können. Es gibt verschiedene Ansätze, die abhängig vom Einsatzgebiet des mobilen Ad-hoc-Netzwerkes gegeneinander abgewogen werden können.

- Fluten
- Reaktive Protokolle
- Proaktive Protokolle

Bei Routingalgorithmen, die nicht auf Fluten basieren und eingegangene Pakete auf Basis lokaler Informationen weitergeleitet werden, werden sogenannte Routingtabellen aufgebaut. In diesen Routingtabellen werden, wie es in Abbildung 2.5b gezeigt wird, die von einem Knoten aus erreichbaren Ziele, die Entfernung in Hops zu den genannten Zielen sowie der Nachbarknoten, an welchen der Knoten die Nachricht an den Empfänger weiterleiten muss, um das Ziel zu erreichen, gespeichert.

Der Unterschied zwischen solchen Routingalgorithmen und Fluten wird in Abbildung 2.5 veranschaulicht: Während in 2.5a die Pakete über alle Knoten verteilt werden, obwohl das Ziel E das Paket bereits erhalten hat, wird in 2.5b das Paket nur über die Verbindung BD übertragen. Dort wird erneut eine solche Entscheidung getroffen. Das Paket wird so lange nach diesem Prinzip weitergeleitet, bis es am Ziel eintrifft. Eine Routingtabelle, die die Basis für die beschriebene Entscheidung ist, enthält Informationen über alle nur mittelbar durch Nachbarn erreichbaren Knoten (in 2.5b sind das 5 erreichbare Ziele, die nur über Nachbarn erreicht werden können). Globales Wissen wird von den einzelnen Knoten nicht verlangt. Es wird nur eine lokale Entscheidung getroffen, wohin das Paket weitergeleitet werden muss, um es dem Ziel zuzustellen.

2.4.1 Fluten

Der robusteste Ansatz des Routings in einem Ad-hoc-Netzwerk stellt das Fluten des Netzwerkes dar. Fluten stellt die naivste Form des Routings dar, da ein Knoten, der ein Paket erhält, keine Entscheidung über eine bestimmte Weiterleitung trifft, sondern das Paket allen anderen Nachbarn weiterleitet.

Das Paket durchläuft das gesamte erreichbare mobile Netzwerk und wird auch an Knoten gesendet, die nicht an einer erfolgreichen Zustellung beteiligt sind. Es trifft, wenn keine Netzwerkpartitionierung vorliegt, auf mindestens einem Pfad an seinem Zielknoten ein. Dieses Verfahren ist sehr robust, da alle parallel vorhandenen Wege, einen Zielknoten zu erreichen, genutzt werden und findet zudem immer den kürzesten Pfad, da alle Möglichkeiten in Anspruch genommen werden, eine Nachricht zu versenden. Der Hauptnachteil dieses Verfahrens liegt in der Netzwerkauslastung und dem damit in mobilen Ad-hoc-Netzwerken verbundenen Energieaufwand, der für ein erfolgreiches Routing nicht unbedingt erforderlich ist. Im Gegenzug dazu müssen keine Kontrollnachrichten zur Erhaltung von Routingtabellen ausgetauscht werden. Dies kann bei Netzwerken, in denen sehr selten Nachrichten versendet werden, günstig sein.

2.4.2 Reaktive Protokolle

Reaktive Protokolle erstellen Routingtabellen zu Zielknoten, wenn es erforderlich ist, das heißt, sobald Daten von einem Knoten zu einem anderen Knoten übertragen werden sollen.

Nutzdaten werden mit solchen Verfahren nur an Knoten weitergeleitet, die auf dem Weg zum Ziel liegen. Versendete Nutzdaten wandern deswegen nur durch die Teile des Netzwerks, die für die Zustellung erforderlich sind.

Beispiel für reaktive Protokolle: Ad-hoc On-Demand Distance Vector Ein Distanzvektor-Verfahren [Tan03] basiert auf Routingtabellen, die neben dem Ziel auch die Anzahl an Hops, die bis zum Ziel zurückgelegt werden müssen, und den Ausgang, über welchen das Datenpaket weitergeleitet werden muss, um näher ans Ziel zu kommen, speichert. Die Anzahl an Hops ist interessant, um bei Meldungen anderer Knoten eine Entscheidung treffen zu können, ob eine anderen angebotene Route zum gleichen Ziel mit weniger Hops erreicht werden kann. Ist dem so, wird der Eintrag in der Routingtabelle ersetzt.

Für Ad-hoc-Netzwerke wurde das Distanzvektor-Verfahren unter dem Namen Ad-hoc On-Demand Distance Vector modifiziert, um auch in diesen unter energetischen Aspekten sinnvoll funktionieren zu können. Das Verfahren speichert, wie für Distanzvektor-Methoden üblich, eine Routingtabelle mit dem Ziel, der Anzahl an Sprüngen und dem Nachbarn, an den das Paket geleitet werden soll. Routen werden aber nur dann aufrecht erhalten, wenn es notwendig ist. Soll ein Paket von einer Quelle zu einem bestimmten Ziel transportiert werden und es existiert noch keine Route, generiert das Ad-hoc On-Demand Distance Vector Protokoll eine Route-Request-Nachricht und versendet diese über einen Broadcast. Trifft diese Nachricht auf einen Knoten, der einen Weg zum Ziel kennt oder auf das Ziel selbst, generiert dieser eine Route-Reply-Nachricht mit der Antwort. Die Antwort umfasst alle Knoten, die auf dem Weg zum versendenden Knoten liegen. Beim Weiterleiten der Route-Request-Nachrichten wird diese Historie an die Nachricht angehängt, sodass die Route-Reply-Nachricht diesen Weg auch

zurück nehmen kann und ein Broadcast vermieden wird. Trifft die Route-Reply-Nachricht am Knoten ein, der ein Route-Request ausgelöst hat, kann das eigentliche Paket versendet werden.

Durch die je nach Knotendichte bestehende Redundanz des Netzwerkes treffen unterschiedliche Route-Reply-Nachrichten ein. Knoten auf dem Weg der Route-Reply-Nachrichten speichern sich nicht nur das Ziel, sondern können auch ihre eigene Routingtabelle von dieser Information profitieren lassen, indem Hops ebenfalls der Tabelle hinzugefügt werden. Für diese Knoten ist dann für den Fall, dass ein Paket zu ihnen weitergeleitet werden soll, kein Route-Request notwendig.

Knoten mit aktiven Routingtabellen beobachten die Verbindungen zu den Nachbarn, über welche sie die entsprechenden Ziele erreichen können. Fällt ein solcher Nachbar aus, generiert der Knoten eine Route-Error-Nachricht und versendet sie. Knoten, die diesen Knoten als Zwischenschritt nutzen, wissen nun, dass der Zielknoten nicht mehr über diesen Zwischenschritt erreichbar ist und müssen wieder ein Route-Request starten, wenn sie weiterhin Interesse an einer Verbindung haben. [PR99]

Ein Hauptnachteil von reaktiven Routingverfahren liegt in der Latenz, die beim Versenden einer Nachricht eventuell auftreten kann. Existiert keine Route, kann das mehrere Gründe haben: Es wurde noch nie mit dem gewünschten Ziel gearbeitet oder die Route wurde zu lange nicht benutzt und wurde deswegen wieder aus der Routingtabelle entfernt. Folglich muss je nach Netzwerkauslastung eventuell eine Route aufgebaut werden, was zur Konsequenz hat, dass eine Nachricht bis zum Ende des Routenaufbaus zurückgehalten werden muss.

Der Vorteil solcher Routingverfahren in Ad-hoc-Netzwerken liegt darin, dass Routen, die nicht genutzt werden auch nicht aufgebaut und unterhalten werden. Das spart Energie ein, was sich in einer höheren Lebensdauer der mobilen Knoten widerspiegelt. Auch bei wenig versendeten Paketen lässt sich so Energie einsparen. In einem gut ausgelasteten Netzwerk sind dagegen viele Routen bereits aufgebaut, sodass die anfängliche Latenz selten bis gar nicht auftritt. [RT99]

2.4.3 Proaktive Protokolle

Proaktive Protokolle erstellen und unterhalten Routingtabellen zwischen den Knoten des mobilen Ad-hoc-Netzwerkes unabhängig vom Nachrichtenaufkommen im Netzwerk im Hintergrund.

Pakete, die über das Netzwerk versendet werden, passieren nur Knoten, die auf dem Weg von der Quelle zum Empfänger liegen.

Proaktive Protokolle: Optimized Link State Routing Das Link-State-Protokoll [Cis] verteilt Informationen über die gesamte Netztopologie an alle Router, sodass alle die gleiche Sicht auf das Netzwerk haben. Durch den Austausch sogenannter Hello-Pakete ermittelt ein Router die Erreichbarkeit von Nachbarroutern, welche überprüfen, ob diese Änderung bereits für sie bekannt ist. Die Nachricht erreicht durch Fluten das gesamte Netzwerk. Bei Änderungen der Netzwerktopologie werden die optimalen (oft die kürzesten) Pfade nach einer definierten Metrik gesucht. Die Suche passiert meist durch einen Shortest-Path-Algorithmus wie zum Beispiel den Dijkstra-Algorithmus. Dieser Schritt kann bei großen Netzen zu einem erhöhten Aufwand führen. Bei einem großen Netz benötigt jeder einzelne Knoten zur Speicherung der Topologie ebenfalls mehr Speicherplatz.

Auf Basis des Link State Routing wurden Vorschläge gemacht, in welcher Form diese Verfahren auf die Anforderungen eines Ad-hoc-Netzwerkes angepasst werden können. Hieraus wurde das optimierte Link State Routing entwickelt. Es handelt sich dabei um ein proaktives Routingverfahren. Wenn ein Netzwerk genutzt wird, welches eine große Anzahl an Veränderungen erfährt und somit oft Routingtabellen aktualisiert werden müssen, empfiehlt sich der Einsatz eins Link-State-Algorithmus. Distanzvektor-Protokolle skalieren nicht optimal mit der Größe von Netzwerken, da die für die Aktualisierung erforderlichen Pakete schnell groß werden [TEI].

Die Optimierung des Link-State-Routings zum Optimized Link State Routing umfasst die Verkleinerung der überwachten Nachbarn und eine Reduzierung des Overheads, der durch Fluten der Nachrichten in das Netzwerk entsteht. Es werden bei diesem Ansatz von jedem Knoten Multipointrelais determiniert, über welche sich Topologie-Kontrollnachrichten über das gesamte Netzwerk verteilen lassen. Durch Hello-Pakete können alle Nachbarn eines Knoten erreicht werden. Aus diesen Nachbarn wird die minimale Anzahl an Nachbarn ausgewählt, um ebenfalls alle 2-Hop-Nachbarn erreichen zu können. Durch die gewählten Multipointrelais werden Nachrichten über das gesamte Netzwerk verteilt. Eine detaillierte Beschreibung des Optimized Link State Routing findet sich in [JMC+01].

Pakete können bei proaktiven Verfahren im Gegensatz zu reaktiven Verfahren garantiert latenzfrei versendet werden. Dies wird durch ein erhöhtes Kontrollnachrichtenaufkommen erkauft. Da die Routen jederzeit unterhalten werden, passiert dies unabhängig von ihrem Nutzungsgrad. Eine Route, die niemals genutzt wird, kostet das Netzwerk Energie, die sich im Hinblick auf diese Route nicht amortisieren kann, was zu einer Verminderung der Lebensdauer von mobilen Knoten führt. [RT99]

3 Das Systemmodell

Die in dieser Arbeit ausgeführten Methodiken bilden zusammen ein Rahmenwerk, welches als "mobile Ad-hoc Complex Event Processing-Framework", oder kurz mAhCEP-Framework, bezeichnet wird.

Dieses Kapitel bespricht die Entitäten, die im CEP-System, dem mAhCEP-Framework und dem mobilen Ad-hoc-Netzwerk vorkommen.

3.1 Modell des CEP-Systems

CEP-Instanzen: Komponenten, die im CEP-System ausgeführt werden, können einen der folgenden drei Typen aufweisen:

- **Sensor**
- **Operator**
- **Konsument**

Die unterschiedlichen Komponenten stellen verschiedene Teile des Flusses in einem Operatorgraph eines CEP-Systems dar. Innerhalb des CEP-Systems können die Komponenten durch ihre Funktionalität voneinander unterschieden werden.

Im mAhCEP-Framework werden nur die Parameter der Ereignisströme zwischen diesen teilnehmenden Komponenten betrachtet. Welche Komponenten über einen betrachteten Ereignisstrom in Kontakt stehen, wird vom mAhCEP-Framework nicht bewertet. Deswegen können alle drei Typen für die Methodiken dieser Ausarbeitung zu einem Typ zusammengefasst werden, der folgend als CEP-Instanz bezeichnet wird.

Definition einer CEP-Instanz: Eine CEP-Instanz besteht aus einem 5-Tupel (G, I, In, Out, F), wobei die einzelnen Elemente wie folgt definiert werden:

- **Operatorgraph** *G* bezeichnet den Operatorgraphen, an welchem die CEP-Instanz beteiligt ist.
- **Identität** *I* bezeichnet die CEP-Instanz innerhalb des Operatorgraphen G. Die Identität muss innerhalb eines Operatorgraphen eindeutig sein.
- **Eingehende Links** *In* ist die Menge der für diesen Operator eingehenden Ereignisströme.
- **Ausgehende Links** *Out* ist die Menge der für diesen Operator ausgehenden Ereignisströme.

- **Status** *F* definiert, ob eine CEP-Instanz migrierbar oder einem mobilen Knoten fest zugewiesen ist.

Bei einer CEP-Instanz, die dem Typen Sensor angehört, existieren keine eingehenden Links, demzufolge ist ihre Menge *In* leer. Entsprechend ist die Menge *Out* einer als Konsument aktiven CEP-Instanz mangels ausgehender Links leer.

CEP-Instanzen werden auf mobilen Knoten des mobilen Ad-hoc-Netzwerks ausgeführt. Ein mobiler Knoten kann hierbei nicht nur eine CEP-Instanz, sondern auch mehrere CEP-Instanzen betreiben. Um eine Eindeutigkeit bei parallel auf einem Knoten ausgeführten CEP-Instanzen garantieren zu können, ist es erforderlich, für eine Instanz eine Identifikation zu vergeben sowie diese Instanz einem Operatorgraphen zuzuweisen. Neben identisch benannten CEP-Instanzen verschiedener Operatorgraphen ist so auch eine Unterscheidung von CEP-Instanzen des gleichen Operatorgraphen, die zeitgleich auf dem selben Knoten ausgeführt werden können, möglich.

Links: Über Links können im CEP-System benachbarte CEP-Instanzen Ereignisse austauschen. Für das mAhCEP-Framework sind zur Bewertung eines Links weitere Parameter notwendig, um den energetischen Aufwand abschätzen zu können, der durch einen Link entsteht. Diese Parameter sollen im Folgenden definiert werden.

Definition eines Links: Ein Link besteht aus einem 6-Tupel (P, G, I, S, F, H), wobei die einzelnen Elemente wie folgt definiert werden:

- **Partner** *P* ist der Kommunikationspartner für diesen Link. Wird der Link innerhalb einer CEP-Instanz gespeichert, um mit dem entsprechenden im Operatorgraphen benachbarten CEP-Instanzen kommunizieren zu können, steht an dieser Stelle der mobile Knoten, auf welchem die benachbarte CEP-Instanz ausgeführt wird. Entsprechend steht dann in den Stati der beiden benachbarten CEP-Instanzen jeweils der Kommunikationspartner. In einem Operatorgraphen ist die Menge P leer, da keine mobilen Knoten betrachtet werden, auf denen CEP-Instanzen ausgeführt werden.
- **Operatorgraph** *G* gibt den Operatorgraphen an, zu welchem der Link gehört.
- **Identität** *I* gibt die Identität der benachbarten CEP-Instanz an, mit welcher die CEP-Instanz kommuniziert. Wird ein Operatorgraph betrachtet, werden beide CEP-Instanzen angegeben.
- **Größe** *S* gibt die Größe eines Ereignisses an, das über diesen Link transferiert wird. Sollen mehrere Werte vorgehalten werden, sind diese durch Kommata zu trennen.
- **Frequenz** *F* gibt die Frequenz an, mit welcher Ereignisse für den Empfänger generiert werden. Die Frequenz bezieht sich auf eine Korrelation, die der Empfänger durchführt. Sollen mehrere Werte vorgehalten werden, sind diese durch Kommata zu trennen.
- **Hopanzahl** *H* gibt die für diesen Knoten gültige Anzahl an Hops an, die notwendig sind, um den Kommunikationspartner zu erreichen oder von ihm ein Ereignis zu erhalten. In einem Operatorgraphen wird diese Eigenschaft nicht betrachtet, da der Operatorgraph keine Aussagen über die physikalischen, sondern lediglich über die logischen Gegebenheiten trifft.

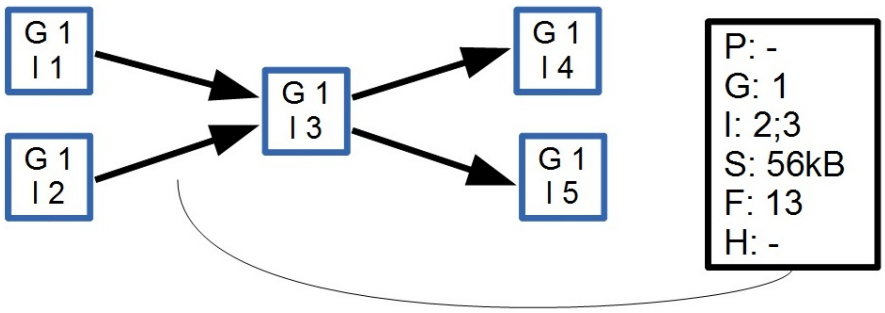


Abbildung 3.1: Ein vereinfachter Operatorgraph mit den Eigenschaften eines Links

Operatorgraphen: Der Ereignisstrom von einem oder mehreren Sensoren zu einem Konsument kann in Form eines Operatorgraphen dargestellt werden, wie dieser unter anderem in [OKRR13] verwendet wurde.

Vorausgehend wurde erläutert, dass für die energetische Bewertung eine Unterscheidung zwischen Sensoren, Operatoren und Konsumenten nicht erforderlich ist und die drei Typen unter dem Begriff einer CEP-Instanz zusammengefasst werden. CEP-Instanzen werden, wie es in Abbildung 3.1 gezeigt wird, durch ein Viereck dargestellt.

Definition eines Operatorgraphen: Ein Operatorgraph G ist ein gerichteter, zyklenfreier Graph, definiert durch ein 3-Tupel (G, C, L), wobei die einzelnen Elemente wie folgt definiert werden:

- **Identität** *G* gibt die Identität des Operatorgraphen an. Die Identität eines Operatorgraphen muss eindeutig sein.
- **CEP-Instanzen** *C* ist die Menge der diesem Operatorgraphen zugehörigen CEP-Instanzen.
- **Links** *L* ist die Menge der Links im Operatorgraphen.

Zwei CEP-Instanzen c_1 und c_2 werden als verbunden bezeichnet, wenn ein Link in L existiert mit I = c_1,c_2 oder I = c_2,c_1.

3.2 Mobile Knoten

Mobile Knoten sind Entitäten, die zusammen das mobile Ad-hoc-Netzwerk bilden. Bedingt durch die Mobilität werden diese durch eine ortsunabhängige Energiequelle gespeist, die nur begrenzte Mengen an Energie aufnehmen und wieder abgeben kann.

Auf mobilen Knoten wird das CEP-System sowie das mAhCEP-Framework ausgeführt. Das CEP-System führt auf dem mobilen Knoten Sensoren, Konsumenten und Operatoren aus, die vom mAhCEP-Framework als CEP-Instanzen abstrahiert werden.

Weiterhin umfassen mobile Knoten Komponenten, die Funktionen zur Kommunikation anbieten. Diese treffen in mobilen Ad-hoc-Netzwerken unter anderem Routingentscheidungen für Nachrichten, die zwischen den mobilen Knoten ausgetauscht werden. Ein mobiler Knoten verfügt mindestens über

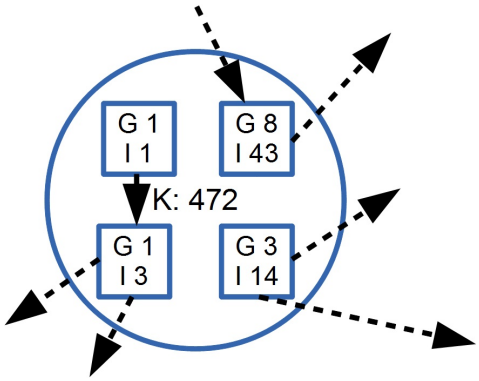


Abbildung 3.2: Ein Knoten führt mehrere CEP-Instanzen aus

eine Schnittstelle, mit welcher er mit anderen mobilen Knoten im Netzwerk Kontakt aufnehmen kann.

Definition eines Knotens: Ein Knoten besteht aus einem 3-Tupel (K, N, C), wobei die einzelnen Elemente wie folgt definiert werden:

- **Identität** *K* ist die Identität, mit welcher ein mobiler Knoten im Netzwerk bekannt ist und angesprochen werden kann. Die Identität jedes mobilen Knotens muss eindeutig sein.
- **Nachbarn** *N* definiert die Menge an Nachbarn, welche K momentan unmittelbar, d.h. ohne Routing durch anderen Knoten, erreichen kann.
- **Ausgeführte CEP-Instanzen** *C* definiert die Menge an Instanzen des CEP-Systems, welche aktuell auf dem mobilen Knoten ausgeführt werden.

3.3 Modell des mobilen Ad-hoc-Netzwerkes

Mobile Knoten kommunizieren über ein drahtloses Medium nur unmittelbar miteinander. Möchten Geräte Nachrichten austauschen, wenn sie sich nicht gegenseitig erreichen können, ist Multihop-Routing notwendig. Reale Bedingungen erschweren einem mobilen Ad-hoc-Netzwerk zusätzlich durch physikalische Gegebenheiten, einen zuverlässigen Dienst zur Verfügung zu stellen. Um die Einflüsse durch die Charakteristika eines mobilen Ad-hoc-Netzwerkes zu dämpfen, müssen in dieser Arbeit einige Annahmen über das System getroffen werden.

Netzwerkpartitionierung In dieser Arbeit wird angenommen, dass eine Netzwerkpartitionierung entweder nicht auftritt oder dass das mobile Ad-hoc-Netzwerk nur so partitioniert wird, dass keine Operatorgraphen davon betroffen sind, damit die einzelnen Operatorgraphen und die durch das mAhCEP-Framework initiierte Optimierung weiterhin fehlerfrei arbeiten können.

Stabilität von Verbindungen Bedingt durch die Mobilität ist es in einem realen Ad-hoc-Netzwerk möglich, dass Verbindungen, die durch einen Routingalgorithmus angegeben werden, nicht mehr existieren, wenn der Zielknoten sich aus dem Ad-hoc-Netzwerk bewegt, eine Partitionierung eintritt oder Routingtabellen nicht mehr stimmen. In dieser Arbeit wird angenommen, dass Routingtabellen konsistent sind und eine Verbindung, die vom Routingalgorithmus als existent angegeben wird, auch stabil ist, damit die vom mobilen Ad-hoc-Netzwerk zurückgegebene Hopanzahl korrekt ist.

Homologie von Funkkanälen Durch die Vielfalt von Produkten ist in einem realen Netzwerk mit einer Heterogenität von Übertragungsleistungen zwischen den einzelnen mobilen Knoten zu rechnen. In dieser Arbeit wird davon ausgegangen, dass alle Knoten mit der gleichen Übertragungsgeschwindigkeit und -leistung arbeiten, damit eine äquivalente Bewertung zwischen allen Kommunikationspaaren stattfinden kann.

Topologie während einer Datenübertragung Je nach Dauer einer Datenübertragung ist es in einem mobilen Ad-hoc-Netzwerk möglich, dass sich die Topologie ändert, während die Datenübertragung andauert. In dieser Arbeit wird davon ausgegangen, dass während einer Datenübertragung die Netzwerktopologie stabil bleibt, sodass eine Datenübertragung erfolgreich beendet werden kann.

Keine kompromittierenden Knoten In einem öffentlichen Netzwerk gibt es Teilnehmer, die das Netzwerk nicht bestimmungsgemäß nutzen und ihm stattdessen Schaden zufügen wollen. Solche Fälle müssen in realen Systemen durch Sicherheitseinrichtungen abgefangen werden. In dieser Arbeit wird davon ausgegangen, dass es keine Knoten gibt, die das System, beispielsweise durch Fälschen von Routingeinträgen, kompromittieren möchten.

Keine Ausfälle und Fehler von Knoten mit CEP-Komponenten Unter realen Bedingungen ist es möglich, dass Knoten durch äußere Einflüsse, durch defekte Akkumulatoren oder durch andere Gründe bedingt plötzlich ausfallen, ohne sich ordnungsgemäß vom Netzwerk abzumelden. Dies hat zur Folge, dass ebenfalls CEP-Komponenten aus dem Netzwerk ausfallen könnten. Besteht keine Redundanz, ist der Operatorgraphen defekt und stellt seine Arbeit ein, da an mindestens einer Stelle keine Ereignisse mehr für eine Korrelation zur Verfügung stehen. In dieser Arbeit wird davon ausgegangen, dass Knoten nicht spontan ausfallen, sondern sich ordnungsgemäß vom Netzwerk abmelden.

