

Studiengang: INFOTECH
Betreuer: Dipl.-Inf.Jing Tian
Prüfer: Prof.Dr.rer.nat.K.Rothermel

Beginn am: 01.05.03
Beendet am: 30.11.03

CR-Klassifikation: H 2.4, C 2.4, C 2.2, C.4

Master Thesis Nr. <2111>

Location-Based Data Aggregation in Mobile Ad Hoc Networks

Chi Chen

Fakultät Elektrotechnik, Informatik,
Informationstechnik
Universität Stuttgart
Institut für Parallele und Verteilte Systeme (IPVS)
Abteilung Verteilte Systeme
Universitätsstr. 38
70569 Stuttgart

Table of Contents

2.1.	System Model –Vehicular Network	2
2.1.1.	Characteristics of a Single Vehicle	2
2.1.2.	Characteristics of Vehicular Network	2
2.2.	Spatial Model – Road Network.....	3
2.2.1.	Flat Graph Model	3
2.2.2.	Hierarchical Graph Model.....	4
2.2.3.	Realtime graph Model	5
2.3.	Data Model	6
2.3.1.	Raw Data	6
2.3.2.	Aggregated Data.....	7
2.4.	Software architecture	7
3.1.	Ad hoc Routing Protocols	8
3.1.1.	Position based Routing	8
3.1.2.	Summary of Ad hoc routing	13
3.2.	Data Aggregation in Ad hoc network.....	13
3.2.1.	Hierarchical Gossiping Aggregation	14
3.2.2.	Greedy Incremental Tree with Directed Diffusion	15
3.2.3.	Tiny Aggregation.....	16
3.2.4.	Digest Diffusion	18
3.2.5.	Summary of data aggregation.....	19
4.1.	Objective of Data Aggregation	20
4.2.	Requirement of Data Aggregation	20
4.3.	Classification of Aggregation Strategies	20
4.3.1.	Centralized Strategy	22
4.3.2.	Fully Distributed Strategy	22
4.3.3.	In-network Strategy	22
4.3.4.	Routing/Aggregation Tree based Strategy.....	22
4.3.5.	Location based proxy Aggregation.....	23
4.3.6.	Analysis of Tree based Strategy	23
4.3.7.	Modeling and Mathematical Study:	24
4.3.8.	Comparison and Summary	27
4.4.	Location Based Aggregation Protocol	27
4.4.1.	Obtain Aggregated data	28
4.4.2.	Overcome High Mobility	30
4.4.3.	Reduce Communication Cost	30
4.4.4.	Scalability.....	34
4.4.5.	Performance Enhancement	35
5.1.	Spatial Model.....	37
5.1.1.	Interface.....	37
5.1.2.	Representation of spatial information.....	38
5.2.	GEOCAST	39
5.2.1.	API	39
5.2.2.	Protocol Operations	40
5.2.3.	Summary of GEOCAST.....	45
5.3.	Query Application.....	45
5.3.1.	Format of Query and Response	46
5.3.2.	Message Flow of a Query Operations	46
5.3.3.	Operations	47
5.3.4.	Time control	47
5.4.	Location Based Aggregation.....	48
5.4.1.	Protocol Overview.....	48
5.4.2.	Protocol Messages Format	48
5.4.3.	Protocol Operations	49
5.4.4.	Framework for Different Strategies.....	51
6.1.	NS-2 Class Overview.....	53

6.2.	Class SpatialModel	56
6.2.1.	Class Diagram	56
6.2.2.	Data Structure.....	56
6.2.3.	Common Interface.....	57
6.3.	Class GEOCAST_Agent.....	58
6.3.1.	Class Diagram	58
6.3.2.	Data Structure.....	58
6.3.3.	Functions	59
6.4.	Class MDMApp	61
6.4.1.	Class Diagram	61
6.4.2.	Protocol Operations	61
6.5.	Class LBAG_Agent	63
6.5.1.	Class diagram	63
6.5.2.	Data Structure.....	64
6.5.3.	MessageHandler	65
6.5.4.	Summary of Implementation.....	69
7.1.	Simulation Environment	70
7.1.1.	Mobile Node Setup.....	70
7.1.2.	Spatial Model Setup	70
7.1.3.	Simulation Scenarios.....	72
7.1.4.	Protocol Parameters.....	72
7.1.5.	Query Definition.....	72
7.1.6.	Three Strategies.....	73
7.1.7.	Behavior Emulation by LBAG.....	74
7.1.8.	Location of the Query Sender.....	75
7.2.	Simulation Results	75
7.2.1.	Completeness of Aggregated Data	75
7.2.2.	Communication Overhead from Forwarding	76
7.2.3.	Communication Overhead from Broadcast	78
7.2.4.	Overall Communication Cost	79
7.2.5.	Summary of the Simulation Result.....	80

List of Figures

Figure 2-1 Road Networks	3
Figure 2-2 Flat Graph Model	3
Figure 2-3 Hierarchical Graph Model	5
Figure 2-4 Change level of view	5
Figure 2-5 Realtime Graph Model	6
Figure 2-6 Software architecture	7
Figure 3-1 Voids in Network	9
Figure 3-2 GPSR Routing in Perimeter Mode	9
Figure 3-3 Construction of routing tree	17
Figure 3-4 Network Monitoring architecture	18
Figure 4-1 Classification of Aggregation strategies	21
Figure 4-2 Tree based strategy	22
Figure 4-3 Illustration of Aggregation Strategies	23
Figure 4-4 Message Sequence Chart of In-network aggregation	24
Figure 4-5 Simplified System Model	25
Figure 4-6 Centralized strategy	25
Figure 4-7 Fully Distributed aggregation	26
Figure 4-8 Location Based aggregation	26
Figure 4-9 Message Flow of LBAG	28
Figure 4-10 Hierarchical Aggregation	29
Figure 4 - 11 Location Based Aggregation	31
Figure 4-12 Models Mapping	32
Figure 4-13 AggTimer Sequence Chart	34
Figure 4-14 Possible Bottleneck in the network	35
Figure 5-1 Software architecture	37
Figure 5-2 Common interface of Spatial Model	38
Figure 5-3 Spatial Model	39
Figure 5-4 GEOCAST Algorithm	41
Figure 5-5 Broadcast	42
Figure 5-6 Forwarding a Packet	43
Figure 5-7 Message Lost due to outdated Neighbor Information	45
Figure 5-8 Message Sequence Chart of Query Application	47
Figure 5-9 LBAG Message Flow	48
Figure 6-11 Overview of UML Class Diagram1	53
Figure 6-2 ns-2 Node Architecture	54
Figure 6-3 ns-2 Packet Format	55
Figure 6-4 NS-2 Class Hierarchy	55
Figure 6-5 Spatial Model	56
Figure 6-6 ULM diagram of GEOCAST_Agent	58
Figure 6-7 Function recv()	59
Figure 6-8 Function forward()	60
Figure 6-9 Class Diagram of Query Application	61
Figure 6-10 State Chart Diagram of MDMAApp	62
Figure 6-11 LBAG UML Class Diagram	63
Figure 6 12 LBAG recv function	66
Figure 6-13 SDL of Sender Handler	67
Figure 6-14 Query Responder SDL Diagram	68
Figure 6-15- Aggregator SDL Diagram	69
Figure 7-1 Mobile Node Setup	70
Figure 7-2 Simulation Road Graph	71
Figure 7-3 Spatial Information File	71
Figure 7-4 Simulation with Centralized Strategy	73
Figure 7-5 Simulation with Fully Distributed Strategy	73
Figure 7-6 Simulation for Location based Proxy Strategy	74
Figure 7-7 Location of the Query Sender	75

Figure 7-8 Aggregation Completeness	76
Figure 7-9 Communication Overhead from Forwarding	77
Figure 7-10 Communication cost from Broadcast	79
Figure 7-11 Overall Communication Cost	80

List of Tables

Table 2-1 Properties of an edge	4
Table 2-2 Contents of a Raw Data	6
Table 2-3 Contents of a Aggregated Data	7
Table 4-1 Communication Cost.....	27
Table 4-2 Communication Cost of LBAG	31
Table 5-1 GEOCAST Header Fields	40
Table 5-2 Format of Query.....	46
Table 5-3 Format of Response	46
Table 5-4 LBAG Message Header	48
Table 5-5 QUERY MESSAGE.....	48
Table 5-6 Format of Raw Data.....	49
Table 5-7 Format of Agg Data.....	49
Table 5-8 Format of SUPPRESION.....	49
Table 6-1 Data Structure of Spatial Model.....	57
Table 6-2 Common Interface.....	57
Table 6-3 GEOCAST Header.....	58
Table 6-4 Tcl Functions to define a query	62
Table 6-5 LBAG Header	64
Table 6-6 LBAG_QueryMessage.....	64
Table 6-7 LBAG_RawData	64
Table 6-8 LBAG_AggData	65
Table 6-9 Query Map Entry	65
Table 7-1 Protocol Parameters	72
Table 7-2 Define Query Message.....	72

1. Introduction

With recent technology advances, various sensors and onboard computing systems have enabled a vehicle to become a powerful information gathering and processing platform. Sensors can continuously detect the local environment of a vehicle and generate data and events. Moreover, GPS receiver can provide a vehicle global position information and global time. With the help of a digital map, a vehicle can obtain additional information about the road network topology. All the obtained data can be processed by the onboard computing system and stored locally or distributed to other nodes through wireless communication interfaces. In case that no infrastructure of communication is available, multi-hop routing can enable the delivery of information to a remote vehicle through intermediate vehicles. Thus vehicles on road networks can be connected to a network. The name of “vehicular networks” has been used to refer to such a type of mobile ad hoc networks¹. The vehicles that form the networks are also called mobile nodes.

Most data that are generated by individual nodes have close relationship with locations. For instance, one observed traffic congestion is linked to its happening cite; a vehicle’s current speed is linked to the vehicle’s current position. Therefore, when those data are exchanged between nodes, they must be bound with their location context. The representation of the context can be geographic coordinates or semantic names of the location.