Implementierung von “Power Aware Multi-Access Protocol with Signaling Ad Hoc Networks“ im mobilen Ad-hoc-Netzwerk Um die Sprungdistanz zwischen zwei Knoten unabhängig von der Menge der auf dem Weg anzutreffenden Nachbarn bewerten zu können, wird angenommen, dass in dem mobilen Ad-hoc-Netzwerk, wie es üblich ist, ein Verfahren zur Kollisionskontrolle und zur Energieersparnis, wie es auch in Abbildung 2.4 gezeigt wird, implementiert ist.

3.4 Energiemodell

In Kapitel 2.3 wurde bereits erläutert, dass ein Netzwerkinterface zur drahtlosen Kommunikation für unterschiedliche Modi verschiedene Energieanforderungen aufweist (Tabelle 2.1). In dieser Arbeit wird der Aufwand der Energie für den Austausch zwischen zwei Knoten durch die Anzahl der Hops abstrahiert. In [BBV09] werden verschiedene Energiekosten für unterschiedliche Ereignisgrößen für die Übermittlung über 802.11 angegeben.

Für diese Arbeit soll als benötigte Energie der Mittelwert des Energieverbrauchs für alle ein- und ausgehenden Kommunikationsverbindungen herangezogen werden. Für eine Übertragung müssen auf der empfangenden Gegenseite Energiekosten mit etwa 80% des Energieaufwands des sendenden mobilen Knotens veranschlagt werden [RZ07]. Der Energieaufwand für jede Übertragung aus der Sicht des mobilen Knotens ist somit um den Faktor 1,8 höher. Dieser Faktor steht vor jeder Verbindung, weswegen er im Vergleich des Energieaufwandes nicht betrachtet werden muss. Mobile Knoten, die an einer Übertragung nicht interessiert sind, versetzen ihre Empfänger für die Dauer der Datenübertragung in den Sleep-Modus und müssen nicht die Energie aufwenden, die für den Empfang einer Nachricht erforderlich ist.

Dies führt dazu, dass das Modell des Energieverbrauchs für die Berechnung auf den mobilen Knoten vereinfacht werden kann. Die Übertragung eines Ereignisses für das CEP-System beträgt von einer CEP-Komponente *a* im Operatorgraphen *G* zu einer anderen Komponente *b* dementsprechend einen Aufwand von:

$$Linkkosten = Hopanzahl * Frequenz * Größe \tag{3.1}$$

4 Problemstellung

Dieses Kapitel zeigt, wieso herkömmliche, in einer Infrastruktur ausgeführte Systeme zur komplexen Ereignisverarbeitung nicht den Problemen ausgesetzt sind, die auftreten, wenn ein solches System in einem mobilen Ad-hoc-Netzwerk ausgeführt wird. Am Ende des Kapitels wird die Problemstellung dieser Arbeit formuliert.

4.1 Herkömmliches CEP

In einem herkömmlichen CEP-System werden CEP-Instanzen, die Operatoren repräsentieren, in der Infrastruktur ausgeführt. Konsumenten und Sensoren können wahlweise auf mobilen Endgeräten oder ebenfalls in der Infrastruktur ausgeführt werden. Im Fall, dass CEP-Komponenten auf mobilen Knoten ausgeführt werden, kommunizieren diese über eine Zugangspunkt über einen drahtlosen Kommunikationskanal mit der Infrastruktur.

Unerheblich, wo in der Infrastruktur die CEP-Instanz ausgeführt wird entstehen die Kosten für die Nutzung des Funkkanals pro mobilem Knoten unabhängig von der Konfiguration für jedes Ereignis, welches er generiert oder erhält, genau einmal, da der Knoten direkt mit einer Basisstation der Infrastruktur kommuniziert, wie es in Abbildung 4.1 gezeigt wird. Der Teil des Ereignisstroms vom

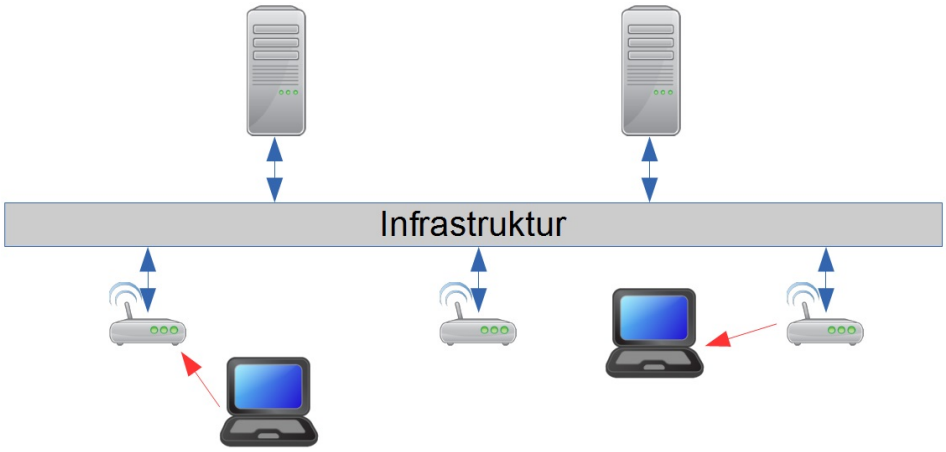


Abbildung 4.1: Ein CEP-System wird mit mobilen Knoten und Operatoren in der Infrastruktur ausgeführt

Sensor zum Konsumenten, der über eine drahtlose Verbindung transportiert wird, wird durch rote Pfeile dargestellt. Ob die CEP-Instanz auf der linken oder der rechten Maschine in der Abbildung ausgeführt wird, ist für die drahtlose Verbindung unerheblich. Die energetische Belastung für die mobilen Knoten bleibt gleich.

Eine Optimierung ist folglich aus dem Gesichtspunkt der Energie, die von mobilen Knoten aufgewendet werden muss, in einer Infrastruktur nicht notwendig, weswegen dort kein speziellen Methodiken angewendet werden müssen. CEP-Instanzen können aus energetischer Sicht der mobilen Knoten in solchen CEP-Systemen manuell vom Administrator des CEP-Systems auf ausgewählte Knoten platziert werden und anschließend den Betrieb aufnehmen.

Bei der manuellen Platzierung von CEP-Instanzen durch einen Administrator oder der automatisierten Platzierung durch Algorithmen, die folgend in Kapitel 9 angesprochen werden, können verschiedene Ansätze zur Optimierung des CEP-Systems herangezogen werden. Solche Ansätze werden durch Metriken ausgedrückt. Eine Metrik gibt ein Maß für die Erfüllung von bestimmten gesetzten Kriterien, sodass mehrere Konfigurationen miteinander verglichen werden können. Häufig wird als Metrik das Bandbreite-Verzögerungs-Produkt angewendet. Dies ist in einem Infrastrukturnetzwerk sinnvoll, um Ereignisse den Konsumenten möglichst schnell zustellen zu können.

4.2 CEP in einem mobilen Ad-hoc-Netzwerk

In einem mobilen Ad-hoc-Netzwerk besteht das gesamte System aus gleichberechtigten, unter Umständen mobilen, Knoten und es existiert keine Infrastruktur. Die Charakteristiken von mobilen Knoten und einem mobilen Ad-hoc-Netzwerk erfordern Metriken für Instanzsetzungen, die in herkömmlichen CEP-Systemen, die in der Infrastruktur ausgeführt werden, nicht erforderlich sind. Im Folgenden werden die Charakteristiken aufgezählt:

Multihop-Routing: Im Gegensatz zu einem Infrastrukturnetzwerk ist jeder Knoten gleichermaßen an Routingentscheidungen beteiligt. Datenpakete, die von einem mobilen Knoten zu einem anderen mobilen Knoten gesendet werden, der sich nicht in unmittelbarer Reichweite der drahtlosen Funkverbindung aufhält, müssen mittels Multihop-Methodiken, die in Kapitel 2 beschrieben werden, weitergeleitet werden, bis sie den Zielknoten erreichen. Dies führt dazu, dass auch mobile Knoten in Ereignisströme involviert werden, die am entsprechenden Operatorgraphen nicht teilhaben, sondern lediglich die logistische Funktion übernehmen. Dies kostet Energie, was sich durch eine verkürzte Lebensdauer bemerkbar macht.

Mobilität: In einem mobilen Ad-hoc-Netzwerk sind mobile Knoten nicht an einen festen Standort gebunden, sondern können von ihren Betreibern bewegt werden. Dies führt dazu, dass die Platzierung einer CEP-Instanz, die zu einem Zeitpunkt günstig war, zu einem späteren Zeitpunkt nicht mehr optimal sein muss. Die daraus resultierenden längeren Wege involvieren, wie bereits ausgeführt wurde, am Operatorgraphen unbeteiligte mobile Knoten.

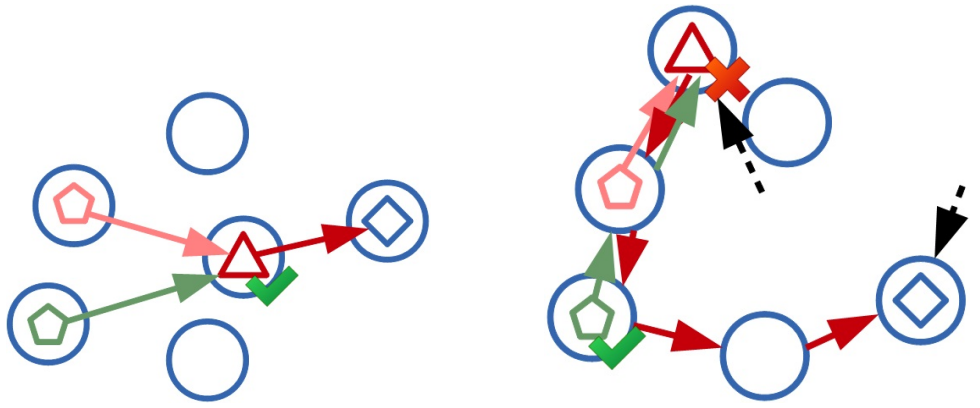
4.3 Optimale Instanzsetzung in einem mobilen Ad-hoc-Netzwerk

Abbildung 4.2 zeigt die Konfiguration eines CEP-Systems in einem mobilen Ad-hoc-Netzwerk. In Abbildung 4.2a wurde eine optimale Instanzsetzung gewählt, die sich in nur insgesamt drei notwendigen Hops äußert. Eine gleichwertig günstige Instanzsetzung wäre auf einem der Knoten möglich, die die Sensoren ausführen. Eine Ausführung der CEP-Instanz auf dem Knoten des Konsumenten würde jedoch die Wahl der Instanzsetzung verschlechtern, da dann anstelle des roten Ereignisstroms die beiden Ereignisströme der Sensoren diesen Funkkanal nutzen würden. Eine Kommunikationsstecke würde demnach doppelt so lange beansprucht, wie bei optimaler Setzung.

In Abbildung 4.2b wird die Mobilität des mobilen Ad-hoc-Netzwerkes durch gestrichelte schwarze Pfeile dargestellt. Die Konsequenz dieser Mobilität ist eine Veränderung der Pfade, die von den Ereignisströmen genutzt werden müssen. Die genutzte Hop-Anzahl ist hier auf sieben angestiegen, obwohl eine (in diesem Fall einzige) günstige Lösung auf dem Knoten existiert, der den grün gefärbten Sensor ausführt. Dort wäre die Hop-Anzahl, die benötigt werden würde, um den Konsumenten bedienen zu können, nur drei. Diese Verbesserung kommt auch dadurch zustande, dass der Knoten mit dem ausgeführten Konsumenten in die Reichweite des Knotens gewandert ist, der in dieser Abbildung auf dem roten Ereignisstrom liegt.

Dies führt zu einem Energiebedarf, der höher ist, als es für die gezeigte Situation erforderlich ist. Eine Metrik für die Instanzsetzung in einem mobilen Ad-hoc-Netzwerk muss zur Maximierung der Gesamtlaufzeit des Systems diese so bewerten, dass eine Wertung von Instanzsetzungen mit der benötigten Energie, die für die Ausführung des resultierenden CEP-Systems erforderlich ist, möglich ist.

Ein Administrator kann zu Beginn einen Operatorgraphen für ein mobilen Ad-hoc-Netzwerk die Anzahl der Hops so optimieren, dass sie möglichst minimal wird. Bedingt durch Topologieänderungen



(a) Eine optimale Platzierung eines Operators **(b)** Suboptimale Platzierung eines Operators

Abbildung 4.2: Verschiedene Konfigurationen eines CEP-Systems in einem Ad-hoc-Netzwerk mit Operatorgraph

muss diese Optimierung unter Umständen regelmäßig und oft durchgeführt werden. Der Administrator eines großen CEP-Systems kann einer solchen Aufgabe nicht gerecht werden, weshalb sie von einem Algorithmus ausgeführt werden muss.

Die optimale Instanzsetzung eines CEP-Systems durch Zuhilfenahme von Vivaldi-Koordinaten in der Infrastruktur ist nach [RDR10] ein NP-hartes Problem. Abstrahiert man das mobile Ad-hoc-Netzwerk zu jedem Moment als festes Netzwerk und die zu diesem Zeitpunkt gültigen Hopanzahlen von mobilen Knoten zu anderen mobilen Knoten als Maß für die Entfernung und richtet die mobilen Knoten entsprechend dieser Entfernungen aus, wie es von [RDR10] bei Vivaldi-Koordinaten passiert, so zeigt sich, dass die optimale Instanzsetzung eines CEP-Systems in einem mobilen Ad-hoc-Netzwerk ebenfalls ein mindestens genauso schweres Problem darstellt. Eine auf mobilen Knoten ausführbare Lösung scheitert neben der Erfordernis von Wissen um andere Knoten bereits hieran, da keine effektiv ausführbare Methode zur Lösung dieses Problems auf mobilen Knoten existiert. Ein solcher Algorithmus benötigt Energie für sehr viele Ergebnisse, von denen nur wenige - oder sogar nur eines - eine optimale Instanzsetzung als Resultat liefern. Gleichzeitig trifft dieser Algorithmus auf die Schwierigkeit, schnell genug ausgeführt zu werden, um an die ändernde Topologie adaptieren zu können. Ein Algorithmus, der basierend auf lokalem Wissen seine Entscheidungen trifft und von diesem Problem nicht betroffen ist, wird im Verlauf dieser Arbeit eingeführt.

Optimale Instanzsetzung ist ein Optimierungsproblem, das durch direkte Wahl einer geeigneten Konfiguration oder einer Folge von Konfigurationen an folgenden Wert konvergiert:

$$optimalInstanceplacement \rightarrow \min \sum_{G} \sum_{L \in G} Linkkosten \tag{4.1}$$

5 Interaktion im und mit dem mAhCEP-Framework

Dieses Kapitel führt das “mobile Ad-hoc Complex Event Processing-Framework“ - kurz mAhCEP-Framework - ein, welches die Problemstellungen, die durch die Charakteristiken eines mobilen Ad-hoc-Netzwerk hervorgerufen werden, durch geeignete Methoden löst. Zunächst wird ein Überblick über die einzelnen Komponenten des mAhCEP-Frameworks gegeben, anschließend wird erläutert, wie sich das mAhCEP-Framework zwischen den mobilen Knoten und dem CEP-System eingliedert.

5.1 Architektur des mAhCEP-Frameworks

Das mAhCEP-Framework besteht aus mehreren Komponenten, die in ihrer Gesamtheit die Funktionalität des mAhCEP-Frameworks bereitstellen. Die verschiedenen Komponenten und ihre Abhängigkeiten sind in der Abbildung 5.1 dargestellt.

Die einzelnen Komponenten weisen folgende Funktionalitäten auf:

- **Migration**

  Innerhalb dieser Komponente werden Migrationsentscheidungen für CEP-Instanzen getroffen oder Aktualisierungen, die aufgrund von Migrationen benachbarter CEP-Instanzen notwendig geworden sind, ausgeführt. Sie wird periodisch aufgerufen, um eigene Instanzplatzierungen zu

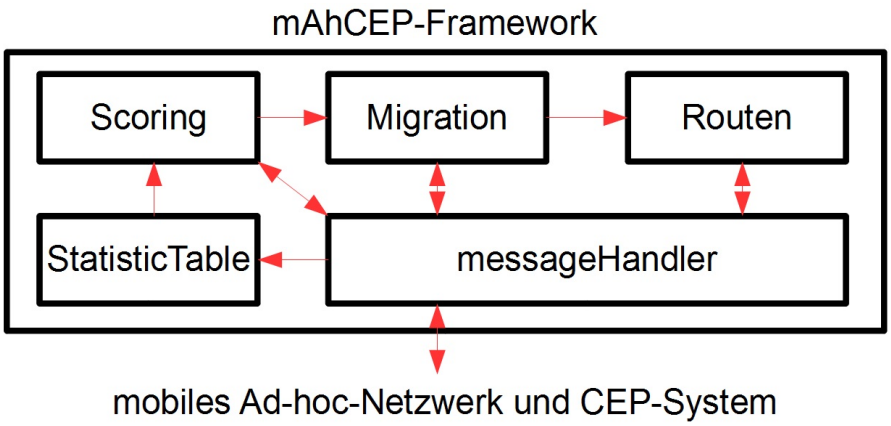


Abbildung 5.1: Die Komponenten des mAhCEP-Frameworks

überprüfen. Treffen Kontrollnachrichten bezüglich benachbarter CEP-Instanzen ein, teilt die Migrations-Komponente die Änderungen dem CEP-System mit.

- **Scoring**

  Soll eine Instanzplatzierung geprüft werden, wird von der Komponente, die die Migration koordiniert, die Scoring-Komponente verwendet. Im mAhCEP-Framework werden die Energiekosten, die auf einem mobilen Knoten für eine CEP-Instanz entstehen, durch ein Scoring repräsentiert. Eine Entscheidung, ob ein mobiler Knoten eine CEP-Instanz an einen Nachbarn übergibt, kann durch Betrachtung und Vergleich der verschiedenen Scores getroffen werden. Ein niedrigerer Score entspricht einer besseren Platzierung, da weniger Energie aufgewendet werden muss, um die CEP-Instanz zu unterhalten.

- **StatisticTable**

  Energiekosten, die durch die Übertragung von Ereignissen entstehen, können nicht nur durch die Anzahl an Hops gemessen werden. Die Belegung des drahtlosen Kommunikationskanals ist im Wesentlichen auch von der Frequenz und der Größe der Ereignisse abhängig. Um eine sinnvolle Energiebilanz auch bei schwankender Ereignisgröße gewährleisten zu können, müssen Statistiken über Ereignisse, die von einer CEP-Instanz zu einer anderen transportiert werden, erhoben werden. Die Komponente StatisticTable realisiert eine Anpassung der Größe und Frequenz durch Messung von ein- und ausgehenden Nachrichten, wobei die Frequenz pro Korrelation angegeben wird. Die erhobenen Daten werden von der Scoring-Komponente genutzt, um eine zuverlässige Aussage zum aktuellen Score der CEP-Instanz für den Knoten treffen zu können, auf welchem das Scoring aktuell ausgeführt wird.

- **Routen**

  Erfolgt eine Migration, müssen benachbarte CEP-Instanzen von dieser Migration unterrichtet werden, damit sie ihre Links für die entsprechenden CEP-Instanzen anpassen können. Wenn nur eine CEP-Instanz migriert, stellt dies kein Problem dar, da die zu ändernde CEP-Instanz keinen Wechsel des mobilen Knotens vollzieht, sondern nur die migrierende. Die Adresse, mit welcher die migrierte CEP-Instanz die benachbarten CEP-Instanzen erreichen kann, ändert sich nicht. Falls mehrere CEP-Instanzen, die benachbart sind, eine Migration ausführen, führt dies bei gleichzeitiger Migration zwangsläufig zu Inkonsistenzen, da beide Nachbarn die jeweils veraltete Adresse nutzen, um ihre benachbarten CEP-Instanzen zu erreichen. In der Folge der Migrationen wird keiner von beiden den jeweils anderen erreichen. Der Operatorgraph fällt damit aus, da keine Korrelation beim Operator, der als CEP-Instanz auf einem der beiden mobilen Knoten ausgeführt wird, mehr stattfindet. Die Routen-Komponente vermeidet Inkonsistenzen, indem sie Zeiger auf mobilen Knoten anlegt, die eine CEP-Instanz an einen anderen mobilen Knoten migriert haben und hält die Zeiger so lange aufrecht, wie es erforderlich ist. Sie wird von der Migrations-Komponente bei einer Migration genutzt. Die Migrations-Komponente teilt den neuen mobilen Knoten, unter welchem die migrierende CEP-Instanz erreichbar ist, der Routen-Komponente mit, sodass diese die entsprechenden Weiterleitungen einrichten kann.

5.2 Architektur des Gesamtsystems

Das in dieser Arbeit vorgestellte mAhCEP-Framework gliedert sich zwischen dem mobilen Knoten und dem CEP-System ein, wie es in Abbildung 5.2 dargestellt ist.

Von den unterschiedlichen Schichten müssen verschiedene Schnittstellen angeboten werden, durch welche benachbarte Schichten mit diesen kommunizieren können.

5.2.1 Kommunikation zwischen dem CEP-System und dem mAhCEP-Framework

In Abbildung 5.2 wird gezeigt, dass das CEP-System und das mAhCEP-Framework durch zwei verschiedene Nachrichtentypen miteinander interagieren können:

- Kontrollnachrichten
- Ereignisse in Ereignisströmen

Kontrollnachrichten

Kontrollnachrichten werden zwischen den beiden Schichten aus einem der drei folgenden Gründe ausgetauscht:

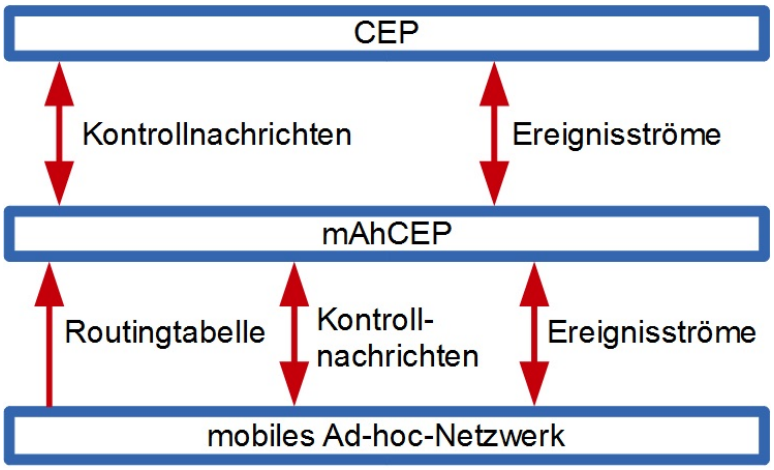


Abbildung 5.2: Das Schichtenmodell mit dem eingebundenen mAhCEP-Framework

Überprüfung einer Instanzsetzung Löst das mAhCEP-Framework die Überprüfung einer Instanzsetzung aus, muss vom CEP-System eine Konfiguration für diese CEP-Instanz an das mAhCEP-Framework zurückgegeben werden, damit für sie ein Scoring erstellt werden kann. Die Daten umfassen dabei die Identifikation der zu überprüfenden CEP-Instanz sowie alle ein- und ausgehenden Links, welche die CEP-Instanz mit ihren Nachbarn verbindet. Diese Funktionalität kann von einer Funktion **getConfiguration(graph Graph, instance Identifier) return cepinstance** bereitgestellt werden.

Migration einer CEP-Instanz Wird vom mAhCEP-Framework auf Basis eines Scorings eine Migration ausgelöst, muss es für das mAhCEP-Framework möglich sein, CEP-Instanzen auf mobilen Knoten zu löschen und anzulegen, um auf dem nicht mehr optimalen mobilen Knoten die CEP-Instanz entfernen und sie auf dem neuen, besseren mobilen Knoten wieder erstellen zu können. Diese Funktionalität muss vom CEP-System bereitgestellt werden. Dies kann in Form einer Prozedur **createInstance(cepinstance Instance)** und einer Funktion **destroyInstance(graph Graph, instance Identifier) return cepinstance** geschehen.

Korrektur von CEP-Instanzen Migriert eine CEP-Instanz, müssen benachbarte CEP-Instanzen ihre Links aktualisieren, um weiterhin Ereignisse auf den entsprechenden Ereignisströmen austauschen zu können. Das mAhCEP-Framework versendet Kontrollnachrichten vom neuen mobilen Knoten, welche eine Korrektur von Einträgen der benachbarten CEP-Instanzen veranlassen. Zur Realisierung dieser Funktionalität muss das CEP-System eine Schnittstelle bereitstellen, durch welche das mAhCEP-Framework Änderungen an momentan ausgeführten CEP-Instanzen vornehmen kann. Diese Schnittstelle wird durch eine Prozedur **modifyInstance(cepinstance Instance)** ausgeführt.

Ereignisströme

Ereignisse innerhalb von Ereignisströmen werden von den Komponenten des CEP-Systems generiert und versendet beziehungsweise empfangen. Da das mAhCEP-Framework für seine Funktionalität auf Statistiken bezüglich der Größe und Frequenz von Ereignissen angewiesen ist, findet der Austausch von Ereignissen zwischen dem CEP-System und dem mobilen Ad-hoc-Netzwerk nicht wie bei herkömmlichen CEP-Systemen üblich direkt, sondern durch die Schicht des mAhCEP-Framework hindurch, statt. Weiterhin müssen bei einer Migration von CEP-Instanzen Ereignisse umgeleitet werden, wenn sie an einem mobilen Knoten ankommen, der die CEP-Instanz, an welche die Ereignisse adressiert sind, nicht mehr betreibt. Diese Aufgabe wird von der Routen-Komponente übernommen. Damit beide Schichten Ereignisse austauschen können, muss vom CEP-System eine Schnittstelle angeboten werden, die es dem mAhCEP-Framework erlaubt, Ereignisse zuzustellen. Eine Prozedur **receiveMessage(message Message)** realisiert diese Anforderung. Das CEP-System ist durch eine Prozedur **sendMessage(message Message)**, die vom mAhCEP-Framework zur Verfügung gestellt wird, in der Lage, Ereignisse zu versenden.

5.2.2 Kommunikation zwischen dem mAhCEP-Framework und dem mobilen Ad-hoc-Netzwerk

Unterhalb der Schicht des mAhCEP-Frameworks befindet sich das mobile Ad-hoc-Netzwerk. Zwischen diesen beiden Schichten werden Kontrollnachrichten zwischen mAhCEP-Instanzen, die auf verschiedenen mobilen Knoten ausgeführt werden, sowie Ereignisse, die von CEP-Instanzen generiert und empfangen werden, ausgetauscht. Diese Funktionalität wird von zwei Prozeduren zur Verfügung gestellt. **sendMessage(message Message)** wird vom mobilen Ad-hoc-Netzwerk angeboten, um es dem mAhCEP-Framework zu ermöglichen, Nachrichten zu versenden. **receiveMessage(message Message)** steht vom mAhCEP-Framework zur Verfügung und dient dem mobilen Ad-hoc-Netzwerk zur Zustellung von Ereignissen und Kontrollnachrichten.

Die Scoring-Komponente benötigt zur Ermittlung des Score für eine CEP-Instanz Zugriff auf die Routingtabelle des mobilen Knotens, um die Hopzahl aus Sicht des aktuellen mobilen Knotens für benachbarte CEP-Instanzen ermitteln zu können. Das mobile Ad-hoc-Netzwerk muss diese Möglichkeit durch eine Funktion **getHopCount(identifier Identifier) return Natural** bereitstellen.

6 Adaptionsalgorithmen

Nachdem im vorausgegangenen Kapitel die Interaktion der verschiedenen Komponenten des mAhCEP-Frameworks ausgeführt wurde, wird in diesem Kapitel auf die einzelnen Komponenten und die Funktionsweise der Algorithmen des mAhCEP-Frameworks eingegangen. Es wird auf die statistische Erfassung von Ereignissen eingegangen, die notwendig ist, um Voraussagen über den Energieaufwand treffen zu können. Dann wird der Ablauf einer Prüfung der Instanzplatzierung von CEP-Instanzen ausgeführt, welche mit lokalem Wissen durchgeführt wird. Diese überprüft für eine CEP-Instanz, ob sich diese in ihrem momentanen Optimum befindet oder ob die Migration dieser CEP-Instanz energetische Vorteile ergibt. Folgend wird beschrieben, wieso bei Migrationen Konsistenzprobleme auftreten können und wie diese gelöst werden können. Danach geht das Kapitel auf die Migration ein, die CEP-Instanzen dynamisch zwischen mobilen Knoten verschiebt. Das Kapitel wird mit dem Verfahren für den Beitritt und Abgang von mobilen Knoten und Operatorgraphen abgeschlossen.

Im mAhCEP-Framework kann für jede auf dem Knoten ausgeführte CEP-Instanz ein Zustandsdiagramm, wie es in Abbildung 6.1 gezeigt ist, angenommen werden. Die erwähnten Prozeduren werden im Laufe dieses Kapitels in den entsprechenden Abschnitten eingeführt.

Migration einer CEP-Instanz Die für das CEP-System im Hintergrund stattfindenden Optimierungen beginnen mit der Migration einer CEP-Instanz auf einem mobilen Knoten *K*. Eine Neuerstellung einer CEP-Instanz wird analog ausgeführt, indem das CEP-System eine CEP-Instanz startet. Dies kann wie eine Migration gesehen werden. Die Migration umfasst hierbei die Übergabe der Konfiguration der CEP-Instanz an das CEP-System sowie der statistischen Daten des mAhCEP-Frameworks an die **StatisticTable**.

Normalbetrieb Das System geht dann in den Normalbetrieb, in welchem Ereignisse durch das mAhCEP-Framework hindurch zwischen dem CEP-System und dem mobilen Ad-hoc-Netzwerk ausgetauscht und zusätzlich von **messageStats** in der **StatisticTable** Statistiken über diesen Austausch geführt werden.

Abfrage Nachbarn Entschließt sich das mAhCEP-Framework, eine CEP-Instanz eventuell migrieren zu wollen, müssen die Nachbarn *N* des mobilen Knoten *K* über die Bedingungen, die sie der CEP-Instanz geben können, befragt werden. Diese Aktion wird durch die Prozedur **getScores(cepinstance Instance)** der Scoring-Komponente ausgelöst.

Warten auf Antworten Der mobile Knoten *K*, der die Prozedur **getScores(cepinstance Instance)** aufgerufen hat, wartet auf die Antworten seiner Nachbarn *N*, die ihrerseits ihren eigenen Score berechnen müssen. Die Prozedur **calcScore(cepinstance Instance)** kann zur Ermittlung des eigenen Scores herangezogen werden.

Auswertung Scores Der mobile Knoten *K* bewertet nach einer Zeitgrenze, innerhalb welcher die Antworten eintreffen müssen, die abgegebenen Scores der benachbarten mobilen Knoten *N*. Es müssen nicht von allen Nachbarn Nachrichten eintreffen. Nachbarn mit größeren und dementsprechend schlechteren Scores verzichten zur Energieeinsparung auf eine Antwort. Das mAhCEP-Framework entscheidet nun auf Basis der vorhandenen Daten, ob eine Migration der CEP-Instanz sinnvoll ist. Ist sie nicht sinnvoll, geht der Zustand wieder über in den Normalbetrieb, ansonsten wird die Migration gestartet.

Migration CEP-Instanz Der mobile Knoten *K* beendet die im CEP-System ausgeführte CEP-Instanz und sendet diese mit den Konfigurations- und Statistikdaten des mAhCEP-Frameworks an den übernehmenden Knoten *K'*.

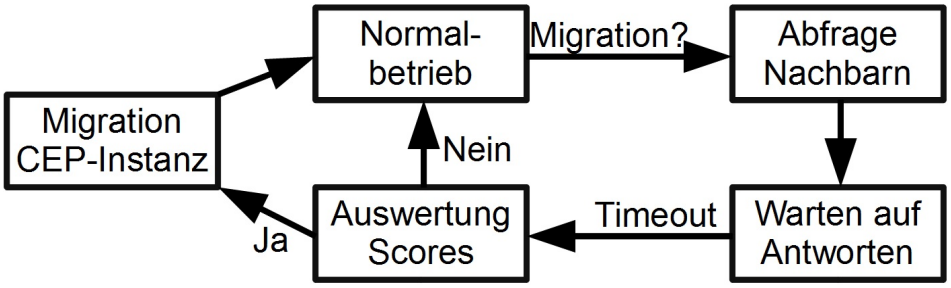


Abbildung 6.1: Der Zustandsautomat für Migrationsentscheidungen einer CEP-Instanz

Die im Folgenden beschriebenen Komponenten dienen dazu, diesen Vorgang auszuführen.

6.1 Zählung von Ereignissen

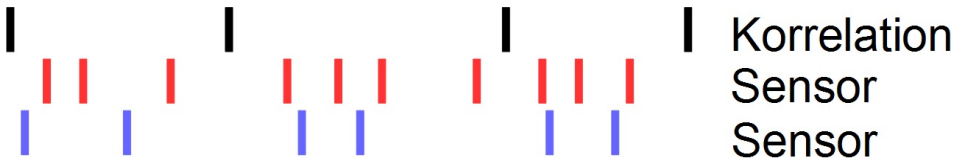


Abbildung 6.2: Frequenz- und Größenermittlung bezogen auf die stattfindenden Ereigniskorrelationen

Um Informationen über Frequenzen und Größe von Ereignissen zu erhalten, müssen die ein- und ausgehenden Ereignisströme des CEP-Systems vom mAhCEP-Framework überwacht werden. Die Formel 3.1 zur Bewertung der aufgewendeten Energie bedingt die Kenntnis der oben genannten Fakten. Schwankende Ereignisgrößen und -frequenzen, wie sie in einem CEP-System vorkommen können, verhindern ein Annehmen von Festwerten. Bei einer Annahme von Festwerten durch den Administrator des CEP-Systems zur Einrichtung eines Operatorgraphen können Energiebilanzen, die nach längerer Zeit erstellt werden, unzutreffend sein, wenn sich die charakteristischen Größen eines Ereignisstroms ändern. Ereignisströme werden vom CEP-System zunächst an das mAhCEP-Framework übergeben, bevor sie ausgesendet werden, somit kann eine Zählung von ein- und ausgehenden Ereignissen stattfinden.

6.1.1 Die Tabelle StatisticTable

Im mAhCEP-Framework werden die Werte der durchschnittlichen Ereignisgröße und -frequenz zur Laufzeit ermittelt und bei Veränderungen aktualisiert. Dies wird mit Hilfe einer Tabelle realisiert, welche als **StatisticTable** bezeichnet wird und in welcher zu jedem Link, der gerade auf einem mobilen Knoten ausgeführt wird, die Parameter der durchschnittlichen ermittelten Größe und Frequenz gespeichert werden. In der Tabelle werden in den einzelnen Zeilen der Operatorgraph, die ursprüngliche und die adressierte CEP-Instanz, die dazu gehörenden physikalischen Knoten sowie die durchschnittlichen Größen und Frequenzen und die letzten ermittelten Werte vorgehalten. Die CEP-Instanzen dienen hierbei der Zuordnung der eingegangenen Ereignisse, die physikalische Knotenadressierung dient der eigentlichen Optimierung durch das mAhCEP-Framework, da durch dieser Werte die Anzahl an Hops ermittelt werden kann, die dieser Link beansprucht.

Als Bezugspunkt für die Durchschnittsermittlung wird jeweils pro CEP-Instanz das Aussenden eines Ereignisses herangezogen, wie es in Abbildung 6.2 gezeigt wird, sodass sich für die Frequenz die Einheit

$$\frac{Ereignisse}{Ereigniskorrelation} \tag{6.1}$$

ergibt. Die Zeit zwischen zwei Aussendungen wird folgend als Korrelationsfenster bezeichnet. Der in rot dargestellte Sensor sendet dabei drei Ereignisse pro Korrelation an den Operator, wobei in einem Korrelationsfenster vier Ereignisse übertragen werden. Der in blau dargestellte Sensor sendet konstant mit einer Frequenz von 2 Ereignissen pro Korrelation. Für Operatoren bedeutet das Aussenden eines Ereignisses eine erfolgte Ereigniskorrelation, wobei Aussendungen an mehrere Empfänger für eine identische Ereigniskorrelation einmal zu bewerten sind.

Als Designparameter lassen sich die letzten n gemessenen Nachrichtengrößen und die letzten m ermittelten Nachrichteneingänge festlegen. Je kleiner diese Werte gewählt werden, desto schneller konvergiert das System gegen schwankende Werte. Die geeignete Wahl wird vom Administrator des mAhCEP-Frameworks festgelegt.

OG	Instanzen	Link	∅ Größe	∅ Frequenz	letzte n Größen	letzte m Nachrichten
1	9;10	13;56	3 kB	5	3;3,1;2;9	5;5;5;5

Tabelle 6.1: StatisticTable zur Bestimmung der Nachrichtenfrequenz und -größe

Algorithmus 6.1 messageStats: Mittelwertbildung von Größe und Frequenz von Ereignissen

```
procedure MESSAGESTATS(Message)
    if Message.Sender = Me.ID then                                  // im Fall einer Aussendung
        for each Line in StatisticTable where Line.OG = Message.OG ∧
            Line.CEPInst = Message.CEPInst do                       // alle betroffenen Zeilen bearbeiten

            Line.AverageFreqency := (Line.AverageFreqency + Σ_{i=1}^{m} Line.LastMessages(i)) / (m+1)

            Line.AverageSize := (Line.AverageSize + Σ_{i=1}^{n-1} Line.LastSizes(i) + Line.LastSizes(n)/Line.LastMessages(m)) / (n+1)
            SHIFTVALUES(LastMessages)                               // letzte Werte löschen und neues
            SHIFTVALUES(LastSizes)                                  // Intervall beginnen
        end for
    else
        with Line in StatisticTable where Line.OG = Message.OG and
          Line.CEPInst := Message.CEPInst                           // eingegangene Nachricht in Statistik nehmen
          Line.lastMessages(m)++                                    // in letztem Nachrichtenintervall hochzählen
          Line.lastSizes(n) := Line.lastSizes(n) + Message.Size     // Größe aufaddieren
        end with
    end if
end procedure
```

6.1.2 Errechnung der Mittelwerte aus den Daten der StatisticTable

Bei jeder Aussendung einer CEP-Instanz, die am mobilen Knoten betrieben wird, muss eine Berechnung der Mittelwerte für den Abschluss des aktuellen Korrelationsfensters stattfinden. Während des Korrelationsfensters werden die Werte für die Größe und die Anzahl der eingetroffenen Ereignisse in den entsprechenden Datensätzen der **StatisticTable** aufaddiert. Mit der Korrelation der eingegangenen Ereignisse müssen die entsprechenden Mittelwerte aus diesen Werten gebildet werden.

In Algorithmus 6.1 wird der Ablauf einer solchen Prozedur, die als **messageStats** bezeichnet wird, erläutert. Zuerst überprüft **messageStats**, ob das eingegangene Ereignis vom mobilen Knoten selbst gesendet wurde. Hierzu wird die im Ereignis enthaltene Absender-ID mit der eigenen Knoten-ID verglichen.

Handelt es sich um ein Ereignis vom mobilen Knoten selbst, so wird, wie in Abbildung 6.2 dargestellt ist, das ausgehende Ereignis als Maß für die Frequenz herangezogen. Der Durchschnitt wird gebildet durch die Addition und anschließende Teilung des Gesamtdurchschnitts und der m beziehungsweise n letzten Aufzeichnungen. Bei der Größe wird im aktuellen Intervall der Mittelwert der gemessenen

Paketgrößen ermittelt, indem die Einzelgrößen aufaddiert und durch die Anzahl der eingetroffenen Ereignisse geteilt werden. Eine Funktion **shiftValues** verschiebt die gemessenen Werte in das jeweils nächste Feld, um das erste Feld für die neue Messung freizumachen, wie es auch in der Funktionsgleichung 6.2 gezeigt ist.

Sollte es sich nicht um ein vom Knoten versendetes Ereignis handeln, sondern um ein eingegangenes, wird der Zähler für den entsprechenden Link um eins inkrementiert und die Größe des Ereignisses wird der Größe der im bisherigen Intervall eingetroffenen Ereignisse hinzu addiert.

$$\textbf{shiftValues("1;4;2;5;5")} \longrightarrow \text{"}4; 2; 5; 5; 0\text{"} \tag{6.2}$$

6.2 Bewertung von Instanzplatzierungen

Ziel des mAhCEP-Frameworks ist eine Optimierung der Anzahl an Hops, die im CEP-System notwendig sind, um mit allen CEP-Komponenten arbeiten zu können. Formel 3.1 hat bereits erläutert, wie die Kosten für einen Link zustande kommen. Daraufhin wurde als Problemstellung die **optimale Instanzsetzung** 4.1 formuliert, an die sich die Konfiguration des CEP-Systems durch die Nutzung des mAhCEP-Frameworks annähern soll. Der hier eingeführte Algorithmus zur Bewertung von Instanzplatzierungen arbeitet dezentral auf Basis von lokalen Entscheidungen und bewertet dabei nicht den gesamten Operatorgraphen, sondern nur die CEP-Instanz, zu welcher gerade eine Bewertung ausgeführt wird. Er spart somit die Kosten für eine Kommunikation ein, die entstehen würden, wenn ein globales Wissen für die Entscheidung erforderlich wäre. Die Metrik, die den Bedarf der Energie für die Konfiguration ausdrückt, wenn die CEP-Instanz auf diesem mobilen Knoten ausgeführt werden würde, wird als Score bezeichnet, der Vorgang einer Anfrage als Scoring. Die Scores der benachbarten mobilen Knoten werden von der in diesem Abschnitt beschriebenen Komponente ermittelt.

6.2.1 Scoringtabellen

Um die von Nachbarn getroffenen Aussagen bezüglich ihrer Scores auf dem anfragenden mobilen Knoten speichern zu können, wird zu Beginn einer Abfrage für die entsprechende CEP-Instanz eine Scoringtabelle angelegt.

Diese Tabelle 6.2 besteht aus zwei Spalten. In der ersten Spalte wird der Nachbar *K'* benannt, der eine Antwort auf die Anfrage vom mobilen Knoten *K* gegeben hat. Die Sortierung erfolgt aufsteigend der Scores, die von den mobilen Knoten erzielt wurden.

Mit dem Abschluss der Scoringanfrage kann die Scoringtabelle nach der Entnahme des besten Ergebnisses wieder aufgelöst werden. Sie wird dann nicht mehr benötigt.

mobiler Knoten	Score des mobilen Knotens
644	5.225
346	5.421
13	5.469

Tabelle 6.2: ScoringTable eines Knotens mit drei Nachbarn

Algorithmus 6.2 getScores: Sammeln von Nachbarscores, um eine Instanzsetzung bewerten zu können

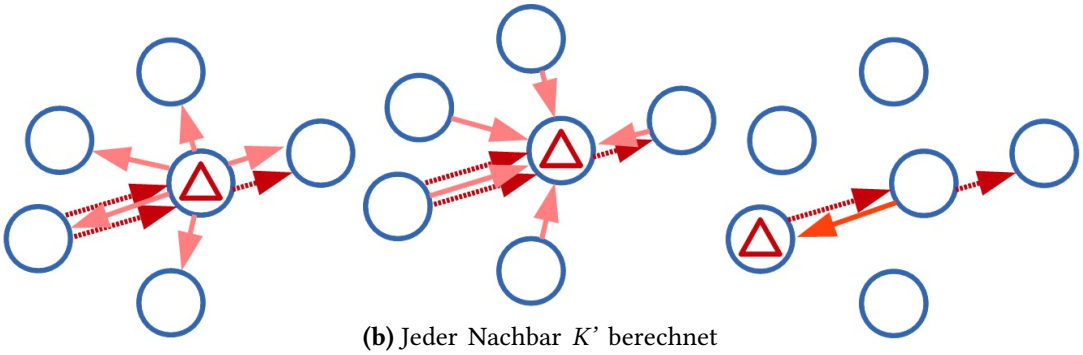
```
function getScores(Instance)
    clear(Instance.F)                    // leeren von für die Anfrage nicht benötigten Zuständen

    for each Link in Instance.In ∧ Instance.Out do
        Link.F := getAverageFrequency(Link.P, Link.G, Link.I)
        Link.S := getAverageSize(Link.P, Link.G, Link.I)
                  // die notwendigen Mittelwerte werden in die CEP-Konfiguration geschrieben
    end for
    float OwnScore := calcScore(Instance)                    // eigenen Score berechnen
    message Message := createMessage(QUERYSCORE, Instance, OwnScore)
                                                             // Kontrollnachricht erstellen
    createScoringTable(Instance.G, Instance.I)
                                                 // eindeutige Tabelle für Ergebnisse erstellen
    handleMessage(message)
    wait(timeout)                                            // Timeout abwarten
    scoringtable ScoringTable := getScoringTable(Instance.G, Instance.I)
    destroyScoringTable(Instance.G, Instance.I)
                                                 // ScoringTable zwischenspeichern und zurückgeben
    return ScoringTable
end function
```

6.2.2 Anforderung von Scores der Nachbarn

Für die Überprüfung einer Instanzplatzierung muss ein mobiler Knoten einen Vergleich zwischen sich und seinen Nachbarn ausführen können. Der Vergleich wird durch die Anfrage von Scores ermöglicht. In Algorithmus 6.2 wird der Ablauf der Anfrage von Scores an die benachbarten mobilen Knoten gezeigt.

Die Prozedur **getScores(cepinstance Instance)** wird von einem mobilen Knoten aufgerufen, wenn er für eine auf sich ausgeführte CEP-Instanz ermitteln will, ob eine Migration unter den momentanen Umständen gewinnbringend ist oder ob die CEP-Instanz auf diesem mobilen Knoten sinnvoll betrieben wird.

Der mobile Knoten entfernt zuerst die für das Scoring unwesentlichen Konfigurationsdaten aus der übergebenen CEP-Instanz, um die Nachricht zu verkleinern. Da für das Scoring die in der **StatisticTa-**



(a) Der mobile Knoten K fragt seine Nachbarn N nach ihren Scores

(b) Jeder Nachbar K' berechnet seinen Score für die betreffende CEP-Instanz und übermittelt das Ergebnis zurück an K

(c) Der mobile Knoten K entschließt sich zu einer Migration

Abbildung 6.3: Ablauf von Abfrage und Migration für eine CEP-Instanz durch das mAhCEP-Framework

ble gespeicherten aktuellen Werte erforderlich sind, wird die CEP-Konfiguration mit den vorhandenen Werten beladen.

Danach wird vom mobilen Knoten der eigene Score errechnet, der für die betriebene CEP-Instanz und den aktuellen Werten der **StatisticTable** aktuell ist. Diese Information wird zur Einsparung von Kommunikation über den drahtlosen Funkkanal benötigt. Mobile Knoten, die einen schlechteren Score erzielen, erkennen so, dass ihre Antwort dem aktuell ausführenden mobilen Knoten nicht von Nutzen ist, da dieser die CEP-Instanz nicht an einen mobilen Knoten abgibt, der einen energetischen Mehraufwand bedeuten würde. Eine Antwort kann deswegen ausbleiben.

Der mobile Knoten erstellt eine Kontrollnachricht, an welche er die Konfiguration der CEP-Instanz, die nun mit den Werten der **StatisticTable** beladen ist, anhängt. Gleichzeitig wird eine Scoringtabelle für das aktuell ausgeführte Scoring erstellt, in welche Antworten von Nachbarn eingetragen werden. Danach wird die Nachricht an das mobile Ad-hoc-Netzwerk übergeben.

Nach einem Timeout wird das beste Ergebnis ermittelt, auf dessen Basis dann die Entscheidung getroffen werden kann, ob eine Migration stattfindet, oder ob die CEP-Instanz auf diesem mobilen Knoten momentan energetisch günstig ausgeführt wird. Nicht alle Nachbarn müssen nach Ablauf des Timeout geantwortet haben. Später eintreffende Antworten werden in die Scoringtabelle nicht mehr mit einbezogen.

Da **getScores** durch einen Timeout blockiert wird, bis es die ermittelten Scores verwertet, muss diese Prozedur nebenläufig ausgeführt werden, da sonst das System blockiert ist, bis der Timeout verstrichen ist.

6.2.3 Berechnung des eigenen Scores

Bei der Anfrage der Nachbarn müssen vom anfragenden mobilen Knoten sowie von allen angefragten Nachbarn die Scores errechnet werden, die sich für eine CEP-Instanz ergeben. Die Funktion **calcScore**

Algorithmus 6.3 calcScore: Attraktivität eines Nachbarknotens für eine zu verschiebende CEP-Instanz

```
function calcScore(Instance)
    float Score := 0                                              // Score-Variable
    for each Link in Instance.In ∧ Instance.Out do                // Menge aller Links
        Score := Score + getHopCount(Link.P) ∗ Link.F ∗ Link.S    // Scoring nach 3.1
    end for
    return Score                                                  // ermittelte Score zurückgeben
end function
```

berechnet für eine ihr übergebene CEP-Instanz I den Score, den der Knoten, auf dem die Berechnung stattfindet, erzielt.

Wird ein mobiler Knoten mittels Kontrollnachricht von einem Nachbarn bezüglich seiner Score für eine CEP-Instanz angefragt, wird diese Kontrollnachricht an die Scoring-Komponente des mAhCEP-Frameworks übergeben. Dort wird der Funktion **calcScore** mit der CEP-Konfiguration ausgeführt. Diese Funktion benötigt zur Determinierung der Entfernung zu benachbarten CEP-Instanzen Zugriff auf die Routingtabelle des mobilen Ad-hoc-Netzwerkes.

In Algorithmus 6.3 ist gezeigt, wie **calcScore** den Score des mobilen Knotens K, auf welchem die Funktion ausgeführt wird, für die CEP-Instanz C bestimmt.