It is highly desirable for a vehicle to obtain information from other nodes, so that it can become aware of the environment beyond its observing range. A mobile node can obtain local data of other individual nodes directly by ad hoc routing. But some other values, like the properties of a network, can only be obtained through aggregation operations. The aggregated data can provide some high level information. For example, from the aggregation of speed of nodes on a road, average speed can be computed. If the average speed becomes very low, congestion or red traffic light ahead can be inferred.

Several data aggregation strategies have been proposed for MANETs, especially for sensor networks [SENSORNETS]. However, due to the high mobility and expected large scale, those strategies are not applicable to the vehicular networks. One objective of this thesis is to develop a new data aggregation protocol that can solve the challenges of vehicular networks. Another objective of the thesis is to include semantics of geographic information in data aggregation.

The rest part of the thesis is organized as following:

Chapter 2 gives an overview of the vehicular network, its related spatial models, its data models and the software architecture of a mobile node. Chapter 3 gives an overview of the related work concerning on ad hoc routing and data aggregation in MANETs. Chapter 4 presents the conceptual study. Data aggregation strategies are classified and illustrated. Our new location based proxy strategy is proposed. The performances of difference strategies are computed with mathematical models and compared. Finally the protocol based on location based proxy aggregation is outlined. 5 emphasizes on the protocol design of LBAG and the interfaces, with which it interacts with neighboring modules. The behaviors of LBAG protocol are described. Finally, based on its behavior, the idea of a framework for data aggregation is proposed and explained. Chapter 6 introduces the software implementation of LBAG and other related modules. Chapter 7 presents the simulation environment and the settings of the scenarios. The results are analyzed according to several metrics. Chapter 8 concludes this thesis work.

¹ Mobile ad hoc networks MANETs are autonomous system of mobile routers (and associated hosts) connected by wireless links—the union of which form an arbitrary graph. The routers are free to move randomly and organize themselves arbitrarily; thus, the network’s wireless topology may change rapidly and unpredictably. Such a network may operate in as standalone fashion, or may be connected to the large Internet. [MANET]

2. System Overview

This chapter gives the background information and the presumptions of this thesis. System model introduces the characteristics of vehicular networks. Spatial model shows what kind of spatial information can be used. Data model defines the structure of information that is exchanged between nodes in the network. Finally, the layering structure of the complete software architecture of a mobile node is shown and the relationship between software modules is illustrated.

2.1. System Model –Vehicular Network

2.1.1. Characteristics of a Single Vehicle

Vehicles equipped with sensors, computing systems and communication systems become intelligent. They will have the following characteristics:

Environment Sensing

Intelligent vehicles can sense local environment with various sensors. For example, the mechanical sensors can measure the pressure of tires and slippery of road surface. The acoustic sensors can compute inter-vehicle distances. Thermal sensors can get the atmosphere temperature and photo detectors can measure the intensity and direction of sunlight.

Powerful Computing Capability

Powerful onboard processors can handle the realtime electronic signals from sensors and process the messages exchanged between mobile nodes. Large amount of storage space is now easily to get with low cost. In addition, energy generated by motor gives power consumption no constraint.

Location Information Awareness

GPS² system can provide a vehicle its location precisely. More and more vehicles are equipped with GPS receivers, which can provide them realtime geographic position and global time. With digital maps, road network topology is available too.

Wireless Communications

Development in communication technology increases the diversity of wireless interfaces. The available choices are RF, GPRS³, infrared, Bluetooth⁴, WLAN 802.11, and the coming 3G mobile. A vehicle can have several wireless communication interfaces working in parallel. A vehicle can communicate with other vehicles or access points through wireless interfaces.

With above characteristics a vehicle can act as a full-fledged computer in a mobile Ad hoc network.

2.1.2. Characteristics of Vehicular Network

When vehicles are connected by wireless links autonomously, they form a vehicular ad hoc network. Vehicular network belongs to mobile ad hoc network, but it has its unique properties.

Constrained Node Spatial Distribution

Vehicles normally run on roads. They cannot move into rivers and seldom into open fields. So the distribution of mobile nodes inside a given area is non-uniform, but constrained by road networks.

Dynamic Topology

Vehicles can move at high speed. The range of the speed is from 0 to 250kmh. Therefore nodes' positions change very fast, which causes the frequent changes of a vehicular network's topology.

² Global Position System

³ General Radio Packet Service

⁴ A short range wireless connectivity standard

Frequent Partition

The distribution of nodes on roads is not uniform but time-variant. It is widely observed that vehicles are often clustered on roads. There are frequent network partitions in vehicular networks, especially when the density of nodes on road networks is low.

2.2. Spatial Model – Road Network

As mentioned above, road network is highly related to vehicular networks. It constrains the movement pattern of nodes and determines the distribution of vehicles. Therefore, spatial information of road networks can help mobile nodes understand their environment. In order to let software access the spatial information, each node includes a spatial model to accommodate all known spatial information. The representation of spatial models varies according to the complexity of spatial information. Here we introduce three types of spatial models: flat graph model, hierarchical graph model and realtime graph model.

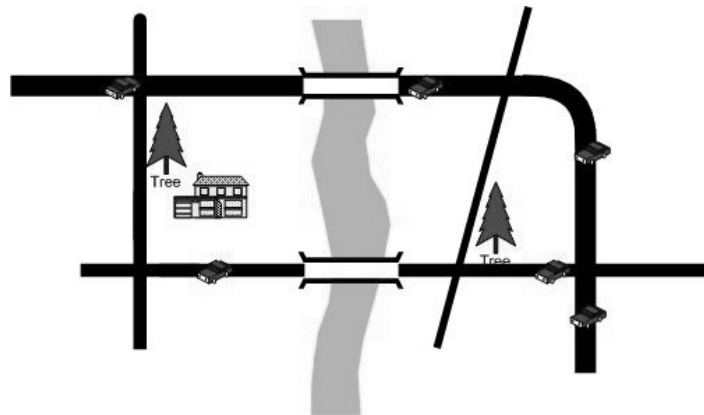


Figure 2-1 Road Networks

2.2.1. Flat Graph Model

The flat graph model only stores the topology of the road network and the related geographic coordinates. A 2D graph as shown in Figure 2-1 can be used to represent a road network.

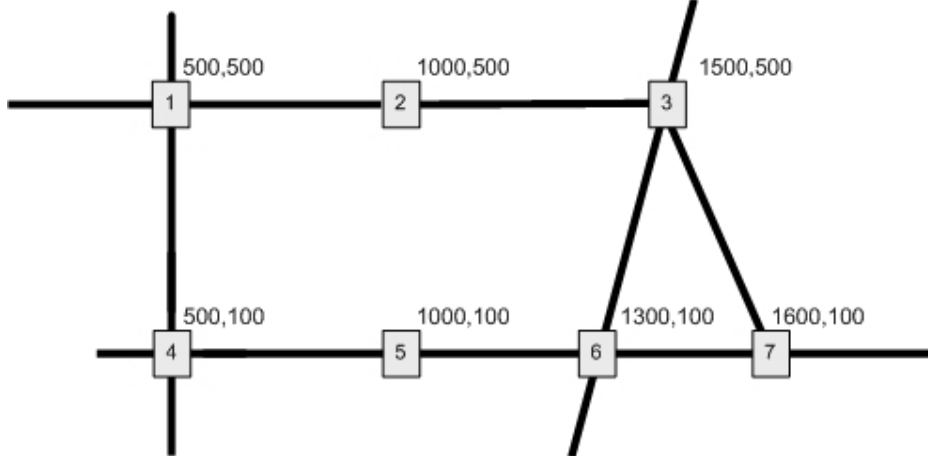


Figure 2-2 Flat Graph Model

The vertices of the graph denote the road crossings and the edges denote the road segments. The property of an edge is the length of the road segment. Flat Graph Model can provide the following operations on

spatial information:

Locating a Point: Given the coordinate of a point, Flat Graph Model returns the edge, where the point is located

Computing shortest Path: Given the start and end coordinates, Flat Graph Model returns the shortest path and its total length.

2.2.2. Hierarchical Graph Model

We noticed that pure geographic information is not enough to understand the environment. The flat graph model cannot tell the relationship between two edges. Semantics of geographic information are needed to determine the relationship between edges and vertices. Semantic information includes name of roads, level of importance, road quality, transportation capacity and so on.

Some semantics of geographic information indicates certain kind of hierarchical structure. For example, the road's level of importance may have a hierarchy like the following:

- Level 1 Road name
- Level 2 Section name
- Level 3 Segment name

So when we integrate semantics of geographic information into flat graph model, we increase the complexity of spatial information. Here we introduce a Hierarchical Graph Model to represent the more complex spatial model.

The road network in hierarchal model is also represented by a graph, but the properties of vertices and edges are extended. The properties of a vertex include the coordinates as well as the name of it. The properties of an edge include not only the length of the segment but also the name for the segment and the name of the section or road it belongs to. The possible properties of an edge are listed in the following table:

Edge Weights:	Road name	Semantic information
	Section name	
	Segment name	
	Road status	
	Level of importance	
	Length	Geographic information

Table 2-1 Properties of an edge

Hierarchical Naming of Edges

If the semantic meaning of a geographic object has hierarchy in nature, then there are several ways to name the geographic object.

One way is to give the geographic object a name in each hierarchy level. Another way is to link the context name of different levels one by one. For example, if a road consists of several sections and each section also consists of several segments, then according to the first solution, we can give each edge a name in each hierarchy level: a segment level name, a section level name and a road level name. According to the second solution, we can give an edge a name as: RoadX-SectionY-SegmentZ.

No matter which naming paradigm is taken, the spatial model must provide functions to map semantic and geographic information reciprocally.

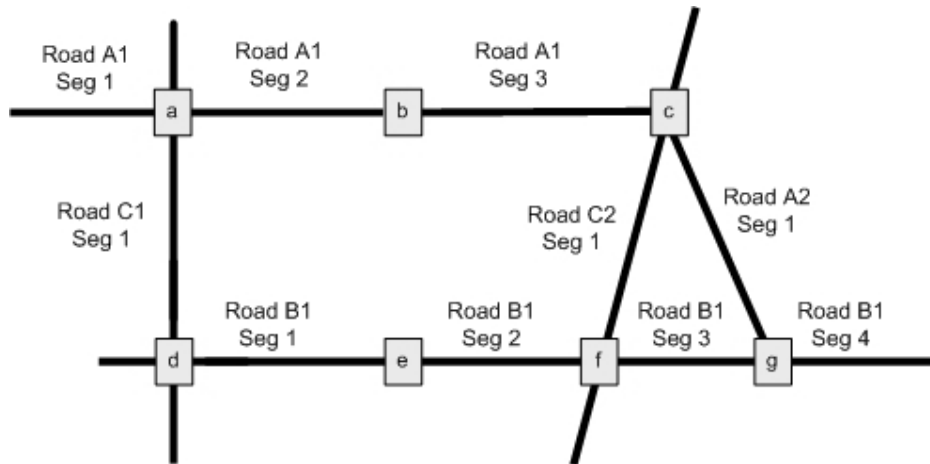


Figure 2-3 Hierarchical Graph Model

Change Level of View

If the semantics of spatial information includes level of importance, then the spatial model can provide different views of a road network, i.e.. For example, the spatial model can hide all the road segments with level lower than 2. Then a new view of the road only includes road segments of level 1 and 2, as shown in Figure 2-4. Such operations help to get the backbone of the road network.

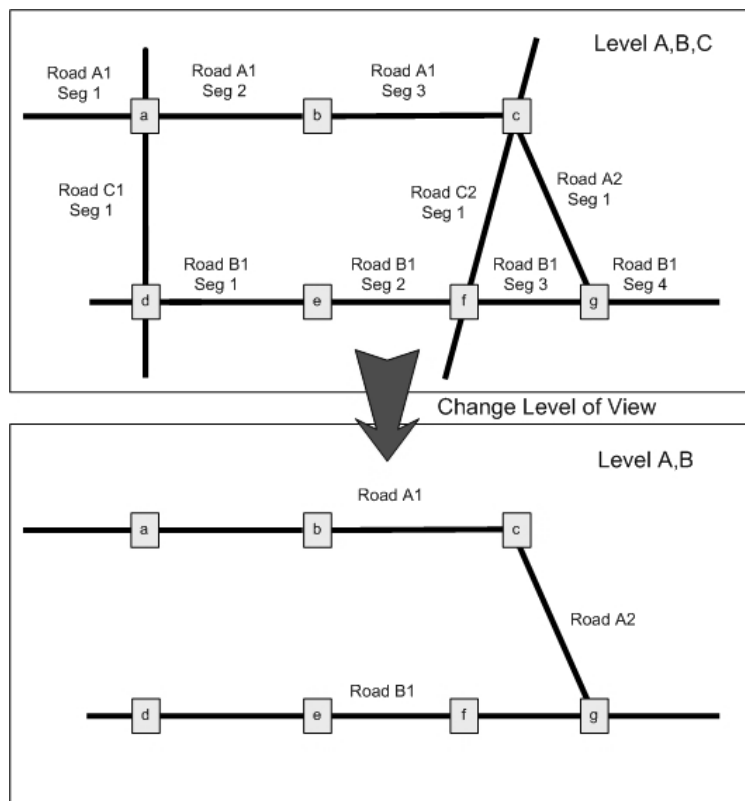


Figure 2-4 Change level of view

2.2.3. Realtime graph Model

If we further extend the spatial information complexity and integrate dynamic and aggregated information into it, we get a spatial model, which can reflect the realtime characteristics of a road network. We name such a spatial model Realtime Graph Model. Here are some examples of dynamic and aggregated

information:

- Average speed of nodes on a road
- Density of vehicles on a road
- Quality of a road, road condition
- Predicted passing time of a road

We can still use a 2D graph to represent a realtime graph model. Edge or vertex properties now include realtime or aggregated data. But all such kind of information should have a related timer, which can trigger the refreshment or deletion after the timer expiration. The spatial model should support following operations:

Update Data: The operation is called to set a new value or refresh an existing value.

Local Aggregation of Data: Spatial model can combine data of several edges together according to the matching of semantic meaning.

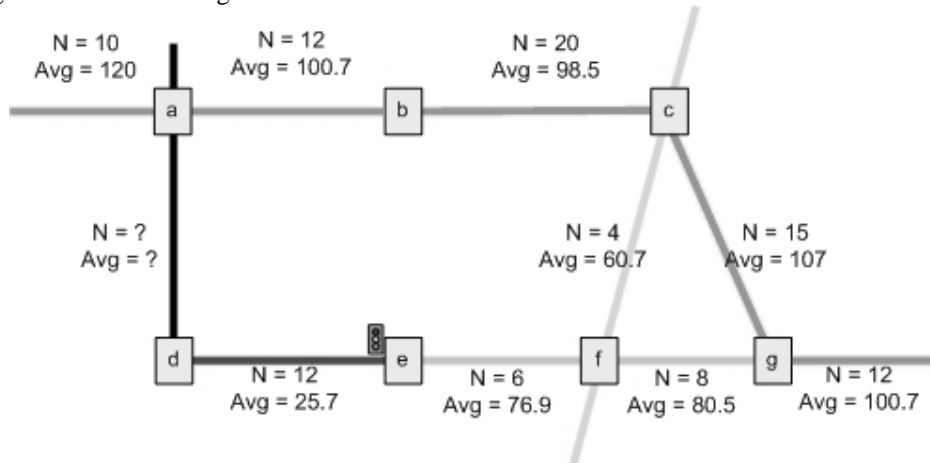


Figure 2-5 Realtime Graph Model

2.3. Data Model

As mentioned before, the data exchanged between nodes should be bound with its context. The context includes node context and geographic context. We classify the data exchanged between nodes into two categories: Raw Data and Aggregated Data

2.3.1. Raw Data

Raw Data are the readings from sensors. They are generated by single node. Most raw data are only meaningful to the node itself, such as the remaining amount of fuel. But some raw data is also useful to the other nodes, e.g. current location or speed of a node. Those kinds of raw data will be distributed in the network. When raw data are distributed, they include their contexts. The context includes when, who or where it is generated and where they are related. Therefore, the raw data should have following structures:

Name of the data
Type of the data
Geographic context
Originator ID
Generation time
Valid time
Data value

Table 2-2 Contents of a Raw Data

Time, location and node context of the raw data are

2.3.2. Aggregated Data

Raw data is only related to one single node. When many raw data are aggregated, the aggregation result is called aggregated data. If the aggregated result is to distribute in the network, it must contain context too. Thus the structure of an aggregated data should have the following structures:

Name of the data
Type of the data
Context of aggregation / Coverage
Originator ID
Generation time
Valid time
Aggregated value

Table 2-3 Contents of a Aggregated Data

2.4. Software architecture

In the above sections, we have described system model, spatial model and data model. Now we can introduce our software architecture of a node.

The software architecture of a node is layered. Application layer is on the top. Various applications can be installed on a node, e.g. as an advanced navigation system or a traffic information-gathering tool. Below the application layer is the data aggregation layer. It provides data aggregation service to applications, so that application can concentrate on abstract data. Data collection and aggregation are performed by the aggregation layer. They are invisible to the application layer. Applications only need to define queries for the interested data. Data aggregation handles the query and does data aggregation. Data aggregation layer is built on ad hoc routing layer, which is responsible for routing packets in mobile ad hoc network. A MAC layer and a physical layer are below the routing layer to manage the common access to the wireless channel. A spatial model is included in each node. In order to let application, aggregation and routing protocols to access the geographic information, it provides them a common.

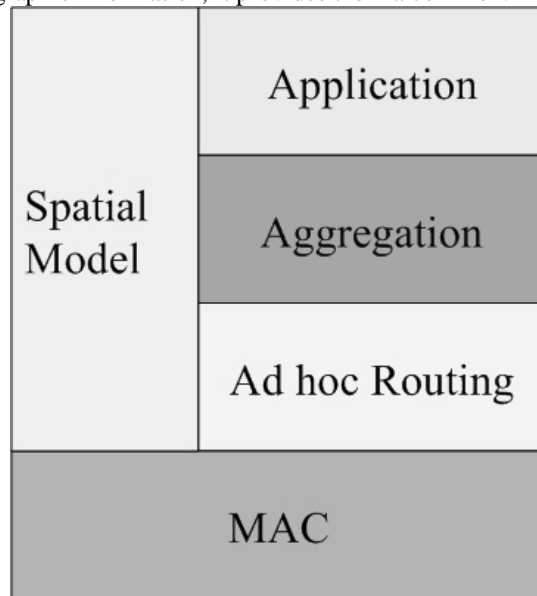


Figure 2-6 Software architecture

3. Related Work

This chapter gives an overview of the existing protocols for data aggregation in mobile Ad hoc networks. But prior to it, an overview of Ad hoc routing protocols is given. It is because our system model assumes no communication infrastructure is available and ad hoc routing must be used to forward packets inside a vehicular network.

3.1. Ad hoc Routing Protocols

Ad hoc routing is different from conventional routing protocols used in fixed computer networks. Due to the limited radio range of antennas, a packet from a sender can only reach neighboring nodes within its radio range. In order to forward a packet to a node or a location out of its radio range, a node must let intermediate nodes relay the packet. This kind of routing mechanism is called multi-hop forwarding.

Existing Ad hoc routing protocols can be classified into two categories: topology based and position based. Topology based protocols use information about the existing links in the network. Each node builds and maintains a routing table about the link state or distance vector. The obtaining of link information can be in either proactive or reactive manner. The proactive manner means each node broadcasts its complete routing information periodically and collects the information from other nodes, regardless whether the information is needed or not. The reactive manner means nodes only exchange routing information on demand.