Es werden für alle ein- und ausgehenden Links die einzelnen Energieaufwände nach der Formel 3.1 berechnet, die für die Unterhaltung des Links erforderlich sind. Die vom mobilen Knoten benötigten Informationen sind hierbei die Hopanzahl sowie die in den Langzeitmessungen durch die StatisticTable ermittelten Werte für die Größe und die Frequenz der über diesen Link übertragenen Ereignisse. Die Summe der errechneten Aufwände entspricht dem Score des mobilen Knotens für die CEP-Instanz. Ob es sich bei dem Link um einen Ein- oder Ausgang handelt, ist für die energetische Bewertung unwesentlich, wie bereits in Kapitel 3 ausgeführt wurde.

Das insgesamt errechnete Ergebnis steht nun als der Score des mobilen Knotens K fest und wird der aufrufenden Funktion oder Prozedur zurückgegeben.

6.2.4 Eingang einer Scoringanfrage

Trifft eine Anfrage eines benachbarten mobilen Knotens bezüglich eines Scores für eine CEP-Instanz auf dem mobilen Knoten ein, muss dieser von seiner Sicht aus einen Score berechnen. Für die Ermittlung des Scores wurden die Konfigurationsdaten der CEP-Instanz und der **StatisticTable**, die für diese Aufgabe erforderlich sind, der Kontrollnachricht, die als *queryScore* bezeichnet wird, angehängt. Der mobile Knoten kann nun, wie bei der Berechnung eines Scores einer auf ihm ausgeführten CEP-Instanz einen Score für die entfernte CEP-Instanz auf Basis seiner Routentabelle und der Konfiguration aus der Kontrollnachricht berechnen.

Sollte der eigene Score nun kleiner und damit besser sein, als der Score des anfragenden mobilen Knotens, wird eine Antwortnachricht generiert, die als *replyScore* bezeichnet wird. Ist der Score schlechter oder identisch mit dem des anfragenden mobilen Knotens, wird keine Kontrollnachricht

Algorithmus 6.4 queryScore: Eingang einer Scoreanfrage

```
procedure QUERYSCORE(Message)
    float Score := CALCSCORE(Message.Instance)        // Score-Variable mit eigenem Score
    if Score < Message.Score then                     // Rückgabe falls besser
        CLEAR(Message.Instance.In)
        CLEAR(Message.Instance.Out)                   // Links für Antwort nicht notwendig
        replyscore Message := CREATEMESSAGE(REPLYSCORE, Message.Instance, Score,
         Message.Sender)
        HANDLEMESSAGE(Message)
    end if
end procedure
```

Algorithmus 6.5 gotScore: Eingang einer Scoreantwort

```
procedure GOTSCORE(Message)
    ADDSCORINGTABLE(Message.Instance.G, Message.Instance.ID,
     Message.Sender, Message.Score)
                                  // eingegangenen Score in die Scoringtabelle übernehmen
end procedure
```

erstellt, um Kommunikation, die nicht erforderlich ist, einzusparen. Der Ablauf dieser Prozedur ist in Algorithmus 6.4 beschrieben.

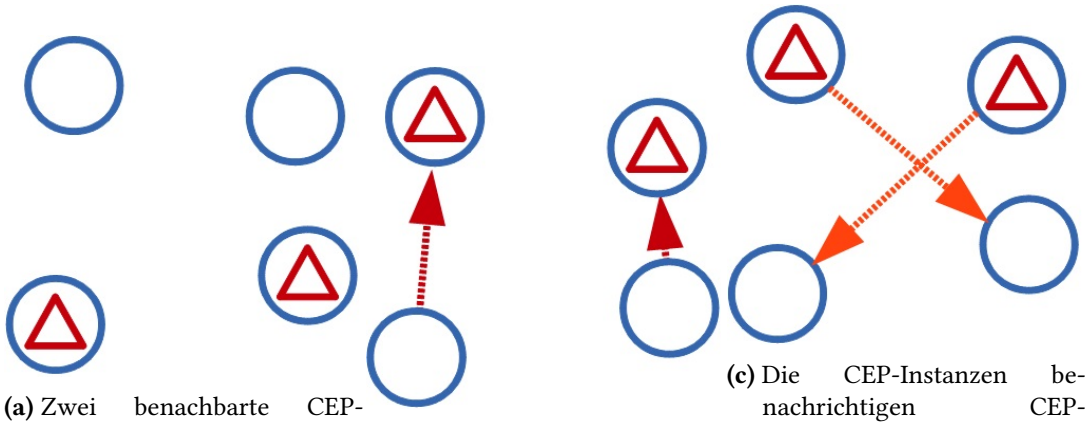
6.2.5 Eingang eines Scores von einem benachbarten mobilen Knoten

Trifft eine von einem angefragten mobilen Knoten generierte *replyScore*-Nachricht ein, kann das Ergebnis in die für diese Instanz erstellte Scoringtabelle eingetragen werden. Eine Prozedur, die diese Aufgabe ausführen kann, ist in Algorithmus 6.5 gezeigt.

6.3 Konsistenzprobleme bei einer Migration

Während dem Migrationsschritt werden die CEP-Konfiguration und die zugehörigen Daten von einem mobilen Knoten *K* auf einen benachbarten mobilen Knoten *K'* übertragen. Dieser Schritt weist mehrere Ziele auf, die vom mAhCEP-Framework erfüllt werden müssen.

- **Migration**: Die CEP-Konfiguration soll von einem mobilen Knoten *K* auf einen anderen mobilen Knoten *K'* übertragen werden.
- **Konsistenz**: Die CEP-Konfigurationen von benachbarten CEP-Instanzen sollen den neuen Ausführungsort der migrierten CEP-Instanz in ihrer Konfiguration festhalten, um generierte Ereignisse an das richtige Ziel adressieren zu können.



(a) Zwei benachbarte CEP-Instanzen entschließen sich zu einer Migration

(b) Die CEP-Instanzen werden gleichzeitig migriert

(c) Die CEP-Instanzen benachrichtigen CEP-Nachbarinstanzen über ihre neue Adresse

Abbildung 6.4: Konsistenzproblem bei gleichzeitiger Migration von benachbarten CEP-Instanzen

- **Kein Ereignisverlust**: Ereignisse, die während einer CEP-Instanz-Migration versendet werden, sollen der migrierten CEP-Instanz zugestellt werden und nicht verloren gehen.

6.3.1 Sperren der Nachbarinstanzen

Zu Beginn der Arbeit sollte ein naives Verfahren erarbeitet werden, um eine Migration durchführen zu können. Ein naives Verfahren, das die im obigen Abschnitt geforderten Punkte einhält, kann nach folgendem Ablauf funktionieren:

- **Migrationsankündigung**: Ein mobiler Knoten K benachrichtigt alle CEP-Nachbarinstanzen, dass eine Migration geplant ist. Dies geschieht durch Kontrollnachrichten des mAhCEP-Frameworks.
- **Sperrung der Nachbarinstanzen**: Die Nachbarinstanzen werden festgesetzt, das heißt, sie selbst können in dieser Zeit nicht migrieren. Ereignisse, die zur migrierenden CEP-Instanz gerichtet sind, werden vom mAhCEP-Framework oder der CEP-Instanz selbst zurückgehalten, damit diese nicht verloren gehen. Es wird eine Bestätigung von jeder CEP-Instanz versendet.
- **Abwarten der Bestätigungen**: Der mobile Knoten K wartet alle Bestätigungen von CEP-Nachbarinstanzen ab und startet mit der Migration, sobald diese eingetroffen sind.
- **Migration**: Der mobile Knoten K migriert die CEP-Instanz auf den mobilen Zielknoten K'
- **Aktualisieren der Informationen und Betriebsaufnahme**: Der neue mobile Knoten K' informiert alle CEP-Nachbarinstanzen über die neue Adresse der CEP-Instanz und erteilt Freigabe. Die CEP-Nachbarinstanzen sind wieder migrierbar und senden der migrierten CEP-Instanz Ereignisse.

Die Anforderung der Konsistenz wird in diesem Verfahren durch das Festsetzen der CEP-Nachbarinstanzen erreicht. Es ist dann nicht möglich, dass zwei CEP-Instanzen, die durch einen Link miteinander unmittelbar benachbart sind, gleichzeitig migrieren, wie in Abbildung 6.4 dargestellt. Die unmittelbare Nachbarschaft durch Links impliziert dabei nicht die unmittelbare Nachbarschaft im mobilen Ad-hoc-Netzwerk.

Die notwendige Benachrichtigung der Nachbarknoten erfolgt in beiden Fällen zu spät, wenn sich die Migration und die nicht fertiggestellte Aktualisierung der CEP-Instanzen überlappen, da die momentan migrierende CEP-Instanz *I* die Aktualisierungsnachricht der bereits migrierten CEP-Instanz *I'* nicht mehr erhält und ihrerseits dann ebenfalls bei erfolgter Migration eine Aktualisierung an die bereits im Vorfeld migrierte CEP-Instanz *I'* schickt, deren Adresse nicht mehr gültig ist. Eine Festsetzung umgeht die gleichzeitige Verschiebung, dadurch tritt das Konsistenzproblem nicht mehr auf. Durch die Pufferung von Ereignissen oder der Blockierung von Ereignisgenerierungen tritt kein Ereignisverlust auf.

Bei der Implementierung dieses naiven Verfahrens fällt auf, dass die Blockierung eines Ereignisstroms Auswirkungen auf den gesamten folgenden Operatorgraphen hat. Fehlt einer CEP-Instanz eine Eingabe, blockiert diese in der Regel so lange, bis der entsprechende Ereignisstrom fortgesetzt wird. Bei sehr großen CEP-Systemen kann bereits eine Migration dazu führen, dass ein Großteil der Konsumenten nicht bedient wird. Zudem wird durch die Sperrung von Nachbarn eine Koppelung von Komponenten erzwungen, wie sie in verteilten Systemen unerwünscht ist. CEP-Instanzen, die migriert werden sollen, sind auf freie Plätze angewiesen. Je größer die Menge von Links einer CEP-Instanz ist, desto höher ist auch die Wahrscheinlichkeit, dass sie gerade durch einen Nachbarn blockiert wird. Ebenfalls ist die Wahrscheinlichkeit für eine sinnvolle Migration bei einer CEP-Instanz mit vielen Nachbarn höher.

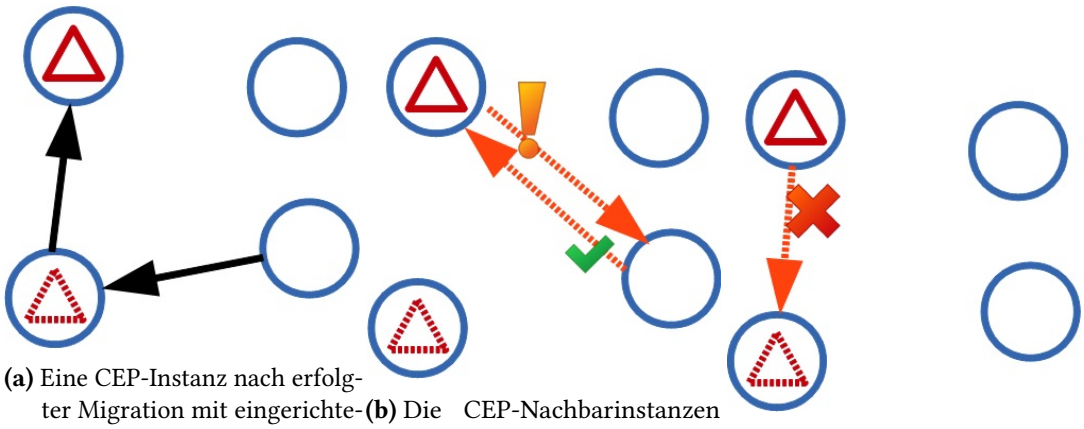
In einem Infrastrukturnetzwerk oder einem Ad-hoc-Netzwerk, in welchem sich die Topologie nicht ändert, konvergiert das System gegen das Optimum. Bei häufigen Topologieänderungen, wie sie in einem mobilen Ad-hoc-Netzwerk üblich sind, bremst dieser Mechanismus die Konvergenz, sodass das System unter Umständen zu träge wird, um an das Netzwerk zu adaptieren. Deswegen soll in dieser Arbeit nicht genauer auf diese Methodik eingegangen werden.

6.3.2 Weiterleiten von Ereignissen und Kontrollnachrichten

Um den Anforderungen, die in der Aufzählung 6.3 gestellt wurden, als auch den Anforderungen eines mobilen Ad-hoc-Netzwerkes gerecht zu werden, wurde im weiteren Verlauf dieser Arbeit ein anderes Verfahren erarbeitet, das erlaubt, Migrationen von CEP-Instanzen entkoppelt von den Zuständen von CEP-Nachbarinstanzen durchzuführen.

Das Verfahren lässt sich in folgendem Ablaufdiagramm beschreiben:

- **Migrationsentscheidung**: Der mobile Knoten *K* entschließt sich, eine CEP-Instanz auf den mobilen Knoten *K'* zu migrieren.



(a) Eine CEP-Instanz nach erfolgter Migration mit eingerichteter Route **(b)** Die CEP-Nachbarinstanzen werden aktualisiert **(c)** Die Route wird gelöscht

Abbildung 6.5: Routenerstellung und -auflösung nach einer erfolgten Migration

- **Migration und Routenerstellung**: Die CEP-Instanz wird migriert. Gleichzeitig wird auf dem mobilen Knoten *K* eine Weiterleitung zum mobilen Knoten *K'* eingerichtet. Diese Weiterleitung bezieht sich nur auf alle Links in einer bestimmten CEP-Instanz in einem bestimmten Operatorgraph.
- **Betriebsaufnahme und Korrektur**: Der neue mobile Knoten *K'* beginnt mit dem Betrieb der CEP-Instanz. Gleichzeitig benachrichtigt er alle CEP-Nachbarinstanzen, dass die CEP-Instanz nun auf dem Knoten *K'* ausgeführt wird.
- **Ändern der Nachbarkonfigurationen**: Die CEP-Nachbarinstanzen ändern ihre Einträge im CEP-System und versendet eine Bestätigungsnachricht vom Typ *acknowledgement*. Diese Nachricht dient der Verwaltung der Routenketten, die im Verlauf der Migration angelegt werden.
- **Löschen der Routen**: Sobald alle CEP-Nachbarinstanzen die Kenntnis des neuen Knoten durch eine Nachricht bestätigt haben und alle Ereignisse umgeleitet wurden, löscht jeder mobile Knoten seinen Teil der Routenkette und tritt somit aus dieser aus.

Dieses Verfahren erlaubt es, CEP-Instanzen auch dann zu migrieren, wenn sich momentan CEP-Nachbarinstanzen in einer Migration befinden. Gegenüber dem naiven Verfahren mit der Sperrung gibt es hier keine Information, in welchem Zustand sich eine CEP-Nachbarinstanz befindet, da hierzu keine Notwendigkeit besteht.

Die geforderten Ziele der Konsistenz werden durch die eingerichteten Routen erreicht, Ereignisse gehen durch die gleichzeitige Weiterleitung nicht verloren.

6.3.3 Anlegen und Löschen von Routen

Um das mobile Ad-hoc-Netzwerk nur in dem Maße zu belasten, wie es erforderlich ist, soll eine Route dann eingerichtet werden, wenn eine CEP-Instanz migriert und wieder aufgelöst werden, sobald alle CEP-Nachbarinstanzen Kenntnis vom neuen Ausführungsort haben. Das mAhCEP-Framework

Zielinstanz	Operatorgraph	Von	Zu	Quellinstanzen	Seit	Bis
11	23	22	23	32	312	355
11	23	22	23	44	444	467
11	23	22	23	31	325	444
11	23	22	23	43	523	574

Tabelle 6.3: Routeninformation auf einem mobilen Knoten

hinterlässt für die CEP-Instanz am nicht mehr aktuellen mobilen Knoten *K'* einen Zeiger auf den neuen Knoten *K*. Dieser Zeiger beinhaltet alle Elemente, wie sie in Tabelle 6.3 aufgeführt werden.

Die Spalten “Zielinstanz“ und “Operatorgraph“ werden benötigt, um eindeutig identifizieren zu können, ob eingegangene Ereignisse und Nachrichten weitergeleitet werden müssen. Die Spalte “Von“ wird benötigt wenn mehrere Migrationen unmittelbar hintereinander erfolgen und so eine Routenkette, wie sie in Abbildung 6.6 gezeigt wird, entsteht. Die Spalte “Zu“ zeigt auf den Knoten, auf dem die CEP-Instanz ausgeführt wird, zu welcher die Route führt. Wurden mehrere Migrationen direkt hintereinander ausgeführt, zeigt dieses Feld auf den nächsten mobilen Knoten innerhalb der Routenkette. “Seit“ zeigt an, von welcher Sequenznummer an die Route gültig wurde. Die Spalte “Bis“ deklariert die Sequenznummer, bis zu welcher der mobile Knoten für Weiterleitungen zuständig ist.

Wurde eine Migration durchgeführt, öffnet das mAhCEP-Framework auf dem alten und ebenfalls auf dem neuen mobilen Knoten eine solche Tabelle, um das Management der Routenkette entkoppelt von den Optimierungen des mAhCEP-Frameworks ausführen zu können. Die Tabellenspalte “Zu“ ist beim letzten mobilen Knoten einer Routenkette nicht belegt. Der mobile Knoten weiß dann, dass er das Ereignis dem mAhCEP-Framework zur Zählung und dem CEP-System zur Verarbeitung weitergeben muss. Gleichzeitig muss eine Verwaltung der Sequenznummern stattfinden. Die Verwaltung muss sicherstellen, dass innerhalb des angegebenen Bereichs alle Ereignisse und Kontrollnachrichten den mobilen Knoten passiert haben. Erst dann wird die Route ungültig.

Da bei einer Migration der Fall auftreten kann, dass ein Ereignisstrom zum neuen mobilen Knoten schneller aufgebaut ist, als der alte Ereignisstrom zum passieren des alten Weges gebraucht hat, ist es möglich, dass eine Kontrollnachricht innerhalb der Routenkette, die ein *acknowledgement* repräsentiert, früher ankommt. Trifft diese Nachricht ein, befindet sich in ihr ebenfalls eine Sequenznummer *n*. Für den empfangenden mobilen Knoten ist diese Sequenznummer die erste Nachricht, die er erhält. Sein Vorgänger muss demnach alle Sequenznummern, die kleiner als diese Sequenznummer *n* sind, weiterleiten. Dies teilt er mit einer Kontrollnachricht des Typs *sequencemax* seinem Vorgängerknoten mit. Ein Vorgängerknoten muss in diesem Fall existieren, da sonst kein *acknowledgement* versendet worden wäre. Der Vorgänger weiß nun Bescheid, für welches Fenster an Nachrichten er zuständig ist. Sobald das gesamte Sequenznummernfenster abgearbeitet wurde, kann der mobile Knoten, wenn er keinen Vorgänger hat, aus der Routenkette aussteigen. Es ist nur dann möglich, da ein mobiler Knoten keine Einsicht hat, ob seine Vorgänger bereits alle Nachrichten, für die sie zuständig sind, zu ihm weitergeleitet haben. Erst, wenn der mobile Knoten vorgängerlos geworden ist, kann er aus der Routenkette ausscheiden und seinem Nachfolger eine Kontrollnachricht vom Typ *discard* senden. Dieser weiß nun, dass er selbst der letzte mobile Knoten innerhalb einer Routenkette geworden ist.

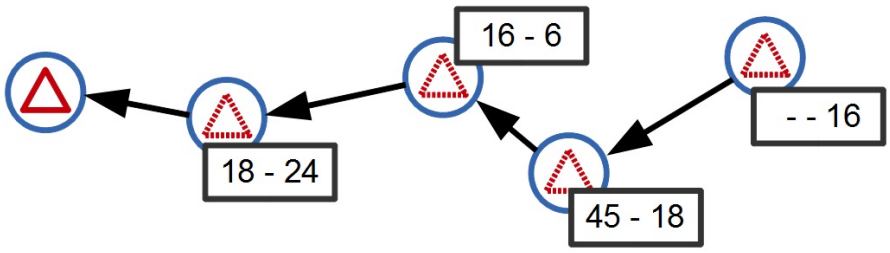
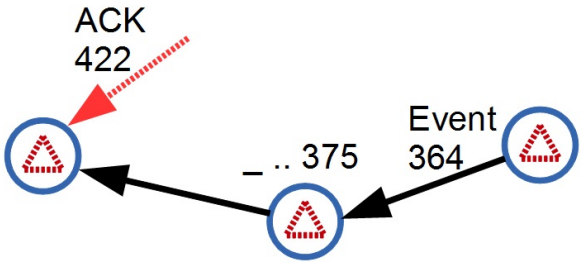
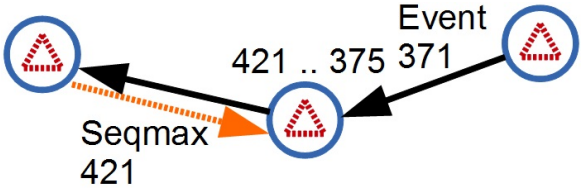


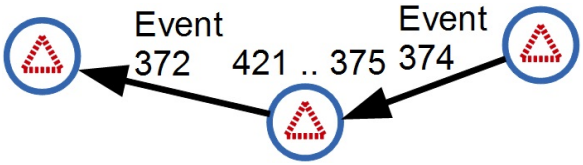
Abbildung 6.6: Mehrere Migrationen führen zu einer Routenkette



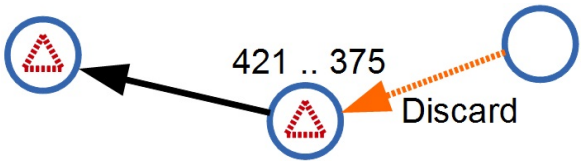
(a) Ein mobiler Knoten erhält eine Bestätigung für eine Aktualisierung nach einer Migration



(b) Der mobile Knoten sendet an seinen Vorgänger in der Routenkette eine Nachricht, die seinen Bereich für Weiterleitungen definiert



(c) Der letzte mobile Knoten in der Routenkette sendet das letzte Paket, für das er zuständig ist



(d) Danach tritt er aus der Routenkette aus und sein Nachfolger wird zum Ende der Routenkette

Abbildung 6.7: Ein mobiler Knoten sendet seinem Vorgänger Informationen über seine Zuständigkeit

Dieses Verfahren beginnt so oft von neuem, bis kein Nachfolger mehr existiert, da der mobile Knoten selbst auch die zugehörige CEP-Instanz unterhält.

Dieses Vorgehen entkoppelt die Routenkette von den Migrationen und erlaubt eine Verkürzung auf das minimal notwendige Maß. Zudem wird die Auflösung der Routenkette nicht zentral, sondern verteilt koordiniert, sodass der Aufwand für das mAhCEP-Framework, auf welchem die CEP-Instanz ausgeführt wird, klein gehalten wird. Das Verfahren wird in Abbildung 6.7 in einem Ausschnitt dargestellt.

6.3.4 Anlegen von Routen bei einer Migration

Migriert eine CEP-Instanz von einem mobilen Knoten *K* auf einen anderen mobilen Knoten *K'*, wird auf dem Knoten *K'* für jeden Link der CEP-Instanz des Operatorgraphen jeweils eine Route eingerichtet.

- **createRoute(graph Graph, identifier Identity, identifier To, int minseq, identifier From)**:

  Diese Prozedur erstellt auf dem mobilen Knoten einen Eintrag in der Routentabelle für die durch die Parameter *Graph* und *Identity* betitelte CEP-Instanz, die an den mobilen Knoten gerichtet wird, der mit *To* bezeichnet ist. Bei einer Weiterleitung auf sich selbst wird das Feld mit einem leeren Identifier belegt. Durch den Parameter *minseq* wird die Sequenznummer angegeben, ab welcher der mobile Knoten für eine Weiterleitung zuständig ist. Wenn bereits eine Weiterleitung auf dem aktuellen mobilen Knoten existiert (in diesem Fall ist es eine Weiterleitung auf sich selbst, um die Routenkette selbstverwaltend zu halten), wird die minimale Sequenznummer nicht ersetzt, da der mobile Knoten noch nicht alle Nachrichten vermittelt hat, für die er zuständig ist. Hätte er bereits alle Nachrichten vermittelt, wäre kein Eintrag vorzufinden. Die korrekte Behandlung ist eine Vereinigung der beiden Sequenznummernmengen. Der Parameter *From* wird bei einer Weiterleitung auf sich selbst gesetzt, damit für die Routenkette entschieden werden kann, wer der Vorgänger des aktuellen mobilen Knotens ist. Dieser Parameter wird nur bei der eingehenden Migration, nicht aber bei der ausgehenden Migration gesetzt. Bei einer ausgehenden Migration können zwei Fälle auftreten: Es existiert noch keine Routenkette. Dann gibt es auch keinen Vorgänger. Im zweiten Fall existiert bereits eine Routenkette. Da bei der Migration bereits ein Verweis erstellt wurde, ist dieses Feld schon belegt und muss nicht geändert werden.

6.3.5 Verwaltung der Routenkette

In diesem Abschnitt wird auf die verschiedenen Verwaltungsschritte einer Routenkette eingegangen.

Eingehendes Acknowledgement Migriert eine CEP-Instanz und sendet eine Nachricht vom Typ *announcement* an die benachbarten CEP-Instanzen, antworten diese CEP-Instanzen dem mobilen Knoten mit einem *acknowledgement*. Diese Nachricht ist nicht an die Komponente für die Migration, sondern an die Komponente für die Routenketten gerichtet, da mit dieser Nachricht die Routenketten verwaltet werden. Migriert eine CEP-Instanz weiter und trifft das *acknowledgement* erst später am betreffenden mobilen Knoten ein, ändert dies nichts an seiner Behandlung.