Position based protocols rely on information about geographic positions of the nodes in the network. The protocols assume that each node is aware of its geographic position. Routing nodes do not maintain information about the network topology and link information. They make the forwarding decision based on their knowledge of local environment and spatial information inside the packet. Generally, when a routing node tries to forward a packet, it checks its neighboring nodes to find out one that has shortest Euclidian distance to the target position, and forwards the packet to that neighbor. Through so-called greedy forwarding, packets approach the target positions hop by hop.

Position based routing protocols have great advantages over topology based protocols for Ad hoc networks with high mobility, because the high mobility of nodes will cause the frequent and rapid change of network topology. When this changing frequency is faster than the link information obtaining speed, it is not possible for topology-based protocols to get correct link information anymore. On the contrary, position based protocols make forwarding decision only based on current position, instantaneous neighbors location and destination position. Complete knowledge of network topology is not necessary. So global topology change does not affect their performance greatly.

The following two protocols GPSR [KK00] and SAR [THRC03] are both position-based protocols. They both use geographic greedy forwarding, but have different solutions to solve greedy forwarding failures.

3.1.1. Position based Routing

1.1.1.1. GPSR

GPSR stands for Greedy Perimeter Stateless Routing. It is a popular position based routing protocol. Its system model is MANETs with mobile nodes. It relies on the assumption that each node can get its current position and is able to get other nodes' positions with the aid of a location service.

GPSR runs with beaconing mechanism. Each node periodically broadcast a beacon within its radio range, which contains node's identifier and geographical position, so that all the neighboring nodes can know its instantaneous position. Each node maintains a neighbor table to record its known neighbors. The entry of a neighbor table is soft stated and will be deleted after the related timer expires, unless a refresh beacon reset the timer. In order to save the bandwidth due to beaconing, GPSR optimize the performance by piggybacking the beacons on every packet that is sent. Additionally, all nodes run their network interface in promiscuous mode, so that each node within the sender's radio range can get the piggybacked beacon inside every packet sent.

GPS receiver can inform a node about its current position. Neighbor table maintains position information about its surrounding nodes and the location service interprets the target node's position. Those are all the information needed by GPSR to forward a packet. It has two working modes: greedy forwarding mode and perimeter mode.

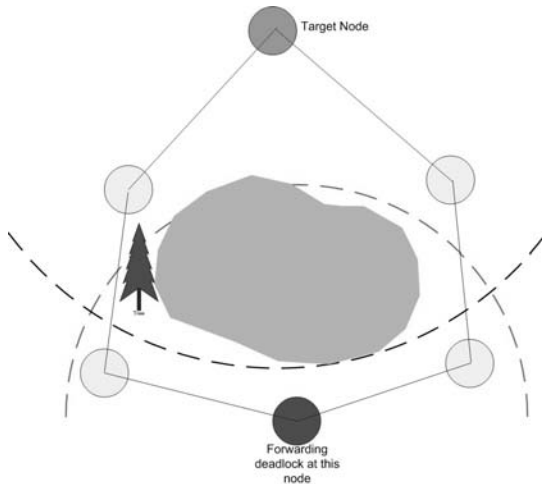


Figure 3-1 Voids in Network

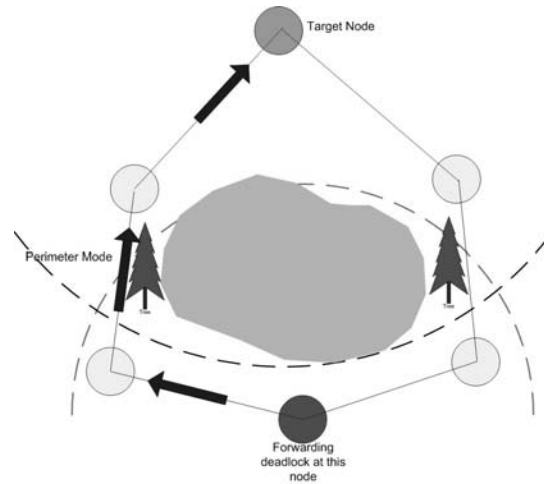


Figure 3-2 GPSR Routing in Perimeter Mode

Greedy forwarding is the default mode of packet forwarding by GPSR. When a node receives a packet, it first checks whether it is the target node. If not, it will forward the packet. First, it tries to check if the target node is listed in its neighbor table. If yes, it means the target node is directly reachable (within its radio range). So it will send the packet to it directly. If the target node is not its neighbor, then it will forward the packet to a neighbor, which is listed in its neighbor table and geographically closest to the target node. This process keeps on till the packet reaches the target node. The greedy forwarding algorithm does not depend on any topology knowledge of the network. No routing table is needed at all. Routing nodes handle the incoming packet by checking the target position and their current neighbor tables. So it is a stateless solution.

Greedy forwarding works well and efficiently when there is no distribution hole of nodes between the packet originator and the target. But distribution holes, which are called *voids*, can be caused by geographical obstacles like mountains or large buildings. In case that if there is a *void* between the packet originator and target, then the greedy forwarding may get deadlock at a node on the board of the *void*, because the node may not find a neighbor that is geographically closer to the destination than itself. (Depicted in Figure 3-1) The perimeter mode of GPSR is used to overcome the possible network *voids*. The idea is to use “right hand” rule to detour *voids*. The protocol works as follows: each packet starts with greedy mode. When the packet reaches a node on the board of a *void* and fails to find a neighbor geographically closer to the target than itself, that routing node switches the routing state of the packet, which is a variable in the packet header, from `GPSR_GREEDY` to `GPSR_PERIMETER`. Then the routing node will choose a neighbor based on “right hand” rule as the receiver of the packet. When that neighboring node receives the packet, which is in `GPSR_PERIMETER` state, it will first check whether there is a neighboring node geographically closer to the target than itself. If yes, the packet will return to greedy state and be forwarded to that neighbor. But if no such neighbor is available, the packet will keep on being forwarded in `GPSR_PERIMETER` state.

1.1.1.2. SAR

SAR stands for Spatially Aware Routing, which is another position-based routing protocol for MANET. Mobile nodes are equipped with wireless communication interfaces and GPS receivers. The movement of mobile nodes is constrained by road networks. SAR is the first routing protocol that uses the environmental spatial information. The spatial model, which is the basis for routing algorithm, is to provide common, high level abstractions of spatial objects and relationships. The author proposed an easy way to construct a spatial model. That is based on spatial information extracted from Geographic Information System, such as digital maps.

To forward a packet from the sender to the target node, SAR allows the sender to define the routing path specifically by a list of geographic positions, so-called “waypoints” in the packet header. Such an algorithm is called geographic source routing (GSR). The use of geographic routes is suitable for vehicular networks, because such a route is stable and not affected by the nodes’ mobility. Before sending out a packet, the sender needs to compute the route from the sender to the target. Result route of Dijkstra’s shortest path algorithm can be used as default routing path. The sender will then have its spatial mode translate the route into a list of waypoints, which will be inserted into the packet header.

As the packet travels along the given waypoints, it will reach the waypoints one by one. The waypoint, that a packet is being forwarded to, is called the current waypoint. SAR manages the sequential activation of new waypoints. The current waypoint get obsolete if the packet reaches a node whose distance to the current waypoint is smaller than a certain threshold, then the next waypoint is activated.

When a packet is being forwarded to the current waypoint, geographic greedy forwarding is used. When a routing node receives a packet, it will first check if the target is directly reachable. If not, it checks if it is already closer enough to the current waypoint. Otherwise, it checks its neighbor table to find a neighboring node, which has the minimum Euclidean distance to the current waypoint. In case that it successfully finds such a neighbor, it will set the address of that node as the destination on MAC layer and forward the packet to it.

If there is no network partition or *voids* along the routing path, a packet will reach the waypoints one after another and finally arrive the last segment of the route, where the target node is located. However, vehicle networks often have partitions, especially during the early deployment phase, when only small number of vehicles is equipped with the required communication devices. SAR solves the problem of network partition by using extended spatial information. It is further assumed that each node knows its movement destination of the near future. That means each node knows the next and the next but one intersection it will travel. This information is included in the beacon messages, so that a node can know the future movement trajectories of its neighbors.

When a routing node finds no suitable neighbor is currently available, instead of dropping it, SAR suspends the message delivery for a certain period of time and waits for new nodes coming into its

communication range. The suspended packet is put into a local so-called suspension buffer. The node will frequently review each packet with respect to availability of a suitable neighbor that has recently come into communication range. If none is found that is geographically closer to the next waypoint, the neighbor list is checked for the node whose next but one hop lies on the GSR of the packet. If the node has finally found a suitable neighbor the packet is forwarded to that node. With such extension of the basic greedy forwarding, SAR enables a packet be delivered to the target position with high success rate, even in case of sparse network.

1.1.1.3. GEOCAST

We notice that GPSR and SAR both use location service to resolve the address of target node to target position. However, since we are lack of an infrastructure or globally manage system in MANET, there is no central server for a location service possible. Therefore it is mandatory for the sender to trigger an address to position resolution procedure before a packet is sent out. This procedure takes time, costs bandwidth and also introduces additional delay.

However, in many cases, sender does not care about whom it is contacting with, but where it's contacting. In other words, sender only wants to contact with one or some nodes locally at the position it desires. In such cases, there is no need to specify target node in the packet at all. Furthermore, even if the target is one single node, it would be good to specify an area instead of a point to indicate the node's position, because the target node might have changed its position due to mobility, during the time when the packet approaches. Therefore sender needs to define a geographic area where it wants the packet to be forwarded to.

The term Geocast was invented to refer to a variation of conventional multicast. Different from the unicast, in which only one node is set as destination in the packet header, Geocast specifies the destination by a description of a geographic region. The representation of the geocast region can be a rectangular or a circle. All nodes inside that region should receive and process the packet.

Up to now, there are several geocasting protocols proposed, such as LBM, GeoGRID, GeoTORA and mesh based Geocast routing. All those protocol use plain flooding inside geocast region. Their difference is how a packet is forwarded into the geocast region. Here we give introduction of LBM and GeoGRID, which are representatives of two categories: flooding based and routing tree construction based.

1.1.1.4. Location based Multicast

Location based Multicast [KV98] is the first published flooding based geocast protocol. It's objective is to decrease delivery overhead of geocast packets, by reducing the forwarding space for geocast packets, while maintaining accuracy of data delivery. Instead of flooding a geocast packet to the whole network, LBM defines conditions for a node to decide whether to keep on forwarding a received packet.

To reduce the space of flooding, LBM gives two schemes. The first one defines an area, so-called

forwarding zone, for a geocast packet, where flooding of the packet is allowed. All nodes in the forwarding zone forward the packet to their neighbors. When a node receives a geocasting packet, it will discard the packet if it is not inside the forwarding zone. The simplest shape of forwarding zone is a rectangular, which includes the sender and the geocasting region.

The second one does not define a forwarding zone explicitly. Instead, it defines a distance increment parameter δ . Whether to forward a geocast packet is based on the position of the current node and the position of the geocast region. When a node B receives a geocasting packet from A, it checks whether it is at least δ closer to the geocast region than A. ($\text{Distance}_A \geq \text{Distance}_B + \delta$). Only when it is true, node B can forward the packet to its neighbors. This approach ensures that after every transmission the packet gets closer to the geocasting region.

1.1.1.5. GeoGRID

GeoGRID is based on the unicast protocol GRID. GeoGRID uses location information to define the forwarding zone and elect a special host, i.e. gateway, in each grid area that is responsible for forwarding the geocast packets.

This protocol partitions the geographic area of the MANET into many logic grids. In each grid a gateway node is elected. Gateway is the only node that does packet forwarding in a grid. Therefore, when sender sends a packet to geocasting region, the packet is only forwarded by the gateways in the forwarding zone.

GeoGRID use two methods of distributing geocast messages. Flooding based method allows any gateway in the forwarding zone to rebroadcast the message. Ticket based method allows only ticket-holding gateways to rebroadcast. The author states that by issuing tickets blind flooding can be avoided.

Within a geocast packet, number of tickets is included in the header. The sender set the number of tickets proportional to the size of geocasting region. The sender will send the packet to its neighboring gateways, which are closer to the geocasting region. If the initial number of tickets is T (T is equal to the number of grids in the geocasting region) and there are N neighbor gateways are closer, then the sender will send each of them a packet. But the number of tickets inside the packet is evenly divided. So each of the neighboring gateways receives a packet with only T/N tickets. Then they will keep on forwarding the packet to the geocasting region according to the above algorithm. When a gateway receives a packet from another gateway, it will not discard the packet, though it has received one before. It will forward the packet further, because the author wishes to use this redundancy to increase the arrival rate of the geocast packet.

Gateway election is very important for GeoGRID. It is proposed to let the node which locates nearest to the physical center of a grid become the gateway. Another solution is to base on weight. The weight of a node can be inversely proportional to its speed. When nodes inside a grid do not know one existing gateway for the grid, they start to elect gateway by sending out BID messages. When a node hears a BID, but finds it is closer to the center of grid than the BID originator, it will

send out its BID to suppress the heard one. Eventually no node is able to suppress the BID from the node closet to the center of the grid. Then that node becomes the gateway. Later on, the gateway will periodically broadcast GATE message to inform other nodes about its existence. If due to temporary link failure a non-gateway node fails to hear a GATE message within a predefined time period, it will broadcast BID to try to start a new election. But the gateway, if existing, will suppress this BID by its GATE message. When the gateway leaves its current grid, it should broadcast a RETIRE packet to trigger a new round of election in the grid.

Therefore, as long as a node is elected as gateway, it will take the responsibility till it moves out of the grid. But in some cases, due to temporary network partition, multiple gateways would appear in one grid. To solve this problem, the protocol suggest that when a node, who assumes itself as the gateway, hears a GATE packet from another gateway from a location closer to the center of its grid, it will silently turns itself as a non-gateway without sending any packet.

3.1.2. Summary of Ad hoc routing

Ad hoc routing for MANETs does not have standard protocol yet. The performance of protocols largely depends on the system model of applications. For vehicle networks, which have the characteristics of high mobility and rapid topology changing, position based protocols have advantages because their forwarding decision is stateless and only based on neighboring geographic information. SAR takes consideration of spatial model and movement trajectories, therefore it results in a high success rate of packet forwarding in sparse network. It is a good choice for vehicular network routing protocol. Apart from it, we noticed that in vehicular networks, it's not possible to obtain the exact position of a continuously moving node, and in many cases, where a node is contacting is more popular than to whom a node is contacting. So geocast must be supported by Ad hoc routing protocols. LBM gives the idea of incremental-distance δ and GeoGRID gives the idea of network division and gateway election. Both of them are good ideas for protocol design.

3.2. Data Aggregation in Ad hoc network

MANET applications often require cooperation among a large number of nodes. One example is to continuously monitor an area and report events. Another example is a node sends out a query about interested data to a number of nodes. Thus many individual data needs to be collected and extracted to form some higher-level information. The procedure of data collection and extraction is called data aggregation.

To perform data aggregation in MANET many problems has to be settled: like limited power, unstable network topology, etc. The communication cost is the biggest problem for data aggregation, because wireless link is an open media with limited bandwidth. Forwarding large number of data introduces the network much communication traffic load. In addition, much communication will drain the battery power of sensors quickly. Thus how to reduce the communication cost must be an important factor of the aggregation protocol.

Furthermore, unlike the conventional fixed network, where TCP/IP has been selected as the standard routing protocol, MANET does not have a standard routing protocol. Thus developing a data aggregation protocol must be based on the selected routing protocol. In the following sections, we give an overview of the data aggregation protocols developed by other groups. We will first study them and later give a conclusion of those protocols taking vehicular networks into consideration.

3.2.1. Hierarchical Gossiping Aggregation

Hierarchical gossiping aggregation [MFHH02] aims to be a fault-tolerant, scalable solution to the problem of accurately and scalably calculating global aggregates in large MANETs. The author states that, in real life networks member nodes are prone to arbitrary crash and recovery, therefore it is impossible to have a correct protocol that always calculates the exact aggregates value at all member nodes. He strives to reduce the message and time complexity of the protocol and increases the completeness of the final result.

The work is based on a system model where a large number of nodes residing in an unreliable asynchronous network. An underlying routing protocol enables any member node to send messages to any other member. Furthermore each member node has a list of other member nodes it knows about.

This work is the first one dealing with the scalability problem of data aggregation in MANETs. The author first points out that in case of large number of member nodes, neither fully distributed nor centralized solution is scalable. Because for fully distributed solution, that each node sends its data to every other member nodes, the message complexity is $O(N^2)$, which is not affordable when N is large. For centralized solution, that each node sends data to a special member node denoted as a leader, large amount of data from all other member nodes will overload the bandwidth limited wireless link at the leader, thus cause link failure or data loss.

The author thus proposes a Grid Box Hierarchy based solution. The idea of Grid Box Hierarchy is to divide the whole N member nodes into N/K grid boxes with an average of K members per grid box. K is a constant integer chosen independent of N and is well known to all member nodes.

Then the global aggregation is calculated bottom-up in this hierarchy in $\log_k N$ phases. In the first phase, each group member estimates the aggregate of data of all members in its own grid box. In each following subsequent phase, each node evaluates the value of aggregate over the set of aggregates belonging to the same height subtree as itself.

Therefore the key problems to solve are 1) what mechanism does one use to build such a hierarchy and 2) what is the actual protocol used to calculate the global aggregate in the hierarchical manner described above.

The author proposes to use a well-known hash function H to construct the hierarchy. The purpose

of the hash function is to map each node to a grid box of K members. The hash function given by the author requires public knowledge of H , K and N . H maps the unique node identifiers randomly into the interval $[0,1]$. Then a node with identifier M_j would belong to a grid box with the address $H(M_j)*N/K$. In addition, notice that any arbitrary group member M_j can calculate the grid box address of any other group member M_l (that is present in its view) this is simply $H(M_l)*N/K$. Thus at each phase of the global aggregate calculation, node M_j would know about all the members in its view that belong to M_j 's height- i subtree in that phase. Another hash function proposed by the author is geographic location based, but he doesn't give concrete solution of it.

With the hierarchy by the constructed by hash function, the aggregation can be done in the bottom to up manner described above. The author first analyzes a leader election approach and claim random message delivery failures and process crashes can arbitrary affect the completeness of the aggregate value. Then he introduces the gossiping approach. The algorithm of the approach consists $\log_k N$ phases.

At phase 1, a node M_j gossips with members in its own grid box, about individual data of itself or the data heard. After $K \log N$ gossip rounds, M_j aggregates the known data from members of its grid box.

At phase i , in every gossip round in phase i , M_j choose a few gossipees randomly from the set of all members in the same subtree of height i as itself. M_j then sends these gossipees a randomly selected aggregate value from among the known aggregates for the height $(i-1)$ child subtree of M_j 's height- i subtree. When M_j has either managed to obtain the values of the aggregate for all its $(K-1)$ sibling sub-trees, or have $K \log N$ gossip rounds it do aggregation.

At final phase, when M_j finds itself in phase $\log_k N + 1$, it has estimates of the global aggregate. The protocol then terminates at M_j .

3.2.2. Greedy Incremental Tree with Directed Diffusion

Aggregation with greedy incremental tree is a data aggregation protocol based on directed diffusion [IEGH02]. Directed diffusion is an approach to attribute-based communication for wireless sensor networks. It is application oriented. Data is named using attribute-value pairs. When a sender (source) wants to disseminate a sensing task throughout the sensor network, it will send out an interest for named data. Nodes that hear and match the interest, i.e. sinks, will send out gradients to the source. The gradients take the routes with minimal delay. When source receives gradients, it will start to draw data from the sink along the route reinforced by gradients.

If a large number of nodes match the interest, multiple data can be aggregated in the network, when they flow to the interest originator along routes with gradients. But in order to minimize the

total communication cost for data aggregation, the author proposes that the data will be routed along a newly created greedy incremental tree to the source instead of the route with gradients.