In Algorithmus 6.6 ist das Prinzip gezeigt, mit welchem eine eingehende Aktualisierungsbestätigung abgearbeitet wird. Zunächst erstellt der mobile Knoten eine Nachricht an seinen Vorgänger, womit diesem die höchste Sequenznummer, für die er zuständig ist, mitgeteilt wird. Diese Sequenznummer ist um eins kleiner, als die Aktualisierungsnachricht. Folgend wird die eigene Untergrenze mit der Prozedur **setLowerBound** auf die nächste eintreffende Sequenznummer gesetzt.

Maximum setzen Trifft ein *acknowledgement* bei einem Nachfolger in der Routenkette ein, erhält sein Vorgänger eine Nachricht, die seine Zuständigkeit zur Weiterleitung von Nachrichten definiert. Diese Zuständigkeit ist in einer Nachricht mit dem Typ *sequencemax* enthalten. Sie dauert bis zur Nachricht an, die eine Sequenznummer aufweist, die um eins kleiner ist, als das eingetroffene *acknowledgement*.

Algorithmus 6.6 incomingAck: Ein mobiler Knoten bestätigt eine Aktualisierung

```
procedure INCOMINGACK(acknowledgement Message)
    message Out := CREATEMESSAGE(SEQUENCEMAX, Message.Sequencenumber − 1,
     GETPREDECESSOR(Message.G, Message.I, Message.Link))
                          // Erstellung einer Nachricht für Vorgänger mit Zuständigkeitsrahmen
    HANDLEMESSAGE(Out)

    SETLOWERBOUND(Message.G, Message.I,
     Message.Link, Message.Sequencenumber + 1)
                                  // Setzen der eigenen unteren Grenze an die nächste Nachricht
end procedure
```

Algorithmus 6.7 checkEnd: Ein mobiler Knoten prüft, ob er aus einer Routenkette ausscheiden kann

```
procedure CHECKEND
    for each Line in Routechaintable do          // alle Zeilen der Routenkettentabelle prüfen
        if NULL(GETPREDECESSOR(Line)) and then NOLEFTPACKAGES then
                                    // falls kein Vorgänger und alle Nachrichten abgearbeitet
            REMOVELINE(Line)                                              // Zeile entfernen
            discard Message := CREATEMESSAGE(DISCARD, Line.G, Line.I, Line.L,
             GETSUCCESSOR(Line))
            HANDLEMESSAGE(Message)                        // aus Routenkette ausscheiden
        end if
    end for
end procedure
```

Ende auflösen Durch die Verwaltung und die Grenzen, die einem mobilen Knoten durch die Tabelleneinträge "Seit" und "Bis" gegeben werden, kann er entscheiden, ob er die Nachrichten, für die er zuständig ist, bereits weitergeleitet hat. Wenn alle Nachrichten weitergeleitet wurden, kann der Knoten aus der Routenkette ausscheiden, wenn er keine Vorgänger mehr hat. Ein Knoten kann nur für sich entscheiden, ob er alle Nachrichten weitergeleitet hat, die er selbst auf direktem Weg empfängt. Treffen Nachrichten bei Vorgängern später ein, als beim aktuellen mobilen Knoten, werden diese auf der Routenkette über ihn weitergeleitet. Ein Ausscheiden ist folglich erst möglich, wenn keine Vorgänger mehr existieren. Existieren noch Vorgänger in der Routenkette, muss der Eintrag so lange gehalten werden, bis die Vorgänger aus der Routenkette ausgeschieden sind.

Ein mobiler Knoten überprüft deswegen seine Routenkettentabelle auf Einträge, in welchen alle Nachrichten abgearbeitet wurden und zu welchen kein Vorgänger existiert. Diese Einträge werden, wie in Algorithmus 6.7 gezeigt, entfernt und der Nachfolger informiert, dass nun er der letzte Teilnehmer der Routenkette ist. Sobald bei diesem Teilnehmer alle Nachrichten vermittelt wurden, verfährt er analog und scheidet als letzter Teilnehmer der Routenkette aus. Mobile Knoten, die noch keine Grenzen definiert haben, können nicht entscheiden, ob alle Nachrichten bereits abgearbeitet wurden. Der letzte mobile Knoten der Kette, auf dem auch die CEP-Instanz ausgeführt wird, kann die Routenkette verlassen, sobald kein Vorgänger mehr existiert. Er kann jedoch auch Teil der Routenkette bleiben, da

Algorithmus 6.8 destroyRoute: Eine Routenkette wird aufgelöst

```
procedure destroyRoute(Graph, Identifier, Link)
    if not Null(getPredecessor(Graph, Identifier, Link)) then
        destroy Message := createMessage(DESTROY, Graph, Identifier, Link,
         getPredecessor(Graph, Identifier, Link))
    end if                                          // Vorgänger benachrichtigen
    removeLine(Graph, Identifier, Link)             // Eintrag entfernen
end procedure
```

sich keine Nachteile ergeben. Wird die CEP-Instanz migriert, muss kein neuer Eintrag erstellt werden, da lediglich einige Einträge im alten Eintrag modifiziert werden.

Eine Prozedur nach *checkEnd* kann periodisch oder beim Eintreffen einer Nachricht ausgeführt werden, um alte Routenketteneinträge zu überprüfen.

Vorgänger entfernen Wenn ein mobiler Knoten aus einer Routenkette ausscheidet, sendet er seinem Nachfolger eine Nachricht vom Typ *discard*. Diese Nachricht löst auf dem Nachfolger die Löschung des Vorgängers aus, sodass der mobile Knoten nun der letzte Teilnehmer der Routenkette ist.

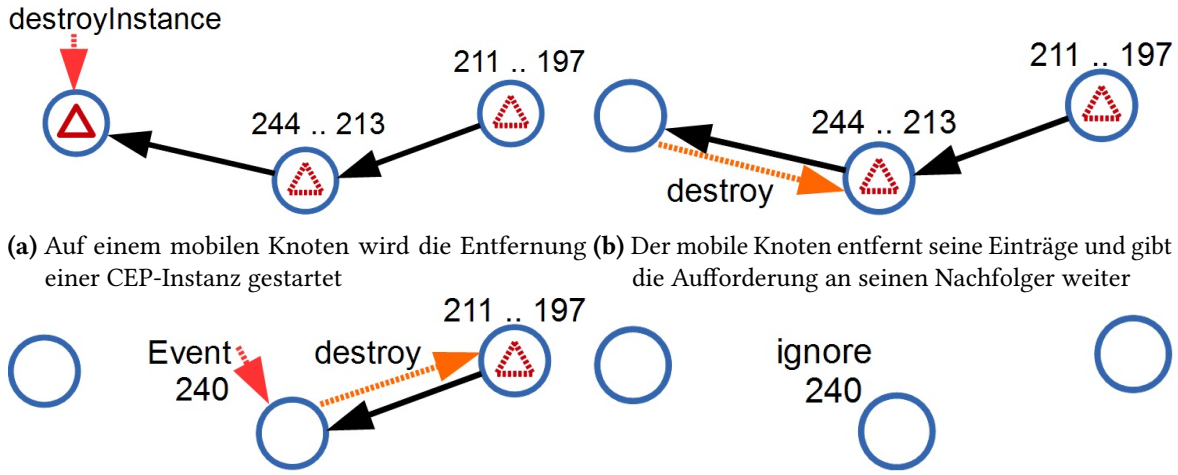
Routenkette auflösen Möchte der Administrator einen Operatorgraphen aus dem System entfernen, wie es in folgenden Abschnitten unter 6.5.4 beschrieben ist, kann die Routenkette aufgelöst werden, ohne noch nicht zugestellte Pakete zu beachten. Bei der Entfernung einer Routenkette sendet der mobile Knoten, auf dem die CEP-Instanz ausgeführt wird, eine Nachricht vom Typ *destroy* an seine Nachfolger, wie es in in Abbildung 6.8 gezeigt wird. Die Nachfolger können daraufhin ihre zugehörigen Routenketteneinträge entfernen und ihre jeweiligen Nachfolger ebenfalls diese Nachricht zukommen lassen. Der Vorgang stoppt beim letzten Teilnehmer der Routenkette, da dieser keinen Nachfolger hat. Der Vorgang ist in Algorithmus 6.8 beschrieben.

6.4 Migration

Dieser Abschnitt beschreibt die Migration von CEP-Instanzen und die periodische Evaluierung von Instanzsetzungen sowie die Aktualisierung von CEP-Instanzen, bei denen die Migration eines Nachbarn stattgefunden hat.

6.4.1 Prüfen der aktuellen Instanzsetzungen

Um die Instanzsetzung der CEP-Instanzen, die auf dem mobilen Knoten ausgeführt werden, regelmäßig kontrollieren zu können, muss eine Routine ausgeführt werden, die periodisch ein Scoring der CEP-Instanzen auslöst, die auf dem mobilen Knoten betrieben werden.



(a) Auf einem mobilen Knoten wird die Entfernung einer CEP-Instanz gestartet

(b) Der mobile Knoten entfernt seine Einträge und gibt die Aufforderung an seinen Nachfolger weiter

(c) Der Nachfolger löscht ebenfalls seine Einträge und informiert seinen Nachfolger. Dabei treffen weiterhin Nachrichten aus dem Netzwerk ein.

(d) Diese Nachrichten werden ignoriert, da weder eine Zielinstanz noch eine Weiterleitung auf dem mobilen Knoten existiert

Abbildung 6.8: Eine Routenkette wird aufgelöst

Algorithmus 6.9 checkConfig: Prüfung der aktuellen Platzierung

```
procedure CHECKCONFIG(Graph, Identity)
    scoringtable ScoringTable
    GETSCORES(GETCONFIGURATION(Graph, Identity))                    // Scores abfragen
    if Empty(ScoringTable) then
        SCHEDULE(CHECKCONFIG(Graph, Identity), scheduletime)
                    // falls Scoringtabelle leer, momentan Platzierung okay, später erneut prüfen
    else
        MIGRATEOUT(Graph, Identity, ScoringTable(0, 0)) // falls nicht leer, Migration auslösen
    end if
end procedure
```

Innerhalb dieser Routine muss eine Scoringtabelle für diese CEP-Instanz angelegt und daraufhin die Anfrage der Nachbarn (im vorausgegangenen Abschnitt mit **getScores** bezeichnet) gestartet werden. Der Ablauf der Routine ist in Algorithmus 6.9 dargestellt.

Nach Ablauf des Timeouts für die Antworten der Nachbarn wird die Scoringtabelle kontrolliert. Ist die Scoringtabelle leer, konnte kein Nachbar des mobilen Knotens einen besseren Score aufweisen als der mobile Knoten selbst. Dann wird die erneute Ausführung der Platzierungsüberprüfung geplant. Der Parameter *scheduletime* ist ein Designparameter, der es erlaubt, die Intervalle, in welchen eine Scoreanfrage ausgelöst wird, einzustellen.

Sollte die Tabelle Einträge enthalten, so repräsentieren alle Einträge ausnahmslos eine mögliche, immer bessere Platzierung, als die momentan vorhandene. Dann wird der mobile Knoten, der das Scoring ausgeführt hat, eine Migration der CEP-Instanz anstoßen. Sollte die nach Scoringtabelle nicht ganz optimale Platzierung tolerierbar sein, ist es möglich, einen Migrationsschwellenwert zu

Algorithmus 6.10 migrateOut: Ein mobiler Knoten versendet eine CEP-Instanz

```
procedure MIGRATEOUT(G, I, To)
    instance Instance := DESTROYINSTANCE(G, I)
    for each Link in Instance.In ∧ Instance.Out do
        Link.S := GETALLSIZES(Instance.P, Instance.G, Instance.I)
        Link.F := GETALLFREQUENCIES(Instance.P, Instance.G, Instance.I)
                                        // Größen und Frequenzen der Links an Instanz anhängen
        DESTROYLINK(Instance.P, Instance.G, Instance.I)
                                                    // und Link aus StatisticTable löschen
    end for
    migrate Message := CREATEMESSAGE(MIGRATE, Instance, To)
    HANDLEMESSAGE(Message)          // CEP-Instanz in Nachricht packen und versenden
    for each Line in Instance.In ∧ Instance.Out do
        CREATEROUTE(Line.G, Line.I, Line.L, To, lastSeqNo)          // Route erstellen
    end for
end procedure
```

definieren. Ein mobiler Knoten entschließt sich dann erst dazu, eine CEP-Instanz zu migrieren, wenn ein anderer mobiler Knoten einen Score liefert, der mindestens um eine bestimmte Differenz oder ein bestimmtes Verhältnis besser ist.

Die Nebenläufigkeit, die von der vormals eingeführten Prozedur **getScores** wegen des Timeout-Intervalls gefordert wird, wird durch die nebenläufige Ausführung der überprüfenden Routine erreicht, die durch die Planung auf dem mobilen Knoten entsteht.

6.4.2 Ausgehende Migration

Sollte sich durch die periodische Prüfung der Instanzsetzung ein benachbarter mobiler Knoten zur Betreibung einer CEP-Instanz als besser herausstellen, muss die CEP-Instanz auf dem aktuellen mobilen Knoten entfernt werden und dem neuen mobilen Knoten per Kontrollnachricht zugestellt werden. Der Ablauf dieses Vorgangs ist in Algorithmus 6.10 dargestellt.

Zunächst muss die CEP-Instanz vom CEP-System geholt und entfernt werden, anschließend müssen die durch die **StatisticTable** erhobenen Werte zusammen mit der CEP-Instanz in einer Kontrollnachricht zum neuen mobilen Knoten versendet werden. Gleichzeitig muss eine Route eingerichtet werden, damit die Nachrichten an den neuen Empfänger weitergeleitet werden können und die Konsistenz sichergestellt wird. Die von der **StatisticTable** erhobenen Daten können vom alten mobilen Knoten gelöscht werden, da diese dort nicht mehr benötigt werden.

6.4.3 Eingehende Migration

Trifft an einem mobilen Knoten eine Kontrollnachricht mit einer CEP-Instanz ein, hat der empfangende mobile Knoten zuvor ein angefragtes Scoring als bestmöglicher Kandidat für sich entschieden. Er erhält deswegen die CEP-Instanz, die er nun ausführen soll.

Zum Aufsetzen der CEP-Instanz werden alle mitgegebenen statistischen Daten in die **StatisticTable** übernommen. Danach erstellt das mAhCEP-Framework auf Basis der Konfiguration in der empfangenen Kontrollnachricht die CEP-Instanz, indem sie dem CEP-System übergeben wird.

Gleichzeitig muss der neue Ausführungsort der CEP-Instanz den benachbarten CEP-Instanzen mitgeteilt werden, damit diese ihre Konfiguration aktualisieren und die Routenketten so schnell wie möglich abgebaut werden können. Ereignisse und Kontrollnachrichten, die vor der Aussendung des *announcement* versendet wurden, erreichen die CEP-Instanz durch die bei der ausgehenden Migration eingerichteten Routenkette.

Um die Routenkette abzuschließen, wird auf dem aktuellen mobilen Knoten ebenfalls eine Route auf sich selbst eingerichtet. Sobald vom Vorgänger ein *discard* eingetroffen ist, existiert keine Routenkette mehr und alle benachbarten CEP-Instanzen sind auf dem aktuellen Stand. Dieses Vorgehen entkoppelt die Zuständigkeit der Wartung von einer durch das mAhCEP-Framework zu steuernden Routine. Stattdessen werden diese Routen bei Nachrichteneingang geprüft. Sollte der mobile Knoten sich zu einer weiteren Migration entschließen, wird der momentane Routenzustand zurückgesetzt, da erneut alle Nachrichten und Ereignisse umgeleitet werden müssen.

Um die Platzierung der CEP-Instanz regelmäßig zu prüfen, wird als letzter Schritt in Algorithmus 6.11 die Prozedur **checkConfig** geplant, welche die periodische Prüfung durchführt. Der Parameter *scheduletime* ist ein Designparameter, der es erlaubt, die Intervalle zwischen den Proben einzustellen.

6.4.4 Migration publizieren

Wird eine CEP-Instanz von einem mobilen Knoten K auf einen anderen mobilen Knoten K' migriert, müssen alle CEP-Nachbarinstanzen, die in der Menge $Instance.In \cup Instance.Out$ enthalten sind, über diesen Wechsel informiert werden, um Konsistenz zu bewahren. Eine rein reaktive Meldung, das heißt sobald eine Nachricht geroutet werden musste, wird eine Aktualisierungsnachricht versendet, ist prinzipiell möglich. Dies führt aber zu verlängerten Routenketten, was sich bei sehr dynamischen mobilen Ad-hoc-Netzwerken besonders bemerkbar macht, da viele Migrationen erfolgen und demnach viele Routen eingerichtet werden. Die Routenketten können erst dann aufgelöst werden, wenn alle CEP-Nachbarinstanzen eines Routenkettenanfangs über die Änderung im Bilde sind. Werden über einen Link nur sehr selten Ereignisse versendet, kann es unter Umständen sehr lange dauern, bis die Aktualisierung erfolgt. Knoten, die Teilnehmer einer Routenkette sind, können nicht aus dem System aussteigen. Prinzipiell wäre es möglich, die zwei Teilnehmer, die vor und nach dem aussteigenden Teilnehmer in der Routenkette sind, als Nachbarn zu konfigurieren. CEP-Nachbarinstanzen, die diesen Knoten aber als aktuellen Knoten kennen, werden dann nicht vollständig mit dem migrierten Knoten kommunizieren können, was zu Inkonsistenzen führen kann, da eine CEP-Nachbarmigration verloren gehen kann. Der Fall, dass ein Knoten innerhalb der Routenkette nicht mehr notwendig ist, aber Teilnehmer nach diesem Knoten noch benötigt werden, wurde im Teil 6.3.3 bereits ausgeschlossen. Es

Algorithmus 6.11 migrateIn: Eine CEP-Instanz migriert auf den mobilen Knoten

```
procedure MIGRATEIN(Message)

    for each Link in Message.Instance.In ∧ Message.Instance.Out do
        CREATELINK(Link.P, Message.Instance.G, Message.Instance.I, Link.F, Link.S)
                                        // erstelle zu jedem Link eine neue Statistik
    end for
    CREATEINSTANCE(Message.Instance)                          // Starte CEP-Instanz
    if Type(Message) = migrate then
        ANNOUNCE(Message.Instance)              // neuen Knoten anderen bekannt machen
        for each Line in Instance.In ∧ Instance.Out do
            CREATEROUTE(Line.G, Line.I, Line.L, NULL, lastSeqNo, Message.Sender)    //
Route erstellen
        end for                                                   // Route auf mich selbst
        SCHEDULE(CHECKCONFIG(Message.Graph, Message.Identity, scheduletime))
                                        // Scoringabfrage planen, falls Migration
    end if
end procedure
```

Algorithmus 6.12 announce: Ein mobiler Knoten gibt die neue Konfiguration bekannt

```
procedure ANNOUNCE(Instance)
    announcement Message

    for each Link in Instance.In do
        CREATEMESSAGE(ANNOUNCEMENT, IN, Instance.G, Instance.I,
        Link.I, Link.P)
        HANDLEMESSAGE(Message)
    end for

    for each Link in Instance.Out do
        CREATEMESSAGE(ANNOUNCEMENT, OUT, Instance.G, Instance.I,
        Link.I, Link.P)
        HANDLEMESSAGE(Message)              // sende an jeden Partner eine Aktualisierung
    end for
end procedure
```

ist also zwingend notwendig, Routenketten so lange wie notwendig, aber so kurz wie möglich, zu unterhalten.

Die Kontrollnachricht über eine erfolgte Migration wird *announcement* genannt.

Algorithmus 6.13 updateConfig: Ein migrierter Nachbar wird aktualisiert

```
procedure UPDATECONFIG(Message)
    instance Instance
    Instance.G := Message.G
    Instance.I := Message.I                          // ID und Graph der CEP-Instanz angeben
    if Message.Link = In then
        ADD(Instance.Out, Message.G, Message.I, Message.Sender)
    else
        ADD(Instance.In, Message.G, Message.I, Message.Sender)
    end if
      // füge entweder zu den ausgehenden oder eingehenden Links ein Objekt in die Menge ein,
                                                     // welches in der CEP-Instanz ersetzt werden soll
    MODIFYINSTANCE(Instance)                                    // modifiziere CEP-Instanz
    message Message := CREATEMESSAGE(ACKNOWLEDGEMENT, Message.Sender,
     Message.G, Message.I, Link.I)
    HANDLEMESSAGE(Message)                                          // sende Bestätigung
end procedure
```

6.4.5 Empfangene Änderungen einpflegen

Empfängt ein mobiler Knoten eine Kontrollnachricht vom Typ *announcement*, muss für eine CEP-Instanz eine Aktualisierung erfolgen, da eine CEP-Nachbarinstanz migriert wurde.

Die erforderlichen Daten zum Aufruf der vom CEP-System bereitgestellten Prozedur **modifyInstance** werden aus der empfangenen Kontrollnachricht extrahiert und in eine Variable geladen, da die Instanz nur insoweit modifiziert wird, wie eine übergebene Konfiguration beladen ist.

Nach dem Aufruf von **modifyInstance** wird eine bestätigende Kontrollnachricht an den migrierten Knoten versendet, welche der Verwaltung der Routenketten dient und als *acknowledgement* bezeichnet wird.

6.5 Bei- und Austritt von mobilen Knoten und Operatorgraphen

In einem realen mobilen Ad-hoc-Netzwerk werden laufend neue Knoten dem Netzwerk beitreten oder es verlassen. Die Betreiber neuer Knoten fordern neue Informationen von der Nutzergemeinde, was es erforderlich macht, Operatorgraphen nicht nur am Anfang, sondern auch zur Laufzeit des Systems definieren und auf das System aufsetzen zu können.

6.5.1 Knotenbeitritt

Da das mAhCEP-Framework ein lose gekoppeltes System ist, welches keine Anmeldung von seinen Nutzern fordert, muss ein Knotenbeitritt nur in den unter dem mAhCEP-Framework liegenden

Schichten bekannt gegeben werden. Hierzu zählt mindestens der Routingalgorithmus, der im mobilen Ad-hoc-Netzwerk verwendet wird. Dem Routingalgorithmus müssen teilnehmende Knoten bekannt gemacht werden, da sie sich sonst unsichtbar innerhalb des mobilen Ad-hoc-Netzwerks aufhalten.

Sobald ein mobiler Knoten Teilnehmer des Netzwerks ist und mit der Ausführung des mAhCEP-Frameworks beginnt, kann er auf durch das drahtlose Kommunikationsmedium von seinen Nachbarn empfangene Kontrollnachrichten antworten. Falls durch seinen Beitritt ein Ereignisstrom in der Übertragung günstiger geworden ist, wird er innerhalb der unteren Schichten Teilnehmer des Routings für die entsprechenden Pakete, was auch zu Migrationen in Teilen des Operatorgraphen führen kann, dessen Ereignisse davon betroffen sind. Dies passiert wegen der Routingmethodik im mobilen Ad-hoc-Netzwerk und völlig unabhängig vom mAhCEP-Framework oder dem CEP-System, was auf dem mobilen Knoten eventuell ausgeführt wird.

Die Aktivität des mAhCEP-Frameworks beginnt bei einer Scoringanfrage, welche der mobile Knoten, der neu hinzugekommen ist, beantworten kann.

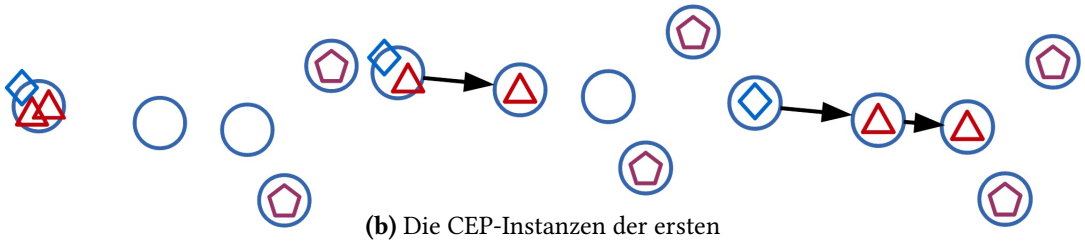
6.5.2 Knotenaustritt

Möchte ein mobiler Knoten das mobile Ad-hoc-Netzwerk und damit das CEP-System und das mAhCEP-Framework verlassen, muss die Konsistenz bestehender Operatorgraphen sichergestellt werden. Führt der mobile Knoten weder eine CEP-Instanz aus noch ist er Teilnehmer einer Routenkette des mAhCEP-Frameworks, kann er aus der Sicht dieser Schichten das Netzwerk verlassen, ohne dass eine weitere Behandlung notwendig ist.

Sollte der mobile Knoten Teilnehmer einer Routenkette sein, muss er innerhalb des Netzwerks verweilen, bis die Routenkette auf ihm aufgelöst werden kann. Er kann erst dann das Netzwerk verlassen, da es sonst zu Inkonsistenzen kommen kann.

Führt der mobile Knoten eine CEP-Instanz aus, muss er die CEP-Instanz migrieren. Wurde die CEP-Instanz migriert, muss der mobile Knoten warten, bis die sich dann aufbauende Routenkette bis zu ihm aufgelöst hat, wie eben erläutert wurde. Die Migration erfolgt ebenfalls auf Basis des Scoring, jedoch muss der Knoten zum Austritt ein hohes fiktives Scoring für sich wählen, damit seine Nachbarn auf die Scoringanfrage antworten können. Es kann nicht der Fall auftreten, dass niemals ein mobiler Knoten auf die Scoringanfrage antwortet und die CEP-Instanz in der Folge nicht migrieren kann, da ein mobiles Ad-hoc-Netzwerk innerhalb eines Operatorgraphen zusammenhängend sein muss, damit dieser Operatorgraph funktioniert. Somit gibt es einen Nachbar, dessen Score nur unterboten werden muss, damit eine Migration vom Optimum weg stattfinden kann. Um zu vermeiden, dass in der Zwischenzeit neue CEP-Instanzen auf den mobilen Knoten migrieren, darf er auf Scoringanfragen nicht mehr antworten, wenn er das mobile Ad-hoc-Netzwerk verlassen möchte. Dies kann durch eine Zustandsvariable sichergestellt werden.

Bei festen CEP-Instanzen, die normalerweise Sensoren und Konsumenten repräsentieren, muss davon ausgegangen werden, dass der mobile Knoten nicht aus dem Netzwerk austreten möchte. Tritt er dennoch aus dem Netzwerk aus, wird der betreffende Operatorgraph nicht mehr funktionieren. Ob sich dies durch eine Migration lösen lässt, muss im Einzelfall entschieden werden. Die Daten eines anderen Teilnehmers können für die Konsumenten innerhalb des Operatorgraphen völlig



(a) Der Operatorgraph wird in anfänglichem Zustand hinzugefügt

(b) Die CEP-Instanzen der ersten Schicht nach den Konsumenten wandern Richtung Optimum

(c) Die CEP-Instanzen der nachfolgenden Schichten wandern ebenfalls Richtung Optimum

Abbildung 6.9: Ein neuer Operatorgraph wird dem CEP-System hinzugefügt

uninteressant sein, sodass eine Migration keinen Nutzen bringt. Dies ist nicht Teil dieser Arbeit und wird dementsprechend nicht bewertet.

6.5.3 Hinzufügen von Operatorgraphen

Das Hinzufügen eines Operatorgraphen kann auf zwei unterschiedliche Weisen erfolgen. Die erste Möglichkeit ist, dass der Administrator die anfängliche Optimierung übernimmt und die mobilen Knoten nach seinen Vorstellungen mit den verschiedenen CEP-Instanzen belädt.

Die zweite Möglichkeit ist, dass die komplette Optimierung durch das mAhCEP-Framework geschieht, wie es in Abbildung 6.9 gezeigt wird. Der Administrator des CEP-Systems braucht hierbei nur die Konsumenten und Sensoren an den entsprechenden mobilen Knoten zu starten und die Operatoren einem der Konsumenten oder Sensoren beizuheften. In der Beispielabbildung wurden die CEP-Instanzen, die Operatoren repräsentieren, beim Konsumenten gestartet und wandern in den darauffolgenden Schritten in Richtung des momentanen Optimums.

Wichtig beim Hinzufügen eines Operatorgraphen ist es, in der Schicht der Konsumenten zu beginnen, damit das CEP-System keine Nachrichten in das mobile Ad-hoc-Netzwerk versendet, bevor ein Empfänger diese annehmen kann. Prinzipiell kann auch bei den Sensoren angefangen werden, jedoch belasten die unnötig versendeten Nachrichten das mobile Ad-hoc-Netzwerk, ohne dass daraus ein Nutzen gezogen wird. Zudem muss jede CEP-Instanz aus Gründen der Konsistenz zunächst als fest in das mobile Ad-hoc-Netzwerk hinzugefügt werden, damit mobile Knoten nicht mit einer Scoringanfrage beginnen und migrieren, bevor alle CEP-Nachbarinstanzen die Aktualisierungsnachricht empfangen können. Um die CEP-Instanzen dann zu losen CEP-Instanzen zu ändern, muss auf den mobilen Knoten lediglich die Prozedur **schedule** mit der Routine, die die Instanzsetzung periodisch überprüft, beladen gestartet werden, welche wiederum mit der entsprechenden CEP-Instanz beladen werden muss, die zu einer losen CEP-Instanz geändert werden soll. Eine Prozedur, die diese Funktionalität anbietet, ist in Algorithmus 6.14 gezeigt. Das Beladen eines mobilen Knotens wird durch eine Nachricht an den mobilen Knotens mit einem Nachrichtentyp *start* realisiert, der ebenfalls der Prozedur **migrateIn** übergeben wird, jedoch dazu führt, dass keine Prüfung der Platzierung gestartet wird.

Algorithmus 6.14 stopFixed: Eine feste CEP-Instanz wird zu einer losen CEP-Instanz geändert

procedure STOPFIXED($Graph$, $Identity$)
  SCHEDULE(CHECKCONFIG($Graph$, $Identity$), $scheduletime$)
end procedure

Algorithmus 6.15 shutdownInstance: Eine CEP-Instanz und ihre Routen werden beendet

procedure SHUTDOWNINSTANCE($Graph$, $Identity$)
  MIGRATEOUT($Graph$, $Identity$)
end procedure

6.5.4 Entfernen von Operatorgraphen

Analog zum Hinzufügen von Operatorgraphen sollte das Entfernen eines Operatorgraphen an der Schicht entstehen, die die ersten Ereignisse versendet, damit das mobile Ad-hoc-Netzwerk nicht versucht, Nachrichten zuzustellen, für die kein Empfänger mehr existiert. Wie bei einer ausgehenden Migration müssen die Statistiken und die CEP-Instanz entfernt werden. Im Gegensatz zu einer Migration muss aber eine Route nicht erstellt, sondern die unter Umständen bestehende Route muss aufgelöst werden, da sonst die Möglichkeit besteht, dass sie nicht mehr freigegeben wird, wenn keine Kontrollnachrichten mehr von mobilen Knoten eintreffen, die in den Operatorgraphschichten liegen, die den Konsumenten zugewandt ist. Ein Administrator darf einen Operatorgraphen also nicht über das CEP-System direkt beenden, da sonst die Konfiguration des mAhCEP-Frameworks zurückbleibt.

Um eine Entfernung einer CEP-Instanz auszulösen, kann die vorausgehend vorgestellte Prozedur **migrateOut** modifiziert werden, sodass diese ohne Angabe eines Zieles verwendet werden kann. Dann wird weder eine Nachricht generiert noch eine Route erstellt, sondern die Route wird aufgelöst. Eine Prozedur, die diese Funktionalität nach außen anbietet, kann wie Algorithmus 6.15 aussehen.

7 Implementierung

In diesem Kapitel werden Prozeduren und Funktionen definiert, die für eine Implementierung der im vorausgegangenen Kapitel eingeführten Algorithmen erforderlich sind. Diese umfassen die Komponenten der **StatisticTable**, des **Scoring**, der **Routenketten** sowie der Nachrichtenbehandlung durch das mAhCEP-Framework.

7.1 Funktionen und Prozeduren für die StatisticTable

Dieser Abschnitt zeigt, welche Prozeduren und Funktionen implementiert werden müssen, um mit der StatisticTable umgehen zu können, wie es in den vorausgegangenen Kapiteln in den Algorithmen vorausgesetzt wird.

7.1.1 Werteentnahme aus der StatisticTable

Um von anderen Prozeduren oder Funktionen auf die **StatisticTable** zugreifen zu können, müssen vier Funktionen implementiert werden, mit welchen auf die **StatisticTable** zugegriffen werden kann. Diese Funktionen durchsuchen die Tabelle nach dem entsprechenden Datensatz und geben das Ergebnis zurück.

Die Funktionen werden im Folgenden definiert. Für jede Funktion wird ein Beispiel aufgeführt, das sich auf Tabelle 6.1 bezieht.

- **getAverageFrequency(partner P, graph G, identity I) return Float**

  Diese Funktion gibt den ermittelten Frequenzmittelwert zurück.

  $$\textbf{getAverageFrequency(13,1,9;10)} \longrightarrow \text{”5”} \tag{7.1}$$

- **getAverageSize(partner P, graph G, identity I) return Float**

  Diese Funktion gibt die ermittelte durchschnittliche Größe zurück.

  $$\textbf{getAverageSize(13,1,9;10)} \longrightarrow \text{”3”} \tag{7.2}$$

- **getAllFrequencies(partner P, graph G, identity I) return Float()**

  Diese Funktion übergibt alle gemessenen Frequenzen, das heißt, sowohl die gemessene Durchschnittsfrequenz als auch die *m* letzten Ereignisse, werden als kommagetrennte Werte zurückgegeben.

$$\textbf{getAllFrequencies(13,1,9;10)} \longrightarrow \text{"}5; 5; 5; 5; 5\text{"} \tag{7.3}$$

- **getAllSizes(partner P, graph G, identity I) return Float()**

  Diese Funktion übergibt alle gemessenen Größen. Sowohl die Durchschnittsgröße wie auch die *n* letzten Werte werden als kommagetrennte Werte zurückgegeben.

$$\textbf{getAllSizes(13,1,9;10)} \longrightarrow \text{"}3; 3; 3; 1; 2\text{"} \tag{7.4}$$

Alle Funktionen benötigen zur Ermittlung des gewünschten Links als Eingabewerte die genauen Partner und die exakte Benennung der CEP-Instanz, die durch die Angabe des Operatorgraphen und der zugehörigen CEP-Instanz-Identifikation geschieht.

7.1.2 Einträge anlegen und löschen

Bei einer ausgehenden Migration oder Auflösung einer CEP-Instanz auf einem mobilen Knoten ist es sinnvoll, alte, nicht mehr benötigte Einträge in der **StatisticTable** zu entfernen. Die eingehende Migration oder Erstellung neuer CEP-Instanzen auf einem Knoten erfordern ein Anlegen der entsprechenden Einträge in der **StatisticTable**.

Diese Funktionalität wird von folgenden Prozeduren angeboten:

- **createLink(partner P, graph G, identity I, float() F, float() S)**

  Diese Prozedur erstellt unter dem angegebenen Namen einen Eintrag in der Tabelle, wobei die statistisch erfassten, übergebenen Daten direkt eingefügt werden.

- **destroyLink(partner P, graph G, identity I)**

  Diese Prozedur löscht den betreffenden Datensatz in der Tabelle.

7.2 Funktionen und Prozeduren für Scoringtabellen

Dieser Abschnitt zeigt, welche Prozeduren und Funktionen implementiert werden müssen, um für Anfragen an benachbarte mobile Knoten die Algorithmen verwenden zu können, die in dieser Arbeit eingeführt wurden.

7.2.1 Erstellen, Schreiben und Löschen von Scoringtabellen

Für eine Scoringanfrage müssen die für die Scoreermittlung zuständigen Prozeduren auf die Scoringtabellen zugreifen können. Hierzu müssen Prozeduren und Funktionen implementiert werden, mit denen das Umgehen von Scoringtabellen realisiert wird. Im Folgenden werden die benötigten Funktionen definiert.

- **createScoringTable(graph G, identity I)**

  Diese Prozedur erstellt für die durch die Parameter angegebene CEP-Instanz eine Scoringtabelle. Die Instanz wird, wie in Algorithmus 6.2 sichtbar ist, durch den Operatorgraphen und die Identifikation der Instanz innerhalb des Operatorgraphen bezeichnet.

- **addScoringTable(graph G, identity I, string Node, float Score)**

  Diese Prozedur schreibt in die benannte Scoringtabelle zum Knoten, der im Übergabeparameter Node bezeichnet wird, die im Parameter Score übergebene Score in die Scoringtabelle. Die Scoringtabelle wird automatisch aufsteigend sortiert. Sollte eine bezeichnete Scoringtabelle nicht existieren, wird der versuchte Eintrag verworfen. Dies kann bei abgelaufener Abfragezeit passieren. Bei gleichem Score wird der neu eingegangene mobile Knoten mit dem mobilen Knoten mit identischem Score zufällig sortiert.

- **getScoringTable(graph G, identity I) return scoringtable(,)**

  Diese Funktion gibt die für den Parameter momentan aktuelle Scoringtabelle zurück.

- **destroyScoringTable(graph G, identity I)**

  Diese Prozedur löscht die durch die Parameter bezeichnete Scoringtabelle.

7.3 Funktionen und Prozeduren für Routenketten

Innerhalb der Routenketten werden verschiedene Kontrollnachrichten ausgetauscht, die unterschiedliche Auswirkungen auf den mobilen Knoten und die dort betriebenen Routenketten haben. In diesem Abschnitt werden die Algorithmen gezeigt, die die entsprechenden Änderungen an der Routenkette vornehmen.

7.3.1 Eingang einer Sequenznummerngrenze

Trifft ein Acknowledgement auf einem Vorgänger innerhalb der Routenkette ein, wird eine Kontrollnachricht des Typs *sequencemax* versendet. Diese kann durch Algorithmus 7.1 behandelt werden. Dieser setzt für die entsprechende Routenkette das Zuständigkeitsmaximum auf die entsprechende Sequenznummer.

Algorithmus 7.1 incomingMax: Ein mobiler Knoten erhält seinen Zuständigkeitsrahmen

```
procedure INCOMINGMAX(sequencemax Message)
    SETUPPERBOUND(Message.G, Message.I, Message.Link, Message.Sequencenumber)
                                                    // maximale Sequenznummer definieren
end procedure
```

Algorithmus 7.2 incomingDiscard: Ein mobiler Knoten wird letzter Teilnehmer einer Routenkette

```
procedure INCOMINGDISCARD(discard Message)
    DESTROYPREDECESSOR(Message.G, Message.I, Message.L)        // entferne Vorgänger
end procedure
```

7.3.2 Löschen eines Vorgängers

Hat ein mobiler Knoten innerhalb einer Routenkette alle Sequenznummern weitergeleitet, für die er zuständig ist und ist er gleichzeitig letzter Teilnehmer der Routenkette, kann er diese verlassen und seinem Nachfolger ein *discard* senden. Diese Kontrollnachricht muss vom Nachfolger behandelt werden. Ein einfacher Algorithmus, wie er in Algorithmus 7.2 gezeigt wird, erfüllt diese Aufgabe.

7.4 Nachrichtenfluss im mAhCEP-Framework

Dieser Abschnitt fasst alle in dieser Ausarbeitung verwendeten Kontrollnachrichten zusammen, die zwischen den mobilen Knoten, die das mAhCEP-Framework ausführen, generiert und versendet werden. Weiterhin wird auf die Behandlung eingehender Nachrichten eingegangen. Diese müssen unterschiedlichen Teilen des mAhCEP-Frameworks zugestellt werden.

7.4.1 Kontrollnachrichten

Um die durch das mAhCEP-Framework getroffenen Optimierungen realisieren zu können, sind verschiedene Typen von Kontrollnachrichten erforderlich, die im Laufe dieser Arbeit eingeführt wurden. In diesem Abschnitt werden alle Kontrollnachrichten aufgelistet und deren Funktion und Inhalt erklärt.

- *start*: Instance

  Diese Nachricht wird an einen mobilen Knoten gerichtet, wenn eine CEP-Instanz auf ihm gestartet werden soll. Sie ist beladen mit der zu startenden CEP-Instanz mit allen sie definierenden Daten. Die Kontrollnachricht löst eine Einrichtung der CEP-Instanz auf dem Zielknoten aus, wobei die CEP-Instanz fest, also nicht verschiebbar, eingerichtet wird. Dies dient dazu, den gesamten Operatorgraphen einrichten zu können, ohne dass einzelne CEP-Instanzen bereits migrieren und es so zu Inkonsistenzen kommt. Eine Entkoppelung von mobilen Knoten kann wie in Algorithmus 6.14 gezeigt erfolgen. Die Kontrollnachricht wird nicht von Teilnehmern

des mAhCEP-Frameworks versendet, sondern vom Administrator des CEP-Systems, wenn ein Operatorgraph hinzugefügt werden soll.

- *stop*: Graph, Instance

  Diese Nachricht führt auf einem mobilen Knoten zum Stop und der Entfernung der im Nachrichtenteil *Graph* und *Instance* betitelten CEP-Instanz. Diese Nachricht wird vom Administrator des CEP-Systems versendet, wenn ein Operatorgraph entfernt werden soll.

- *queryscore*: Instance, Ownscore

  Möchte ein mobiler Knoten Scores zu einer bestimmten CEP-Instanz abfragen, versendet er eine Nachricht vom Typ *queryscore*, welche mit den zur Berechnung notwendigen Daten und seiner eigenen Score beladen wird. Zu den notwendigen Daten gehören insbesondere die ermittelten Daten in der **StatisticTable**. Die eigene Score wird benötigt, um mobilen Knoten mitzuteilen, ob ihre Antwort notwendig ist oder unterbleiben kann, falls ihr Score schlechter ist als der Score des anfragenden mobilen Knoten. Würde der eigene Score nicht mitgesendet, kann ein Nachbarknoten nicht entscheiden, ob seine Antwort notwendig ist oder nicht. Jeder Nachbar müsste somit eine Antwort aussenden. Dies führt vor allem bei einer momentan optimalen Instanzplatzierung zu erhöhtem, nicht notwendigem Energieaufwand.

- *replyscore*: Instance, Score

  Empfängt ein mobiler Knoten eine Scoringanfrage von einem anderen mobilen Knoten, ist bereit, eine CEP-Instanz zu migrieren und erzielt für diese CEP-Instanz einen besseren Score als der anfragende mobile Knoten, versendet er eine Nachricht vom Typ *replyscore*. Die Bedingung, eine Antwort nur dann zu versenden, wenn der eigene Score besser ist als die des anfragenden mobilen Knoten, führt zu einer Verbesserung des Energieaufwandes. Um es dem anfragenden mobilen Knoten zu ermöglichen, die CEP-Instanz zu identifizieren, für die der Score berechnet wurde, wird die Nachricht neben dem Score auch mit der Instanz beladen. Hier reicht die Identifikation und der Operatorgraph der CEP-Instanz aus.

- *migrate*: Instance

  Hat sich ein mobiler Knoten entschlossen, eine CEP-Instanz zu migrieren, erstellt er eine Nachricht vom Typ *migrate*, welche er mit der CEP-Instanz belädt, die migriert werden soll. Die CEP-Instanz wird mit den in der **StatisticTable** ermittelten Werten beladen, da ein neuer Knoten sonst einige Zeit benötigen würde, um eine zuverlässige Voraussage über das Verhalten von CEP-Nachbarinstanzen treffen zu können.

- *announcement*: Graph, Identifier, Linkidentifier

  Wenn die Migration einer CEP-Instanz abgeschlossen ist, müssen CEP-Nachbarinstanzen über die erfolgte Migration informiert werden. Um eine Konsistenz zu erreichen, werden sowohl ein- als auch ausgehende Links benachrichtigt, damit erstellte Routenketten unabhängig von der Ereignisfrequenz schnell entfernt werden können. Die Nachricht wird mit dem Graphen, der Identifikation der CEP-Instanz und dem Linkidentifier beladen, damit benachrichtigte Knoten wissen, um welche CEP-Instanz es sich handelt. Auf der Gegenseite entspricht der neue Linkpartner dem Absender der Nachricht, weswegen dieser nicht gesondert in der Nachricht ausgewiesen werden muss.

- *acknowledgement*: Graph, Identifier, Sequencenumber

  Wenn nach einer erfolgten Migration einer CEP-Nachbarinstanz ein Knoten eine Nachricht des Typs *announcement* erhält, um seinen Linkpartner aktualisieren zu können, muss zur Verwaltung der Routenketten eine Bestätigung des Empfangs an den absendenden mobilen Knoten versendet werden. Damit der mobile Knoten nicht bloß den Absender des *acknowledgement*s kennt, wird die Nachricht mit dem Operatorgraphen sowie der Identifikation der CEP-Instanz beladen. Migriert eine CEP-Instanz sehr häufig hintereinander, ist es für die Konsistenz prinzipiell nur notwendig, das zuletzt versendete *announcement* zu bestätigen, da dieses die aktuellste Aktualisierung bestätigt und eine Routenkette dann entfernt werden kann. Bei der Forderung, alle Ereignisse zuzustellen, darf eine Routenkette erst bei Eingang aller Nachrichten, für die der Teil der Routenkette zuständig ist, erfolgen. Dies kann durch die Angabe der Sequenznummer entschieden werden. Ein später ausgesendetes *acknowledgement* kann bei einer Migration durch bessere Bedingungen schneller am Ziel sein, als ein früher ausgesendetes. Eine Entfernung würde ohne Sequenznummern dazu führen, dass später eintreffende Ereignisse in früheren Punkten der Routenkette verloren gehen. Ein Vorteil wäre eine Reduzierung der Belastung der mobilen Knoten, die durch die längere Aufrechterhaltung der Routenketten produziert wird.

- *sequencemax*: Graph, Identifier, Linkidentifier, Sequenznummer

  Diese Nachricht wird in einer Routenkette von einem mobilen Knoten an einen Vorgänger weitergeleitet, wenn er eine Nachricht vom Typ *acknowledgement* erhält. Der mobile Knoten weiß mit dem Erhalt der Nachricht, dass sein Vorgänger in der Routenkette nur bis zu der Sequenznummer, die um eins kleiner ist als die Sequenznummer des *acknowledgement*, zuständig ist. Dies wird dem Vorgänger mitgeteilt, damit dieser entscheiden kann, wann er alle Nachrichten, für die er zuständig ist, weitergeleitet hat und aus der Routenkette ausscheiden kann.

- *discard*: Graph, Identifier, Linkidentifier

  Hat ein mobiler Knoten, der Teilnehmer einer Routenkette ist, alle Nachrichten, für die er zuständig ist, weitergeleitet und besitzt selbst keine Vorgänger mehr, für die er eventuell noch eine Weiterleitung darstellt, kann er die Routenkette verlassen. Wenn die Bedingungen erfüllt werden, sendet er seinem Nachfolger in der Routenkette eine Nachricht vom Typ *discard*, mit welchem der Nachfolger zum Ende der Routenkette wird, da die Nachricht das Löschen des Vorgängers veranlasst.

- *destroy*: Graph, Identifier, Linkidentifier

  Soll ein Operatorgraph aus dem CEP-System entfernt werden, müssen die Routenketten ordnungsgemäß entfernt werden. Verbleibt eine Routenkette auf einem mobilen Knoten, wird es ihm unmöglich, während des Betriebs das mobile Ad-hoc-Netzwerk zu verlassen, da nicht entschieden werden kann, ob die Einträge noch aktuell oder bereits veraltet sind. Entschließt sich der Administrator des CEP-Systems, dass ein Operatorgraph entfernt werden soll, werden durch eine Migration erzeugte Routenketten durch ein *discard* vom Vorgänger entfernt. Dies wird innerhalb der Routenkette so lange wiederholt, bis alle Teilnehmer aus der Routenkette ausgeschieden sind.

- *event*: Ereignisdaten, Graph, Identifier

  Eine Nachricht vom Typ *event* wird vom CEP-System generiert und enthält Ereignisse, die von einer CEP-Instanz zu einer anderen CEP-Instanz versendet werden. Um die **StatisticTable** unterhalten zu können, hat das mAhCEP-Framework Zugriff auf die Werte, durch die die CEP-Instanz eindeutig identifiziert und charakterisiert werden kann.

7.4.2 Nachrichtenübergabe zwischen dem mAhCEP-Framework und den anderen Schichten

Um das mAhCEP-Framework für das CEP-System transparent gestalten zu können, wurde es so aufgebaut, dass es die mAhCEP-Framework-Schicht, die unter ihm liegt, wie das mobile Ad-hoc-Netzwerk behandeln kann. Die gesamte Kommunikation zwischen den Teilen des CEP-Systems auf den verschiedenen Knoten läuft über die Prozedur handleMessage ab. Das CEP-System übergibt Ereignisse, die an andere Knoten adressiert sind, dem mAhCEP-Framework, welches diese zur Zustellung an den Zielknoten an das mobile Ad-hoc-Netzwerk weiterleitet.

Am mobilen Knoten eintreffende Nachrichten können, wenn sie **handleMessage** erreichen, einen der folgenden zwei Nachrichtentypen aufweisen:

- **Ereignis**

  Ereignisse werden von CEP-Komponenten erzeugt und zwischen diesen innerhalb eines Operatorgraphen, wie er in Abbildung 2.2 dargestellt wird, ausgetauscht.

- **Kontrollnachrichten**

  werden von verschiedenen Instanzen des mAhCEP-Frameworks zwischen den Knoten ausgetauscht, um die Funktionalität zur Verfügung stellen zu können.

Treffen Nachrichten am mobilen Knoten ein, werden sie dem mAhCEP-Framework übergeben. Dieses unterscheiden zwischen Kontrollnachrichten zwischen den mAhCEP-Frameworks auf den verschiedenen mobilen Knoten und Ereignissen, die das CEP-System erwartet und leitet diese entsprechend an das mAhCEP-Framework oder die nächst höhere Schicht, in welcher sich das CEP-System befindet, weiter. Sollte das Ziel durch eine Migration auf einen anderen Knoten verschoben worden sein, wird dies berücksichtigt. **handleMessage** fragt dazu zunächst über eine Funktion **getRouteChain** ab, ob ein Routenketteneintrag existiert. Falls ein Eintrag existiert, muss die Nachricht an den mobilen Knoten umgeleitet werden, der durch den Eintrag bezeichnet wird.