The algorithm of Greedy Incremental Tree is described now. Each event contains an additional attribute E , the energy cost for the delivering this event from the source to the current node. When source detects phenomena, it sends exploratory events with $E = 1$, to each neighbor for whom it has a gradient. After a node receives these previously unseen exploratory events. It adds the cost of that transmission to E before sending. Though such exploratory message is useful for selecting a lowest-energy path, it is not sufficient to construct a greedy incremental tree. To construct a greedy incremental tree that can be shared by multiple nodes, each source on the existing tree also generates incremental cost message, once it receives a previously unseen exploratory message generated by other sources. The incremental cost message is only sent along the existing tree using data gradients. Unlike the energy cost E , the incremental energy cost C can be decreased. Once a tree node receives a previously unseen incremental cost message, it searches for the corresponding exploratory event in its message cache, the node updates c of the incremental cost means to the minimum value between the current and the energy cost E of the exploratory event retrieved from the cache. Before sending the increment cost message to its outgoing data gradients.

Unlike the exploratory event, the incremental cost message contains the minimum energy cost for delivering data from a new source to the existing tree

The interest is periodically sent to neighbors. However, in high mobile environment, the neighborhood relation changes quickly, and their relative position keeps changing. When the path near the event source is reinforced, the topology near the sink might already change. Therefore it's hard to generate a stable reinforced path to transfer the event. Localized interaction and data aggregation. Combining data from several sensors, to save data overlapping within some vicinity. Avoid sending data over a long distance as much as possible

3.2.3. Tiny Aggregation

Tiny Aggregation (TAG) is a service for data aggregation in sensor networks, which consist of small battery powered, computation capable, wireless communication devices.

TAG process aggregates in the network by computing over the data as it flows through sensors, discarding irrelevant data and combining relevant readings into more compact records when possible. In order to deliver the query request to all nodes in a network and route the raw data to the Query Sender, a routing tree must be constructed beforehand.

In TAG, the node that sends out the query request is named root. To construct a routing tree, the root first broadcasts a message. In that message it specifies its own ID and its level or distance from the root. Any node without an assigned level that hears this message assigns its own level to be the level in the message plus one. It also chooses the sender of the message as its parent, through which it will route message s to the root. Each of those nodes then rebroadcast the routing message, inserting their own IDs and levels the routing message floods down the tree in the

fashion, with each node rebroadcasting the message until all nodes have been assigned a level and parent.

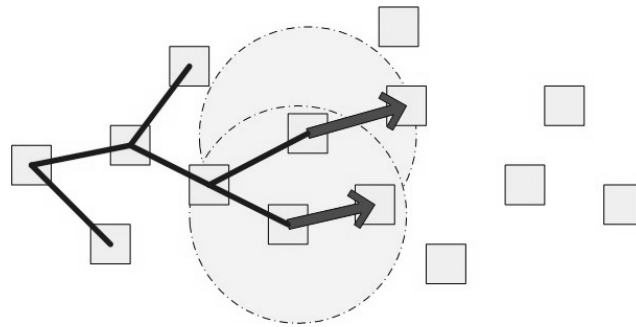


Figure 3-3 Construction of routing tree

In order to overcome the problem of dynamic topology, the routing discovery process is performed periodically. They further extend the algorithm to do better topology maintenance and recovery from temporary loss of the wireless link or node failure. Each node maintains a small fixed sized list of neighbors and monitors the quality of the link to each of those neighbors by tracking the proportion of packets received from each neighbor. When a node n observes that the link quality to its parent p is significantly worse than that of some other node p' , then it choose p' as its new parent if the node p' is as close or closer to the root as p and p' does not believe n is its parent (to prevent routing cycles). If a node n does lost its link with its parent node for some time, then it can start choosing a new node as the new parent. In this case, the node n can even select one node in the routing subtree underneath it as its parent. All child nodes must reselect their parent when they observe that their own parents level has gone up. The construction of a routing tree described above can help the node in a sensor network routing packets to other nodes without the priori knowledge of the network topology. It works well even in case the temporary failure of nodes or part of network.

With the help of the routing tree, TAG aggregates the data in the network. Their proposed aggregation approach consists of two phases: a distribution phase, in which aggregate queries are pushed down into the network, and a collection phase, where the aggregates are continually routed up from children to parents.

The distribution of an aggregate query can be used to construct the routing tree, so that in the collection phase, data can be routed along the tree structure. But in order to let parent nodes combine its own data with the aggregates from its subtree, parent nodes must wait till they hear the aggregated data from their children. TAG accomplish this by forcing a child node deliver its raw data or aggregate of its subtree to its parent within the time interval specified by its parent. The interval is defined by parents and passed to their children. When a node receives a delivery interval from its parent, it will give its children a delivery interval shorter than its own. So that in the collection phase, after it receives aggregate from its children, it has some time to include its own data in the aggregate and transmit it. Its aggregate can then reach its parent on time.

Apart from the aggregation approach, TAG also gives an innovative statement. They claim that the

aggregation service is central to emerging sensor network applications, so that aggregation must be provided as a core service by the system software. Such a service should have generic, simple declarative interface. They proposed the interface following the SQL-style. Basic operations like MAX, MIN, COUNT, SUM, AVERAGE, MEDIAN, and so on should be supported by the aggregation service.

3.2.4. Digest Diffusion

Sensor networks consist of large number of small wireless, battery powered and low mobility sensors. In order to continuously monitor the overall state of networks, a monitoring architecture is needed. Author proposes an efficient protocol to compute aggregates of network properties, the so-called “digest”, with the aid of a tree structure, which is called “digest tree” by the author.

A novel monitoring architecture is provided in this work. Three tools are introduced to work complementarily. The first one “digest diffusion” continuously collects aggregates of network in the background. Triggered by sudden changes in these properties, the second tool “scans” can be invoked to provide global yet aggregated views of system state. Such views can indicate the location of performance problems or impending failure within the network. Finally tool “dumps” can be used to collect detailed node state and debug the problem. The three tools can be invoked at different spatial and temporal scales and will allow accurate yet low-overhead sensor network monitoring.

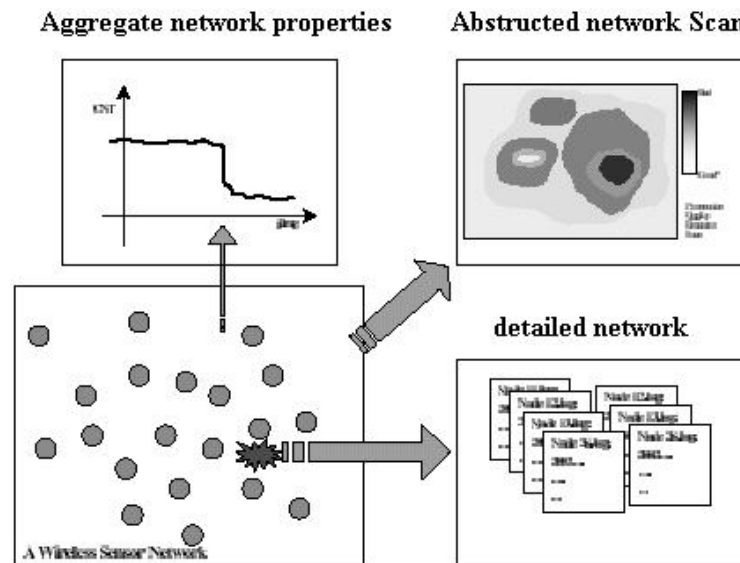


Figure 3-4 Network Monitoring architecture

To compute the digest, i.e. network properties, “digest diffusion” is proposed. It is an in-network aggregation protocol. Each node computes a partial result of the digest and passes the result to other neighboring nodes. (Here all digest are assumed to be decomposable) Partial results are propagated up the tree towards the root.

The algorithm to construct a digest tree follows: Each node first create a tuple $\langle Mi, si, hi \rangle$. Mi is the perceived maximum, its own value at first, si is the source of maximum, hi is the hop distance to maximum. Then the node exchange tuples with neighboring nodes. On receiving a tuple $\langle M_j, s_j, h_j \rangle$ from node j , a node checks whether $Mi < M_j$. If $Mi < M_j$, then this node set j as its parent and set $Mi = M_j$; $si = s_j$; $hi = h_j + 1$; if $Mi = M_j$ then the node compares if $s_j > si$, in order to get a strict maximum. If a node receives another tuple from k , whose $Mk = Mi$ and $sk = si$, then node j will switch its parent to k if $hk < hi$, (k is closer to the maximum). Through such an algorithm a digest tree is constructed, whose root has the maximum value. The author proposes to construct a tree with maximal residual battery energy, with which a tree's life is expected to be long.

Apart from the exemplary values like maximum or minimum, other digest can also be computed with digest tree, such as average. The partial result sent by a node now contains not an exemplary value, but a pair $\langle n, avg \rangle$. When a node receives a result, it can compute the new result,

$$n_i = \sum_{c=1}^k n_{c_j} + 1; a_i = \frac{\sum_{c=1}^k n_{c_j} \cdot a_{c_j} + v_i}{n_i}$$

Thus, with the help of digest tree, network properties can be monitored. The message exchange only happens between neighboring nodes, therefore such a scheme is energy efficient.

However, the author observes that 1) packet loss and link asymmetry is prevalent in wireless networks and 2) time-varying loss and asymmetry can result in oscillating digest tree branches. Those two factors cause significant error in the computed digest. To avoid that, the author proposes to selectively "blacklist" links with poor quality. That is, when possible, each node will try to choose a parent with which it has good and symmetric communication. The performance turns much better with such a modification.