Zusätzlich werden das Eintreffen oder das Versenden und die Größe eines Ereignisses der Prozedur **messageStats** übergeben. Dies ist notwendig, um das Energiemodell, wie es im Abschnitt 3.4 definiert wurde, bestimmen zu können.

Wie in Algorithmus 7.3 ersichtlich ist, überprüft **handleMessage** zunächst, ob der mobile Knoten selbst der Absender ist. Trifft dies zu, kann die Nachricht ohne weitere Behandlung an das mobile Ad-hoc-Netzwerk weitergegeben werden. Sollte es sich um ein Ereignis handeln, muss dieses zur Zählung zusätzlich an **messageStats** übergeben werden. Sollte nicht der eigene mobile Knoten der Absender sein, ist die Nachricht durch das mobile Ad-hoc-Netzwerk zugestellt worden. Dann wird überprüft,

ob es sich um ein Ereignis handelt. Abhängig davon, ob ein Routenketteneintrag existiert oder nicht, wird das Ereignis direkt dem CEP-System zugestellt oder an einen anderen mobilen Knoten in der Routenkette weitergeleitet. Sollte es sich um eine Kontrollnachricht handeln und der mobile Knoten Teilnehmer einer Routenkette sein, muss unterschieden werden, ob es sich um eine Kontrollnachricht für die Routenkette oder oder für den mobilen Knoten, der die zugehörige CEP-Instanz ausführt, handelt. Entsprechend wird die Kontrollnachricht direkt zugestellt oder weitergeleitet.

Kontrollnachrichten, die wie in Abbildung 5.2 vom mAhCEP-Framework an das CEP-System gesendet werden, verlaufen nur unidirektional, weswegen keine Betrachtung des Nachrichtenflusses durch **handleMessage** erforderlich ist, da die Kontrollnachrichten von den Prozeduren und Funktionen direkt aufgerufen werden können.

7.4.3 Verteilung der Kontrollnachrichten an die Komponenten des mAhCEP-Frameworks

Die Prozedur **mahCep** wird immer dann aufgerufen, wenn eine Nachricht von **handleMessage** als Kontrollnachricht für das mAhCEP-Framework identifiziert wird. Die Aufgabe von **mahCep** ist die richtige Zustellung der Nachrichten an die entsprechenden Komponenten des mAhCEP-Frameworks. Die existierenden Kontrollnachrichten wurden bereits im Abschnitt 7.4.1 eingeführt, eine Übersicht über die verschiedenen Komponenten des mAhCEP-Frameworks findet sich in Abbildung 5.1. Die entsprechenden Komponenten wurden im Laufe der Arbeit eingeführt.

7.4.4 Transparenz zwischen dem mobile Ad-hoc-Netzwerk und dem CEP-System

Um das mAhCEP-Framework für die umgebenden Schichten transparent aufzubauen, werden zwei Prozeduren 7.5 und 7.6 angeboten, die vom CEP-System oder dem mobilen Ad-hoc-Netzwerk aufgerufen werden, um Ereignisse der jeweils anderen Schicht zu übergeben.

Algorithmus 7.5 sendMessage erhält eine Nachricht vom CEP-System und übergibt sie an das mAhCEP-Framework

procedure SENDMESSAGE($Message$)
    HANDLEMESSAGE($Message$)
end procedure

Algorithmus 7.6 receiveMessage erhält eine Nachricht vom mobilen Ad-hoc-Netzwerk und übergibt sie an das mAhCEP-Framework

procedure RECEIVEMESSAGE($Message$)
    HANDLEMESSAGE($Message$)
end procedure

Algorithmus 7.3 handleMessage verarbeitet ein- und ausgehenden Kontrollnachrichten und Ereignisse

```
procedure HANDLEMESSAGE(Message)
    if Message.Sender = Me.ID then                    // prüfen, ob der Knoten selbst Absender ist
        if Type(Message) = event then
            MESSAGESTATS(Message)                     // Übergabe des Ereignisses an die Zählung
        end if
        SENDMESSAGE(Message)                          // Nachricht an Ad-hoc-Netzwerk
    else                                              // ist Knoten selbst nicht der Absender, ist er Empfänger
        identifier Identity := GETROUTECHAIN(Message.G, Message.I, Message.L)
                                                      // Identität der empfangenden CEP-Instanz erhalten
        if Type(Message) = Event then
            if NULL(Identity) then
                                                      // prüfen, ob Ereignis und ob keine Routenkette
                MESSAGESTATS(Message)                 // Übergabe des Ereignisses an die Zählung
                RECEIVEMESSAGE(Message)               // Ereignis an CEP-System
            else
                                                      // prüfen, ob Ereignis oder Kontrollnachricht und ob Routenkette
                SETNEWDESTINATION(Message, Identity)
                SENDMESSAGE(Message)                  // Ereignis an CEP-System
            end if
        else
            if NULL(Identity) then
                                                      // prüfen, ob Routenkette
                MAHCEP(Message)                       // Kontrollnachricht an mAhCEP
            else
                if Type(Message) = acknowledgement OR
                    sequencemax OR discard OR destroy then
                    MAHCEP(Message)                   // Kontrollnachricht an Routenkette
                else
                    SETNEWDESTINATION(Message, Identity)
                    SENDMESSAGE(Message)              // Kontrollnachricht weiterleiten
                end if
            end if
        end if
    end if
end procedure
```

Algorithmus 7.4 mahCep: Eingang von Kontrollnachrichten

```
procedure mahCep(Message)
    if Type(Message) = start then
        start(stopFixed(Message.G, Message.I))
                                        // Entkoppelung einer CEP-Instanz von mobilen Knoten
    end if
    if Type(Message) = stop then
        start(migrateOut(Message.G, Message.I))
                                        // Beendigung einer CEP-Instanz und ihrer Routenkette
    end if
    if Type(Message) = queryscore then
        if not Leave then
            start(queryScore(Message))
        end if
                                // Berechnung der Score dieses mobilen Knotens für eine CEP-Instanz
                                        // falls er nicht das mobile Ad-hoc-Netzwerk verlassen will
    end if
    if Type(Message) = replyscore then
        start(addScoringTable(Message.Instance.G, Message.Instance.ID,
          Message.Sender, Message.Score))
                                // eingegangene Score in die entsprechende Scoringtabelle eintragen
    end if
    if Type(Message) = migrate then
        start(migrateIn(Message))                                   // eingehende Migration
    end if
    if Type(Message) = announcement then
        start(updateConfig(Message))
                                        // CEP-Nachbarmigration in CEP-Konfiguration eintragen
    end if
    if Type(Message) = acknowledgement then
        start(incomingAck(Message))                     // Bestätigung der Aktualisierung
    end if
    if Type(Message) = sequencemax then
        start(incomingMax(Message))             // Weiterleitungszuständigkeit festsetzen
    end if
    if Type(Message) = discard then
        start(incomingDiscard(Message))                 // Vorgänger verlässt Routenkette
    end if
    if Type(Message) = destroy then
        start(destroyRoute(Message.Graph, Message.Identifier, Message.Link))
                                        // CEP-Instanz wird beendet und Routenkette gelöscht
    end if
    if Type(Message) = Event then
        start(messageStats(Message))                    // Ereigniszählung StatisticTable
    end if
end procedure
```

8 Evaluation

In diesem Kapitel werden die Evaluationsergebnisse vorgestellt, die bei der Simulation der in dieser Arbeit vorgestellten Algorithmen erzielt wurden. Die Simulation wurde mit dem OMNet++ Simulations-Framework realisiert, das in diesem Kapitel kurz vorgestellt wird.

8.1 OMNeT++ Simulations-Framework und INET-Framework

OMNeT++ ist ein Simulations-Framework, welches akademischen und privaten Nutzern erlaubt, unter der Open-Source-Lizenz Simulationen von Netzwerken durchzuführen. Es ist die Modellierung von Protokollen bis hin zu Hardware möglich. OMNeT++ ist in C++ programmiert, erlaubt dem Nutzer aber neben der Implementierung in C++ ebenfalls, auf andere Programmiersprachen, wie C# oder Java, auszuweichen. Weitere Informationen finden sich in [V+01].

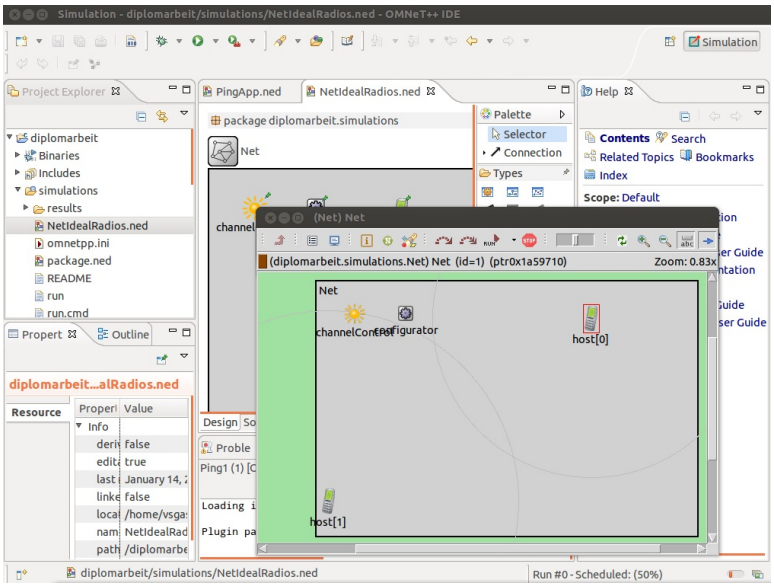


Abbildung 8.1: Das Simulations-Framework von OMNeT++

Das INET-Framework ist ein Open-Source-Projekt [Ine], das ein Paket mit vorgefertigten Modellen zur Simulation in der OMNeT++ Simulationsumgebung enthält. Es stehen unter anderem Protokolle

wie TCP, UDP, IP, IPv6 und Protokolle für drahtlose Kommunikation - zum Beispiel mit 802.11 - zur Verfügung. Eine Simulation von Ad-hoc-Netzwerken ist mit diesem Paket ebenfalls möglich.

8.2 Kosten für den Empfang einer Nachricht

Durch die Angabe von [RZ07], dass ein mobiler Knoten, der im empfangenden Zustand ist, 80 % des Energiebedarfs einer Aussendung zum Empfang aufwenden muss, kann der Energiebedarf als Score modelliert werden, wenn mehrere mobile Knoten die Aussendung eines einzelnen mobilen Knotens empfangen, wie es bei Scoringanfragen der Fall ist. In Gleichung 8.1 wird gezeigt, wie sich die Gesamtkosten für einen Hop im Score zusammensetzen. Nach Auflösung der Gleichung wird als Ergebnis erhalten, dass ein sendender Knoten Kosten von 56 % innerhalb eines Hops verursacht, während ein empfangender Knoten mit 44 % an den Kosten beteiligt ist.

$$Hop = x + 0,8 * x \tag{8.1}$$

Die Gesamtkosten, die in der Evaluierung für die Aussendung von Broadcasts und Scoringanfragen berechnet werden, setzen sich aus den Kosten für einen sendenden mobilen Knoten sowie den Kosten für die am Empfang beteiligte Anzahl an mobilen Knoten zusammen. So kann der Energieaufwand, der bei der Migration und den damit verbundenen Schritten beim Scoring entsteht sowie das Fluten, welches mit den Routenketten verglichen werden soll, als Score ausgedrückt und damit eine Abschätzung gegeben werden, ab wie vielen ausgeführten Korrelationen eine Optimierung der Instanzsetzung sinnvoll ist.

8.3 Simulationsszenarien

Die Evaluation wurde innerhalb verschiedener Szenarien durchgeführt, um verschiedene Aspekte des Algorithmus beleuchten zu können. Die Simulationsszenarien decken verschiedene Änderungen innerhalb der Umgebung ab, in welcher das mAhCEP-Framework, das in dieser Arbeit entwickelt wurde, betrieben wird. Es werden die verschiedene Platzierung, die Variation der Netzwerkgröße, der Einfluss der Knotendichte und der Gewinn, den der Aufbau von Routenketten gegenüber dem Ansatz des Flutens aufweist, betrachtet.

8.3.1 Platzierung einer CEP-Instanz

Durch verschiedene Platzierungen von CEP-Instanzen ergeben sich unterschiedliche Scores zum Beginn einer Optimierung. In diesem Teil soll evaluiert werden, bei wie viel Abstand zum optimalen Score wie viele Migrationsschritte erforderlich sind und welchen Nutzen das mobile Ad-hoc-Netzwerk daraus zieht.

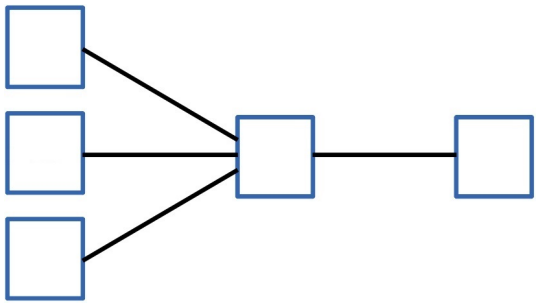


Abbildung 8.2: Der Operatorgraph für die Evaluierung der Ersparnis durch die Optimierung

Simulationsszenario:

- Zur Simulation wurde ein Operatorgraph aufgesetzt, der fünf CEP-Instanzen beinhaltet. Eine der fünf CEP-Instanzen ist dabei mit allen anderen CEP-Instanzen verbunden, wie es in Abbildung 8.2 dargestellt ist.
- Für die Simulation wurde eine quadratische Fläche von 2,25 km^2 erstellt, die jeweils 1500 m Seitenlänge hat. Die Knotendichte auf dieser Fläche beträgt 40 $\frac{Knoten}{km^2}$, insgesamt sind 90 mobile Knoten beteiligt.
- Die Platzierung der CEP-Instanz mit den Links zu allen anderen CEP-Instanzen erfolgt in einem Muster mit äquidistanten Abständen. Die günstigste Endplatzierung ist in jedem Simulationslauf ein mobiler Knoten mit dem Score von 18.

Platzierung	Startscore	Migrationen	Lohnend ab Anzahl Korrelationen
1	20	1	20
2	20	1	19
3	24	1	7
4	25	1	6
5	25	1	5
6	26	2	10
7	27	2	9
8	29	2	7
9	32	3	8
10	33	3	8
11	34	3	8
12	35	3	8
13	37	5	11
14	39	4	8
15	43	4	6
16	47	4	6

Tabelle 8.1: Simulation mit unterschiedlichen Startscores

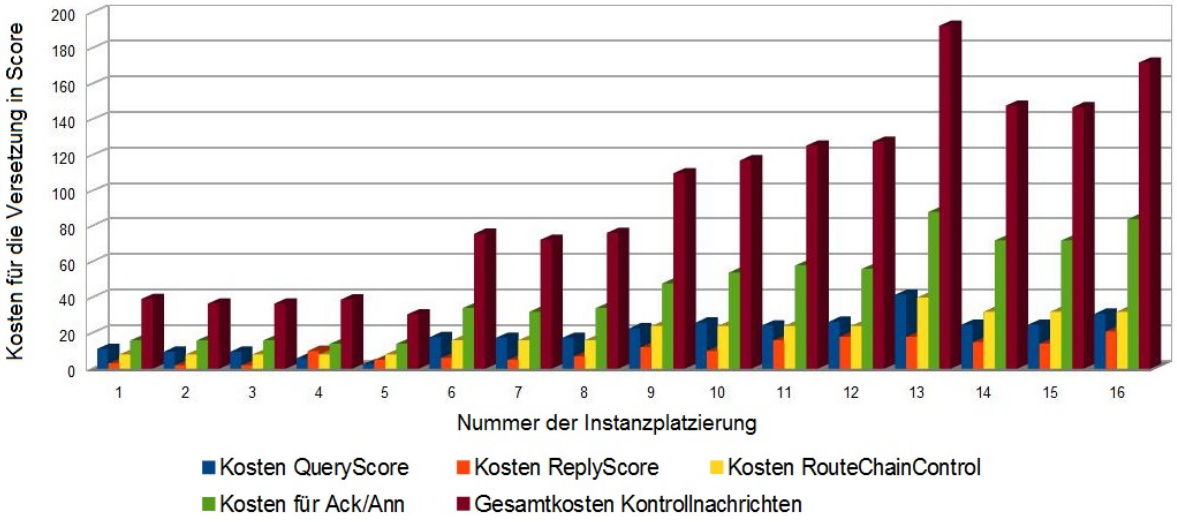


Abbildung 8.3: Die Kostenzusammensetzung für die Migrationen

Tabelle 8.1 zeigt die Ergebnisse der Simulation. Auffallend ist, dass bei einer Instanzplatzierung sehr nahe am Optimum die Amortisierung erst sehr spät einsetzt. Das ist darauf zurückzuführen, dass für sehr wenig Verbesserung lange Nachrichtenwege in Kauf genommen werden, um Announcements und Acknowledgements zu versenden. Deswegen ist der letzte Migrationsschritt regelmäßig der teuerste Schritt innerhalb einer Kette von Migrationen. Die angegebene Anzahl an Korrelationen, ab welcher die Optimierung lohnend ist, ist dabei so angegeben, dass während der Migrationskette keine Korrelationen stattfinden. Sollten während der Migration bereits Korrelationen stattfinden, lohnt sich die Migrationskette bereits nach weniger Korrelationen, da Korrelationen dann bereits günstiger ausgeführt werden, als zur Startplatzierung.

Die Kostenzusammensetzung für die Migrationen bis zum Optimum wird in Abbildung 8.3 gezeigt.

8.3.2 Variation der Netzwerkgröße

Die zweite Evaluation soll zeigen, wie sich die Größe des mobilen Ad-hoc-Netzwerks auf die Kosten der Optimierung auswirkt.

Simulationsszenario:

- Zur Simulation wurde ein Operatorgraph aufgesetzt, der fünf CEP-Instanzen beinhaltet. Eine der fünf CEP-Instanzen ist dabei mit allen anderen CEP-Instanzen verbunden, wie es in Abbildung 8.2 dargestellt ist.
- Für die Simulation wurde eine quadratische Fläche erstellt, die in jeder Simulation in ihrer Größe variiert wurde. Die Knotendichte auf dieser Fläche beträgt 40 $\frac{Knoten}{km^2}$.

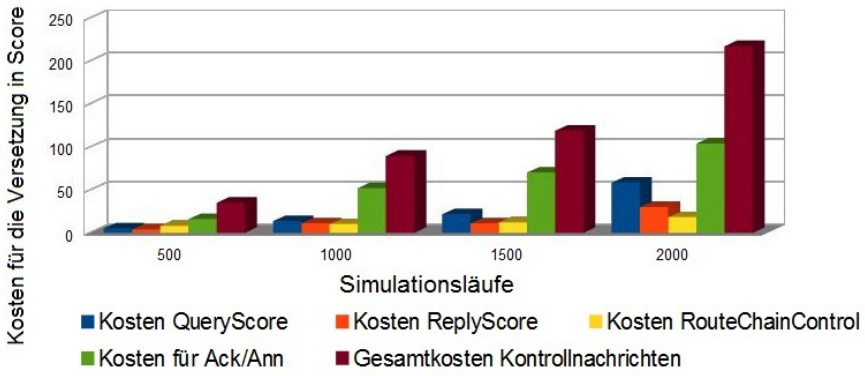


Abbildung 8.4: Kostensteigerung bei vergrößertem Simulationsfeld

- Die Platzierung der CEP-Instanzen erfolgt in jeder Simulation mit dem gleichen Muster, wobei die Abstände proportional mit der Größe des Feldes zunehmen. Die CEP-Instanz, die mit allen anderen CEP-Instanzen verbunden ist, wird an den Konsumenten angeheftet, wie es in dieser Arbeit bei der Aufsetzung eines neuen Operatorgraphen beschrieben wurde.

Seitenlänge	Feldgröße	Anzahl Knoten	Migrationen	Lohnend ab Anzahl Korrelationen
500 m	0,25 km^2	10	2	5
1000 m	1 km^2	40	3	7
1500 m	2,25 km^2	90	4	7
2000 m	4 km^2	160	7	8

Tabelle 8.2: Simulation mit unterschiedlichen Startscores

Die Ergebnisse zeigen, dass bei steigender Feldgröße auch die Anzahl an Migrationen steigt. Die Sprungdistanz, die eine CEP-Instanz mit einer Migration überbrücken kann, ist durch die Reichweite der drahtlosen Funkverbindung begrenzt. Um die durch das gewachsene Feld größere Entfernung zum Optimum überbrücken zu können, sind mehr Migrationsschritte notwendig. Dies schlägt sich in erhöhten Kosten nieder, da zu jeder Migration eine Routenkette aufgebaut wird und Announce- und Acknowledgement-Nachrichten versendet werden müssen. Die Werte, ab welcher Anzahl an Korrelationen eine Migrationskette lohnend ist, ist wie in der vorausgegangenen Evaluierung angegeben, ohne dass Korrelationen stattfinden, während die Optimierung ausgeführt wird.

In Abbildung 8.4 sind die Kosten dargestellt.

8.3.3 Einfluss der Knotendichte

Die dritte Evaluation soll klären, welchen Einfluss die Knotendichte auf die verwendeten Algorithmen hat.

Simulationsszenario:

- Zur Simulation wurde ein Operatorgraph aufgesetzt, der fünf CEP-Instanzen beinhaltet. Eine der fünf CEP-Instanzen ist dabei mit allen anderen CEP-Instanzen verbunden, wie es in Abbildung 8.2 dargestellt ist.
- Für die Simulation wurde eine quadratische Fläche von 2,25 km^2 erstellt, die jeweils 1500 m Seitenlänge hat. Die Knotendichte auf dieser Fläche wird in jedem Simulationslauf verändert.
- Die Platzierung der CEP-Instanzen erfolgt in jeder Simulation mit dem gleichen Muster. Die CEP-Instanz, die mit allen anderen CEP-Instanzen verbunden ist, wird an den Konsumenten angeheftet, wie es in dieser Arbeit bei der Aufsetzung eines neuen Operatorgraphen beschrieben wurde.

Anzahl Knoten	Angefragte Knoten	Antworten	Migrationen	Lohnend ab Anzahl Korrelationen
60				
70	42	12	4	6
80	71	23	5	9
90	52	11	4	6
100	92	21	5	9
110	113	17	5	9
120	136	21	6	10
130	131	18	5	9
140	143	19	5	9

Tabelle 8.3: Simulation mit steigender Knotendichte

In Tabelle 8.3 ist erkennbar, dass durch die in der Scoringtabelle eingebauten Zufallskomponente die Migrationsketten unterschiedliche Wege genommen haben, um das Optimum zu erreichen. Dadurch erhalten manche Wege eine höhere Anzahl an Migrationen. Durch die höhere Knotendichte stehen mehr mobile Knoten zur Verfügung, die einen besseren Score erzielen, als der anfragende mobile Knoten. Daraus resultiert neben mehr Antwortnachrichten auch eine größere Auswahl an möglichen Migrationswegen. Sinnvollerweise müssen in dieser Simulation die Tabellenzeilen betrachtet werden, die die gleiche Anzahl an Migrationen aufweisen. Dort ist sichtbar, dass Anzahl der angefragten mobilen Knoten steigt, je höher die Knotendichte wird. Auch steigt die Anzahl der mobilen Knoten, die auf eine Anfrage antworten. Die Routenketten sind von der Knotendichte nicht betroffen, da die Kosten für den Auf- und Abbau dort von der Anzahl der Migrationen abhängt. Die Simulation hat gezeigt, dass mit steigender Knotendichte die Kosten des Scoring steigen. Dies schlägt sich auch auf die Gesamtkosten nieder. Durch die Rundung auf ganze Korrelationen ist dies in der Tabelle nicht sichtbar. In Abbildung 8.5 sind die steigenden Kosten für das Scoring erkennbar. Gleichzeitig entstehen mehr Migrationswege.

Die Simulation mit 60 Knoten konnte nicht durchgeführt werden, da das mobile Ad-hoc-Netzwerk wegen der geringen Knotendichte eine Partitionierung aufwies, von welcher der Operatorgraph betroffen war. Der Operatorgraph war deswegen nicht funktionstüchtig.