3.2.5. Summary of data aggregation

The above-mentioned protocols aim to get aggregated data while reducing communication cost. The raw data from individual nodes are aggregated in the network when they flow to the sink. The aggregation is based on tree structures, which is constructed for routing or forwarding. The structure of the tree is based on nodes. Tiny Aggregation and eScan both periodically construct a routing tree by control messages. GIT based on the gradients and incremental messages to construct a greedy incremental tree. Each of them considers node failure and link failure and can repair the tree with the aid of recovery methods. The hierarchical gossiping aggregation uses a different strategy. It gives the idea of divided the whole area into grids, and aggregation is done in several phases. In each phase, aggregation is done on a higher level.

We notices that ad hoc routing protocols are closely related to data aggregation protocols. Some proposal even merged them together like GIT (together with Directed Diffusion).

4. Conception

In the above two chapters, an overview of the system and the related work are presented. Intelligent vehicles are capable of generating raw data and process information. When they are connected by wireless links, they form vehicular networks. Intelligent vehicles, i.e. mobile nodes, can exchange raw data within the networks by ad hoc routing protocols. But to get aggregated data, raw data from many mobile nodes must be collected. In this chapter we show how to do data aggregation and which data aggregation strategy is suitable for vehicular networks.

4.1. Objective of Data Aggregation

The objectives of data aggregation in vehicular networks are 1) to get network properties 2) to complete cooperative tasks

Get Network Properties

Network prosperities are useful to mobile nodes, because they can affect node movement patterns or traveling routes. To get it, raw data from many nodes should be gathered and evaluated together. An example of network property is the traffic density of an area. It can help a node to be aware of the distribution of nodes in the road network. By collecting individual information from each node in that area, the density can be computed.

Finish Cooperative Tasks

Apart from monitoring networks, another type of application for intelligent vehicles is to do a surveys or queries on number of nodes. A node sends a query to an area about a certain even. Every node receiving the query should participate in that cooperative task, if it meets the query requirements. They generate their raw data to response the query. Raw data from different nodes are aggregated to give an answer to the Query Sender. An example of cooperative task is “Tell me the number of free spaces left at the auto park place in area A, if you can observe one in your vicinity.” Observation from a single node is only a piece of exemplar with limited credibility. Observations from many nodes will increase the credibility and accuracy of the result.

4.2. Requirement of Data Aggregation

A data aggregation strategy for vehicular networks should be able to solve the challenge of high mobility. In addition, a good data aggregation strategy should be scalable to the size of the network and the number of the queries. When large amount of nodes are involved in data aggregation, the data aggregation strategy should prevent the wireless communication link from blocking. In the following section, different aggregation strategies are classified. Detailed introduction and analysis are given to each of them.

4.3. Classification of Aggregation Strategies

Based on how and where the data aggregation is performed, we can classify the data aggregation strategies into following categories

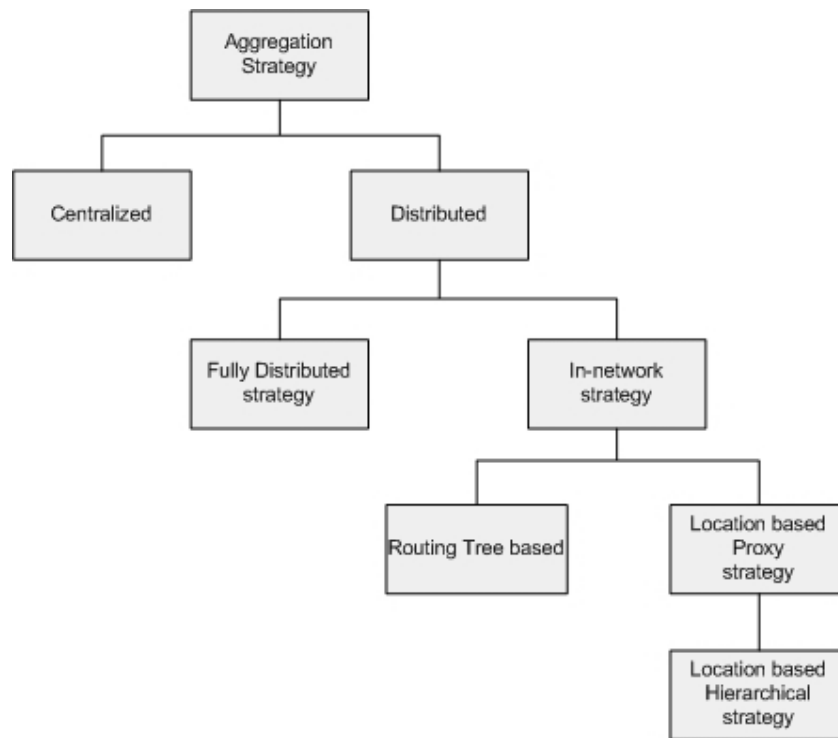


Figure 4-1 Classification of Aggregation strategies

The classification shown in Figure 4-1 has many levels. The criterion of first level is “Who perform data aggregation: the Query Sender or other nodes”. If the Query Sender performs the aggregation itself, then it is centralized strategy, otherwise is a distributed strategy. Within distributed strategies, further classification is done based on “which nodes perform aggregation, all nodes that are queried or some other nodes”. If every Query Responder does aggregation, then it is a fully distributed strategy. If only some nodes do aggregation, then it is in-network aggregation, which means raw data are aggregated when they flow to the Query Sender. For the in-network aggregation, it can be further classified by how the aggregation is performed in the network. We found two strategies belong to the in-network strategy: routing/aggregation tree based strategy and location-based proxy strategy. The routing t/aggregation tree based strategy is based on a hierarchy of nodes, while the location based proxy aggregation is based on a hierarchy of geographic locations.

Following terminologies will be used in the description of aggregation strategies:

- **Query Sender:** the node that starts the query
- **Query Responder:** the node that responses the query
- **Aggregator:** the node that collects raw data and does data aggregation

- **Query:** a query that defines the data to be queried, nodes to be queried and requirements.
- **Raw Data:** data from a Query Responder.
- **Aggregated Data:** aggregated result of number of raw data.

- **Query message:** the message that contains a query

- **Response message:** the message that contains a piece of raw data
- **Aggregate message:** the message that contains a piece of aggregated data.

4.3.1. Centralized Strategy

If the Query Sender collects all response messages and does data aggregation by itself, then such a strategy is called centralized-aggregation. For centralized-aggregation, all Query Responders send raw data to the Query Sender. Each response message is routed independently.

4.3.2. Fully Distributed Strategy

If all Query Responders do aggregation, then such a strategy is called fully distributed aggregation. Upon receiving a query message, a Query Responder will send its response message to other Query Responders. When the period for collecting response message is over, each Query Responder will do aggregation by itself and generate aggregated data. Only one aggregated data needs to be sent back to the Query Sender.

4.3.3. In-network Strategy

In order to save the message transmitted in the network for a query task, in-network aggregation is used. It means raw data are aggregated when they flow to the Query Sender. The Aggregators are distributed in the network. Based on how the aggregators are chosen, the in-network strategies are classified into routing/aggregation tree based and location based proxy strategies.

4.3.4. Routing/Aggregation Tree based Strategy

Routing/Aggregation tree based strategy is one of the in-network aggregation strategies. It is a common data aggregation strategy used in sensor networks. Choosing aggregators is based on a routing/aggregation tree structure. Before raw data are sent to the Query Sender, a tree is constructed with certain methods. This tree is based on nodes. Then the response messages are routed along this tree.

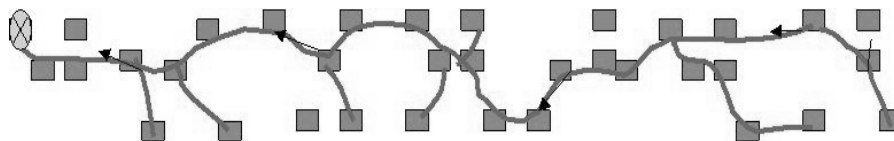


Figure 4-2 Tree based strategy

When a node receives data from its sub-tree, it will collect them and combine them into one aggregated data, which will be forwarded to its parent. Thus it effectively suppresses the number of messages in the whole network.

4.3.5. Location based proxy Aggregation

Another kind of in-network aggregation is location-based proxy aggregation, which is proposed by us innovatively. Instead of selecting specific nodes to take the responsibility of aggregators, this strategy selects the aggregator based on locations. That location is called “aggregation area”. All the nodes inside the aggregation area become aggregators. Response messages are sent to the aggregation area by ad hoc routing. The aggregators will collect the messages and combine them into aggregated data.

Figure 4.3 illustrates different aggregation strategies.

y

Figure 4-3 Illustration of Aggregation Strategies

4.3.6. Analysis of Tree based Strategy

Tree based strategy relies on the routing/aggregation tree, which must be constructed before the raw data are sent back to the Query Sender. Several different approaches have been proposed to the tree construction [IEGH02][MFHH02][ZGE03]. Generally, control messages or the query message is used to construct the tree. However, this strategy has following problems in vehicle networks.

Difficulty in Tree Maintenance

Constructing of a stable routing/aggregation tree is impossible in vehicular networks. The movement speed of vehicles is much higher than that of simple sensors in sensor networks. When nodes travel at speed of 120kpm and with radio range 250m, the duration for the connection between two nodes in opposite direction is only 7.5 seconds. Therefore, even a tree is constructed, a node will lost its connection with its parent or child frequently. If the recovery protocol is used to repair the tree structure, then the recovery procedure must be called frequently. Even if the repairing messages do not over burden the network, the time for convergence is unpredictable and the resulting hierarchy is still unstable. Therefore, tree maintenance in vehicular network is not applicable.

Accumulated Delay

Apart from the unstable tree structure, accumulated delay is also a big problem. In a tree, a node cannot send an arrived message from one of its child immediately. It must put it to the buffer and wait for messages from all of its children. Only when all data arrived, it can do aggregation to combine those data into an aggregated data and send it to its parent. Obviously, its parent must wait longer than this node. Therefore, there is an additional delay on each level of the tree, as shown in Figure 4-3. Therefore reduction of transmission messages by in-network aggregation is with the cost of longer delay.

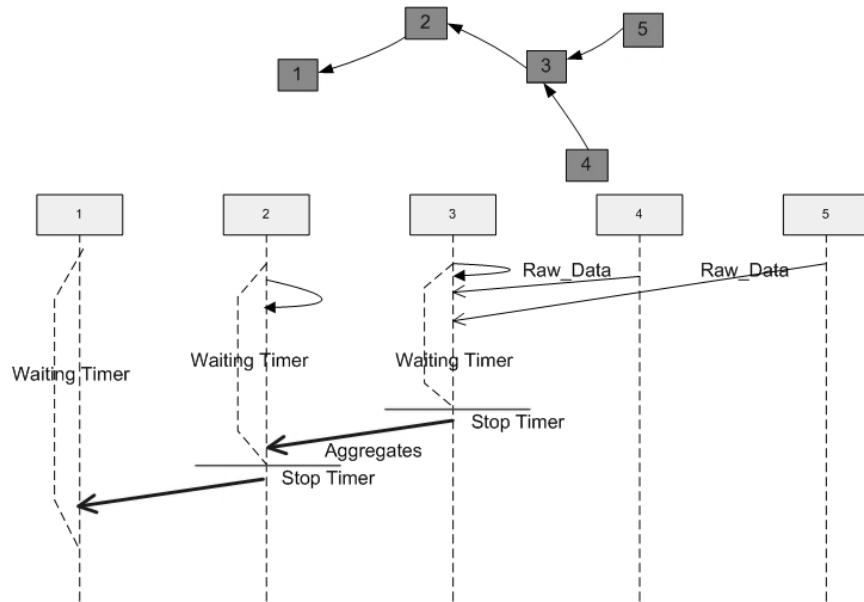


Figure 4-4 Message Sequence Chart of In-network aggregation

In addition, we noticed that the distribution of mobile nodes in vehicular networks is constrained by road networks. Then the constructed tree will be extremely deep. For a deep tree, the problem of accumulative delay will be very serious

Lacking Scalability

Because there is no stable routing tree exists in vehicular networks, it is necessary to construct a routing tree for each query operation. In vehicular networks, each mobile node has the same privilege to start queries. Hence the number of queries exists in the network contemporarily can be much larger than that in a censor network. When there are many queries, many routing tree need to be maintained. Such a strategy then is not scalable.

Due to the above reasons, routing tree-based aggregation is not applicable to the vehicular network.

4.3.7. Modeling and Mathematical Study:

In order to study the properties of the rest strategies, mathematical models are established to give a comparison of the strategies.

First we give the model of our system model: nodes running on road networks. If we assume Query Responders are located in a geographical interlinking area, namely on a continuous road, then response messages are routed along that road. In addition, if there is no loop in the routing path, we can simplify our model to a long strait road, starting from the Query Sender and ends and the query destination area, where Query Responder is located.

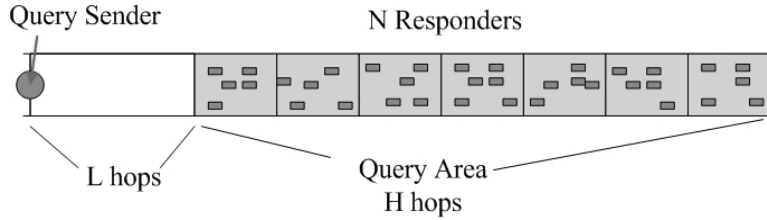


Figure 4-5 Simplified System Model

Parameters

For simplicity we assume nodes are evenly distributed inside the destination area. N respondents want to send messages to a query sender by multi-hop forwarding. The size of query destination area covers H hops. The minimum distance between the Query Sender and query destination area is L hops.

Centralized Strategy

Using centralized strategy, each message will be forwarded independently to the Query Sender.

For the nodes locates farthest to the Query Sender, their messages will be forwarded at least H times inside destination area, and additional L times along the routing path from the destination area to query sender. There are N/H nodes in such case.

Similarly, messages from the nodes, which locate one hop closer to the Query Sender, will be forwarded at least $H-1$ hops inside destination area and L hops along the routing path.

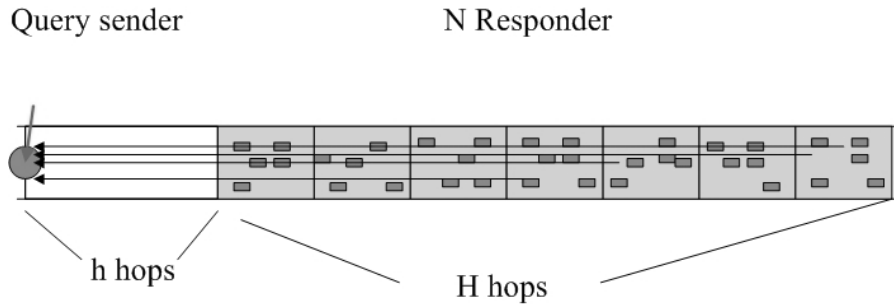


Figure 4-6 Centralized strategy

So the total amount of messages needed inside destination area is:

$$C_{queryarea} = H \times \frac{N}{H} + (H - 1) \times \frac{N}{H} + \dots + 1 \times \frac{N}{H} = \frac{NH}{2}$$

Because there are N responds sent to the Query Sender, so the amount of messages in the network (the network not including query area) $C_{forward} = NL$

Fully Distributed Aggregation

Using fully distributed aggregation strategy, each Query Responder will send its own data to all other Query Responders. If plain flooding is used, then each piece of response message will cause N rebroadcast messages, thus total number of messages in the query area is $N * N = N^2$.

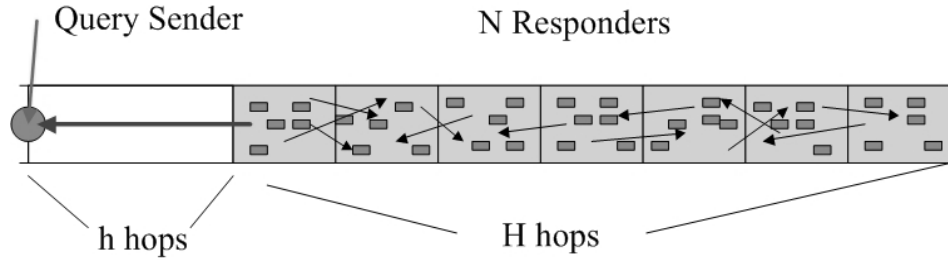


Figure 4-7 Fully Distributed aggregation

So the total amount of messages needed inside destination area: $C_{queryarea} = N^2$

When there is only one aggregate is sent to the Query Sender, then the cost in the network is L message. $C_{forward} = L$

Location based Proxy Strategy

Using Location Based Aggregation, the Query Responders will send their response message to the aggregators. It is assumed that the number of aggregator is M. The distance between the query area and the aggregation area is $L2$ hops. The distance between the Query Sender and aggregation area is $L1$ hops

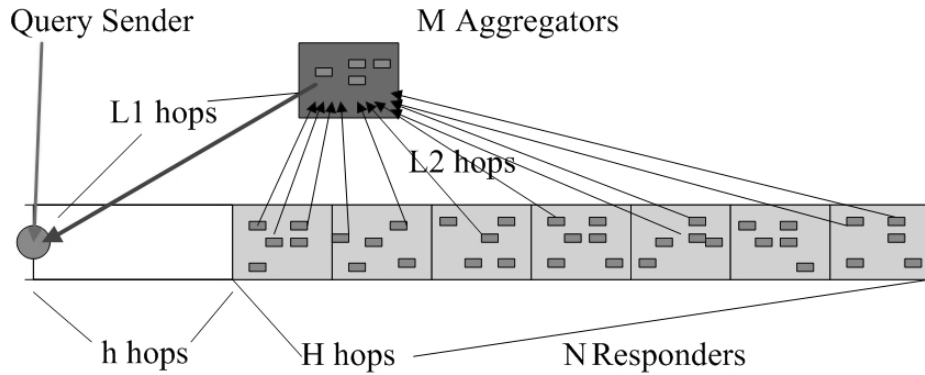


Figure 4-8 Location Based aggregation

N response messages first need to be sent to aggregation area, which will cause $NH/2$ messages in the query area and NM inside aggregation area. The communication cost in the network has two parts: 1) from the query area to aggregation area cause $NL2$ messages; 2) from the aggregation area to the Query Sender cause $L1$ messages.

Therefore the communication cost in query area is $C_{queryarea} = \frac{NH}{2}$; in aggregation area is

$$C_{agg} = MN$$

Communication cost in the networks is $C_{forward} = NL2 + L1$

4.3.8. Comparison and Summary

Table 4-1 shows a summary of the calculation. From it, we can find out the impact of scenario parameters on the communication cost with each strategy.

	Inside query area	Routing	Aggregation Area
Centralized	$NH/2$	NL	
Fully distributed	N^2	L	
Proxy	$NH/2$	$N*L2+L1$	MN

Table 4-1 Communication Cost

During the comparison, we divide the total communication cost into two parts: routing cost and aggregation cost. The aggregation cost denotes the data transmission in the query area and aggregation areas. The routing cost denotes the data transmission between the query area and the Query Sender. The division is based on the fact that, the aggregation cost is related to the specific data aggregation operation, while the routing cost is only responsible for data routing. In other words, the nodes that involve in aggregation cost of communication care about the data aggregation. Their aggregation layer need to process messages related to that specific query. On the contrary, the nodes that involve in routing cost of communication do not care about the data aggregation. The messages are invisible to their aggregation layers.

Centralized strategy needs a lot of routing cost, because response message from each Query Responder needs to be routed to the Query Sender. The load of the routing cost is proportional to the number of Query Responders.

For fully distributed strategy has a large number of aggregation cost. But the fully distributed approach has the minimum routing cost. Only one aggregated message needs to be forwarded from the query area to the Query Sender.

For location based proxy strategy, the communication cost inside the aggregation area belongs to the aggregation cost, because the nodes inside the aggregation area must process the messages. Thus location base proxy strategy splits the aggregation cost in two areas. The amount of messages