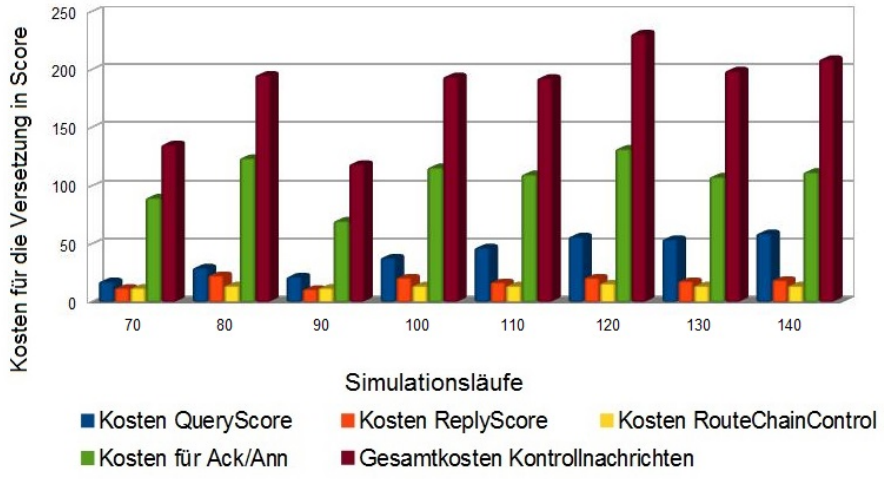


Abbildung 8.5: Kostensteigerung bei größerer Knotendichte

8.3.4 Vergleich von Fluten gegenüber dem Aufbau von Routenketten

Die vierte Evaluation zeigt auf, in welchem Maß eine Einsparung zwischen der Migrationspublikation über die in dieser Arbeit eingeführten Routenketten gegenüber dem naiven Verfahren des Flutens getroffen werden kann. Bei diesem Verfahren werden nicht, wie in der Arbeit vorgestellt, Routenketten aufgebaut, sondern die Benachrichtigung an alle mobilen Knoten des gesamten Netzwerks gesendet.

Simulationsszenario:

- Zur Simulation wurde ein Operatorgraph aufgesetzt, der in jedem Simulationslauf eine andere Anzahl an CEP-Instanzen beinhaltet. Eine der CEP-Instanzen ist dabei mit allen anderen CEP-Instanzen verbunden, wie es in Abbildung 8.6 dargestellt ist.
- Für die Simulation wurde eine quadratische Fläche von 2,25 km^2 erstellt, die jeweils 1500 m Seitenlänge hat. Die Knotendichte auf dieser Fläche beträgt 40 $\frac{Knoten}{km^2}$, insgesamt sind 90 mobile Knoten beteiligt.
- Die Platzierung der hinzugefügten CEP-Instanzen erfolgt in einem Muster mit äquidistanten Abständen. Die CEP-Instanz, die mit allen anderen CEP-Instanzen verbunden ist, wird an den Konsumenten angeheftet, wie es in dieser Arbeit bei der Aufsetzung eines neuen Operatorgraphen beschrieben wurde.

In den Simulationen wurde ermittelt, dass ein mobiler Knoten im Durchschnitt 16,4 benachbarte mobile Knoten besitzt, zu welchen er unmittelbar drahtlos kommunizieren kann. Dies entspricht einer Gesamtnachbarzahl von 1476. Wird eine Nachricht durch das mobile Ad-hoc-Netzwerk geflutet, muss diese im gewählten Szenario insgesamt 90 mal ausgesendet und 1476 mal empfangen werden.

Flutet man die Nachricht durch das mobile Ad-hoc-Netzwerk, entsteht hierbei ein Score, der in Tabelle 8.4 aufgeführt ist.

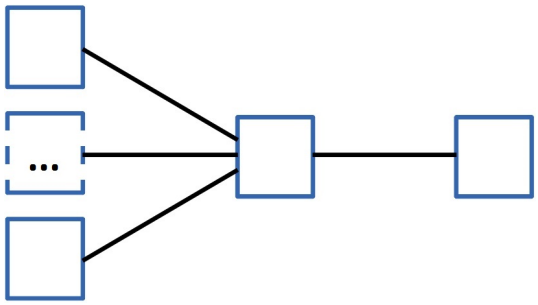


Abbildung 8.6: Der Operatorgraph für die Evaluierung der Routenketten gegenüber dem Fluten

Nachbarn	Migrationen	RouteChainControl	Hops für Ack/Ann	Gesamtkosten
4	4	12	70	82
5	4	14	94	108
6	4	16	128	144
7	4	18	142	160
8	4	20	164	184
9	4	22	184	206
10	4	24	198	222

Tabelle 8.5: Kosten für Routenketten im Simulationsszenario

Aussendungen	Kosten Aussendungen	Empfangszahl	Kosten für Empfang	Gesamtkosten
90	53,1	1476	605,16	658,26

Tabelle 8.4: Kosten für das Fluten im Simulationsszenario

Dieser Score wird für eine Migrationsankündigung verbraucht, die alle benachbarten CEP-Instanzen abdeckt. Dies ist unabhängig davon, von welchem mobilen Knoten zu welchem mobilen Knoten die CEP-Instanz migriert.

Für die Erstellung der Routenketten mussten bei vier Migrationen Weiterleitungen für jede benachbarte CEP-Instanz eingerichtet werden. Damit verbunden ist das Aussenden eines Announcement sowie der Empfang eines Acknowledgement. Die Scores für diesen Vorgang sind in Tabelle 8.4 gezeigt.

In Diagramm 8.7 ist aufgetragen, welche Energiekosten sich im Vergleich zum Fluten ergeben.

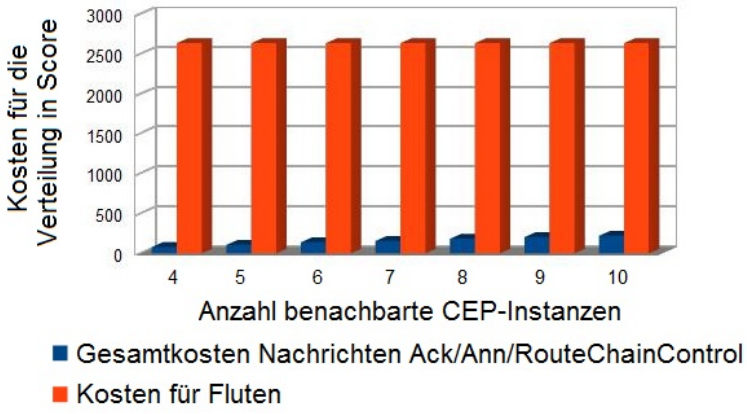


Abbildung 8.7: Kosten für Routenketten gegenüber dem Fluten

Dies entspricht einer Ersparnis in der Simulation von 90 bis zu 97%. Die hohen Energiekosten für das Fluten entstehen durch den vielfachen Empfang, der immer ausgeführt werden muss, wenn ein benachbarter mobiler Knoten einen Broadcast aussendet. Bei Netzwerken mit einer höheren Knotendichte wächst neben den Aussendungen die Anzahl der Nachbarn noch stärker an, sodass die Kosten für ein Fluten in solchen Netzwerken stark ansteigen. Pro Migration muss dieser Vorgang wiederholt werden. Die Kosten, die in Tabelle 8.4 aufgeführt sind, müssen für das Simulationsszenario deswegen mit dem Faktor vier multipliziert werden.

9 Verwandte Arbeiten

Es existieren bereits verwandte Arbeiten auf dem Gebiet der optimierten Platzierung auf dem Gebiet von CEP-Systemen. Diese Arbeiten werden in diesem Kapitel vorgestellt.

9.1 Network-Aware Operator Placement for Stream-Processing Systems

In [PLS+06] werden die Komponenten eines CEP-Systems in feste und lose Komponenten geteilt. Während feste Komponenten immer auf dem gleichen Knoten ausgeführt werden, können lose Komponenten die Knoten im Zuge einer Optimierung wechseln. Oft sind Sensoren feste Komponenten, während Operatoren optimal im Netzwerk verteilt werden können.

Ursprüngliche CEP-Systeme überlassen die optimale Platzierung der dann nur theoretisch losen Komponenten dem Administrator, welcher diese einem festen Knoten zuweist. Eine Optimierung erfordert eine neue Konfiguration durch den Administrator. Um dieses Problem zu umgehen, führt [PLS+06] eine SBON-Architektur ein (SBON steht dabei für stream-based overlay network). SBON kann auf Basis verschiedener Metriken einen Koordinatenraum aufstellen, in welchem die Euklid-Distanz der Knoten der Bewertung in der Metrik entspricht. Als Metrik ist die Latenz verwendbar.

Der erste Schritt ist das Einfügen der Konfiguration in diesen Koordinatenraum. Die zugehörigen Links werden als Federn modelliert, wobei die physikalischen Eigenschaften der Feder als Latenz und Datenrate gesehen werden. Das System ordnet sich dann an den nicht verschiebbaren Komponenten aus und repräsentiert eine optimierte Platzierung.

Im zweiten Schritt wird diese Platzierung auf einen physikalischen Knoten abgebildet. Dazu sucht SBON die k am wenigsten entfernten Knoten und sortiert diese nach dem Abstand zum Punkt im virtuellen Koordinatenraum. In der sortierten Liste wird nun in aufsteigender Reihenfolge nach einem Knoten gesucht, der diesen Operator bereits ausführt. Dieser kann dann für die Konfiguration mit genutzt werden. Findet sich kein solcher Knoten, wird der erste Knoten in der Liste als physikalischer Knoten gewählt, der die gestellten Anforderungen erfüllen kann.

Die Reevaluierung der getroffenen Optimierung wird periodisch lokal auf den Knoten durchgeführt. Eine Oszillation tritt hierbei nicht auf, da die Knoten über identisches Wissen verfügen.

9.2 Solving the Multi-operator Placement Problem in Large-Scale Operator Networks

[RDR10] erklären einen Ansatz zur Annäherung zur im Hinblick auf das gesamte Bandbreite-Verzögerungs-Produkt am meisten optimalen Lösung. Hierzu wird, wie auch in [OKRR13] ein Operatorgraph definiert. Als Annahme wird getroffen, dass Konsumenten und Produzenten nicht beweglich sind, also auf einem einmal zugewiesenen Knoten verbleiben.

Ähnlich zu [PLS+06] wird zur Lösung des Problems ein Latenzraum definiert, der als n-dimensionales kartesianisches Koordinatensystem ausgeführt wird. Der Abstand der Knoten in diesem Raum entspricht deren Latenz in der Realität zwischen zwei jeweiligen Knoten.

Um das Multi-Operator-Placement-Problem (MOP) zu lösen, löst jeder Operator, welcher beweglich ist, das Single-Operator-Placement-Problem (SOP). Eine Lösung wird aus der Berechnung der optimalen Platzierung im Latenzraum bestimmt, welche dann auf einen physikalischen Knoten abgebildet wird. Die Ausführung des Algorithmus wird durch Ereignisse gesteuert, die entweder eine Neusetzung eines benachbarten Operators oder eigene Messungen mit festgestellter Veränderung sein können.

9.3 MigCEP: Operator Migration for Mobility Driven Distributed Complex Event Processing

Fog Computing, wie es von Cisco vorgeschlagen wurde, erlaubt eine Ausführung von Operatoren sehr nahe an der Position von Sensoren und Konsumenten. Im Vergleich zum Cloud Computing, bei welchem Daten einen langen Weg durch das Netzwerk in der Infrastruktur nehmen, können so sehr kurze Latenzzeiten erreicht werden. Eine Kombination beider Konzepte erlaubt ein dynamisches System, welches auf Netzlast mit Platzierung beispielsweise an Routern nahe des mobilen Knotens reagiert, während rechenintensive Operatoren in entfernte Clouds ausgelagert werden.

Um die Platzierung der Operatoren nahe zu den mobilen Knoten aufrecht zu erhalten sind Migrationen erforderlich. Entfernen sich die mobilen Knoten von den Infrastrukturknoten, auf welchen die Operatoren platziert sind, steigt die Netzwerklast und die Latenz erhöht sich. In [OKRR13] werden Algorithmen beschrieben, die sinnvolle Migrationen erkennen, ohne dabei die Kosten, die durch diese entstehen, zu vernachlässigen.

Die Konfiguration des CEP-Systems wird als Operatorgraph dargestellt, wie bereits in Kapitel 2 erläutert wurde. Meldet sich ein mobiler Knoten an einem neuen Zugangspunkt an, verbleibt der Operator auf dem alten Zugangsknoten. Dies führt dazu, dass der alte sowie der neue Zugangsknoten in den Datenstrom zum mobilen Knoten involviert werden und die Netzwerklast sowie die Latenz zunehmen. Eine Migration erzeugt abhängig von der Größe des Operators (welche sich aus fester Größe wie dem ausgeführten Algorithmus und dynamischer Größe, zum Beispiel Ereignispuffer, zusammensetzt) ebenfalls eine Netzlast. Der Aufwand ist nur dann sinnvoll, wenn sich die Migration amortisieren kann.

Jedem Operator wird ein Migrationsplan zugewiesen, welcher einen Satz zukünftiger Migrationsziele enthält, wann der Operator auf dem entsprechenden Knoten ausführbar sein muss und zu welchem

Zeitpunkt eine Migration stattfinden muss, um diesen Zeitpunkt einhalten zu können. Dem Plan werden Messungen von Latenzen, Lasten und Bewegungen von mobilen Knoten zugrunde gelegt. Zum Migrationsplan gibt es zusätzlich einen Zeitplan, durch welchen ermittelt werden kann, welche Kosten und welche Zeitdauer für eine Migration veranschlagt werden müssen. In [OKRR13] werden solche Pläne auch für unsichere Migrationen aufgebaut. Unsicherheiten können entstehen, wenn ein mobiler Knoten plötzlich seine Bewegungsmuster ändert, wie es bei einem Stau mit einer Umleitung passieren kann, wenn das System von einem Kraftfahrzeug genutzt wird.

9.4 Abgrenzung zu dieser Arbeit

Die Situation in einem mobilen Ad-hoc-Netzwerk erfordert, dass mit der begrenzten Ressource Energie sinnvoll umgegangen wird. Durch die gemeinsame Belegung des Funkkanals können längere Wege eine kürzere Latenz bedeuten. Dies führt in einem mobilen Ad-hoc-Netzwerk dazu, dass mehr Energie für die Zustellung eines Datenpaketes aufgewendet wird, als es erforderlich ist.

Unter diesem Aspekt soll diese Arbeit aufzeigen, wie eine Optimierung im Hinblick auf die Anzahl zurückgelegter Hops möglich ist, um so Energie einsparen zu können. Dieses Ziel soll durch einzelne Entscheidungen von mobilen Knoten ohne Aufbau eines Vivaldi-Korrdinatensystems zur Repräsentation der Latenzen erreicht werden.

10 Zusammenfassung und Ausblick

In dieser Arbeit wurden Methodiken eingeführt, die die Platzierung von CEP-Instanzen in einem CEP-System, das in einem mobilen Ad-hoc-Netzwerk ausgeführt wird, einem Administrator abnehmen. Die Knoten eines mobilen Ad-hoc-Netzwerks werden häufig von einer ortsunabhängigen Energiequelle gespeist, die nur begrenzt Energie aufnehmen kann. Zur Minimierung der durch den Betrieb des CEP-Systems benötigten Gesamtenergie wurden in dieser Arbeit Methodiken untersucht, CEP-Instanzen automatisiert so zu platzieren, dass möglichst wenig Energie bei der Vermittlung der Ereignisse durch das mobile Ad-hoc-Netzwerk benötigt wird um die Laufzeit der mobilen Knoten nicht mehr zu begrenzen, als es notwendig ist. Die Arbeit hat gezeigt, dass eine Optimierung mit diesem Ziel auch mit in einem mobilen Ad-hoc-Netzwerk bereits vorhandenem, auf jedem mobilen Knoten verfügbarem Wissen durchgeführt werden kann.

Ein auf den mobilen Knoten ausgeführtes System prüft dazu regelmäßig die Platzierung von CEP-Instanzen, die auf mobilen Knoten ausgeführt werden und migriert diese, falls es eine Platzierung gibt, deren Wahl eine Verbesserung des Energiebedarfs bedeutet. Die getroffene Entscheidung basiert auf den gesammelten Daten, die während der komplexen Ereignisverarbeitung im Hintergrund erhoben werden, um die Frequenz und die Größe von Ereignissen abschätzen zu können. Diese Informationen werden zusammen mit der Entfernung der benachbarten CEP-Instanzen zu einem Score zusammengefasst, der die Attraktivität des mobilen Knotens beschreibt, eine CEP-Instanz auszuführen. Zur Entscheidung verwendete Informationen liegen bei der Berechnung dem mobilen Knoten vollständig vor und müssen nicht von anderen Teilnehmern unter der Erfordernis von Energie bezogen werden, sodass der Algorithmus ausschließlich auf lokalem Wissen arbeitet. Der von einem mobilen Knoten ermittelte Score wird mit seinen benachbarten Knoten verglichen und bei einer möglichen Verbesserung wird eine Migration ausgelöst.

Eine Migration birgt die Gefahr von Inkonsistenzen zwischen den benachbarten CEP-Instanzen. Funktioniert ein Link innerhalb eines Operatorgraphen nicht mehr, bedeutet das oft den Ausfall der folgenden Zweige und das Ausbleiben von fertig bearbeiteten Ereignissen bei Konsumenten des Operatorgraphen. Um dem vorzubeugen, wurde zusätzlich die Methodik der Routenketten entwickelt. Dort werden bei einer Migration Zeiger auf den neuen mobilen Knoten eingerichtet, der die CEP-Instanz betreibt. Diese bleiben so lange erhalten, wie es erforderlich ist.

In dieser Arbeit wurden Operatorgraphen behandelt, die vollständig innerhalb einer Partition betrieben werden. In Katastrophengebieten findet sich nicht überall eine gleichbleibende Knotendichte vor. Nimmt die Knotendichte zu stark ab, können Partitionen entstehen. In ländlichen Gebieten befinden sich Ortschaften sehr oft in einer Entfernung zueinander, die von WiFi nicht oder nicht sicher überbrückt werden kann. Diese Entfernungen sind für Naturkatastrophen in den meisten Fällen

unbedeutend. Die Informationen, die innerhalb einer Partition vorliegen, sind bei solchen Ereignissen auch für andere Partitionen interessant.

Informationen können übertragen werden, solange eine Infrastruktur zwischen den verschiedenen Partition besteht, die in dieser Arbeit nicht betrachtet wurde. Ein zusätzliches Verfahren, welches auch die Infrastruktur nutzt, kann neben der Überbrückung von großen Entfernungen ebenfalls zur Abschätzung eingesetzt werden, ob die Übertragung zwischen zwei mobilen Knoten über die Infrastruktur oder über den drahtlosen Kanal energetisch günstiger ist.

Auch beim Ausfall von Infrastrukturen gibt es Techniken, die es erlauben, Partitionen über weite Entfernungen miteinander zu vernetzen. In Katastrophengebieten werden regelmäßig von Rettungsorganisationen und Funkamateuren Notfunkstationen eingerichtet. Für solche Stationen gibt es Verfahren die es erlauben, durch schmale Bandbreiten mit kleinstem Energiebedarf große Entfernungen zu überbrücken. Diese vorhandene Heterogenität muss in Zukunft durch geeignete Methoden überwunden werden, um solche Übertragungsmedien und die mit ihnen verbundenen Vorteile auch für die komplexe Ereignisverarbeitung nutzbar zu machen.

Literaturverzeichnis

[BBV09] N. Balasubramanian, A. Balasubramanian, A. Venkataramani. Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '09, S. 280–293. ACM, New York, NY, USA, 2009. (Zitiert auf Seite 18)

[BRG99] M. Bansal, R. Rajput, G. Gupta. Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations. In *Mobile Ad-hoc Network (MANET) Working Group, IETF*. 1999. (Zitiert auf Seite 6)

[CCG00] F. Calì, M. Conti, E. Gregori. Dynamic Tuning of the IEEE 802.11 Protocol to Achieve a Theoretical Throughput Limit. Band 8, S. 785–799. IEEE Press, Piscataway, NJ, USA, 2000. (Zitiert auf Seite 2)

[Cis] Link-State Versus Distance Vector. http://docwiki.cisco.com/wiki/Routing_Basics#Link-State_Versus_Distance_Vector. (Zitiert auf Seite 11)

[CM12] G. Cugola, A. Margara. Processing Flows of Information: From Data Stream to Complex Event Processing. *ACM Computing Surveys (CSUR)*, 44(3):15, 2012. (Zitiert auf Seite 5)

[CSAK98] J.-C. Chen, K. M. Sivalingam, P. Agrawal, S. Kishore. A Comparison of MAC Protocols for Wireless Local Networks Based on Battery Power Consumption. In *INFOCOM'98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, Band 1, S. 150–157. IEEE, 1998. (Zitiert auf Seite 8)

[FN01] L. M. Feeney, M. Nilsson. Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, Band 3, S. 1548–1557. IEEE, 2001. (Zitiert auf den Seiten viii und 8)

[Ine] INET Framework für OMNeT++. http://inet.omnetpp.org/. (Zitiert auf Seite 65)

[JMC+01] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, L. Viennot. Optimized Link State Routing Protocol for Ad Hoc Networks. In *Multi Topic Conference, 2001. IEEE INMIC 2001. Technology for the 21st Century. Proceedings. IEEE International*, S. 62–68. IEEE, 2001. (Zitiert auf Seite 12)

[JPDG04] A. Jayasuriya, S. Perreau, A. Dadej, S. Gordon. Hidden vs. Exposed Terminal Problem in Ad hoc Networks. In *Proceedings of the Australian Telecommunication Networks and Applications Conference*. 2004.

[KKR10] G. G. Koch, B. Koldehofe, K. Rothermel. Cordies: Expressive event correlation in distributed systems. In *Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems*, S. 26–37. ACM, 2010. (Zitiert auf Seite 4)

[KORR12] B. Koldehofe, B. Ottenwälder, K. Rothermel, U. Ramachandran. Moving Range Queries in Distributed Complex Event Processing. In *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems*, DEBS '12, S. 201–212. ACM, New York, NY, USA, 2012.

[LLS08] G. T. Lakshmanan, Y. Li, R. Strom. Placement Strategies for Internet-Scale Data Stream Systems. Band 12, S. 50–60. IEEE Educational Activities Department, Piscataway, NJ, USA, 2008.

[LOBC12] A. Liu, M. Olson, J. Bunn, K. M. Chandy. Towards a Discipline of Geospatial Distributed Event Based Systems. In *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems*, DEBS '12, S. 95–106. ACM, New York, NY, USA, 2012. (Zitiert auf Seite 1)

[Obe07] S. Oberoi. Introduction to Complex Event Processing & Data Streams. *SOA World Magazine*, S. 20–24, 2007.

[OKRR13] B. Ottenwälder, B. Koldehofe, K. Rothermel, U. Ramachandran. MigCEP: Operator Migration for Mobility Driven Distributed Complex Event Processing. In *Proceedings of the 7th ACM International Conference on Distributed Event-based Systems*, DEBS '13, S. 183–194. ACM, New York, NY, USA, 2013. (Zitiert auf den Seiten vii, 4, 15, 76 und 77)

[PLS+06] P. Pietzuch, J. Ledlie, J. Shneidman, M. Roussopoulos, M. Welsh, M. Seltzer. Network-Aware Operator Placement for Stream-Processing Systems. In *Data Engineering, 2006. ICDE06. Proceedings of the 22nd International Conference on*, S. 49–49. IEEE, 2006. (Zitiert auf den Seiten 75 und 76)

[PR99] C. E. Perkins, E. M. Royer. Ad-hoc On-Demand Distance Vector Routing. In *Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA99. Second IEEE Workshop on*, S. 90–100. IEEE, 1999. (Zitiert auf Seite 11)

[RDR10] S. Rizou, F. Durr, K. Rothermel. Solving the Multi-operator Placement Problem in Large-Scale Operator Networks. In *Computer Communications and Networks (ICCCN), 2010 Proceedings of 19th International Conference on*, S. 1–6. IEEE, 2010. (Zitiert auf den Seiten 22 und 76)

[RT99] E. M. Royer, C.-K. Toh. A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks. *Personal Communications, IEEE*, 6(2):46–55, 1999. (Zitiert auf den Seiten 11 und 12)

[RZ07] A. Rahmati, L. Zhong. Context-for-wireless: Context-sensitive Energy-efficient Wireless Data Transfer. In *Proceedings of the 5th International Conference on Mobile Systems, Applications and Services*, MobiSys '07, S. 165–178. ACM, New York, NY, USA, 2007. (Zitiert auf den Seiten 18 und 66)

[Sch03] J. Schiller. *Mobile Communications.* Addison-Wesley, 2003. (Zitiert auf Seite 6)

[See] B. Seeger. Complex Event Processing: Auswertung von Datenströmen. http://www.heise.de/ix/artikel/Kontinuierliche-Kontrolle-905334.html. (Zitiert auf Seite 3)

[SPS] Global smartphone sales Q1 2009-Q3 2013, by operating system. http://www.statista.com/statistics/266219/global-smartphone-sales-since-1st-quarter-2009-by-operating-system/.

[Tan03] A. S. Tanenbaum. *Computernetzwerke*. Pearson Studium, 2003. (Zitiert auf Seite 10)

[TEI] TEIA. Link-State-Routing-Protokolle. http://www.teialehrbuch.de/Kostenlose-Kurse/Internet-Technik/16158-Link-State-Routing-Protokolle.html. (Zitiert auf Seite 12)

[V+01] A. Varga, et al. The OMNeT++ Discrete Event Simulation System. In *Proceedings of the European Simulation Multiconference (ESM'2001)*, Band 9, S. 185. sn, 2001. (Zitiert auf Seite 65)

[Vet12] M. Vetter. *Ereigniskorrelation auf energiebeschränkten mobilen Endgeräten*. Bachelor's thesis, Universität Stuttgart, 2012. (Zitiert auf den Seiten 1 und 5)

Alle URLs wurden zuletzt am 17. 04. 2014 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift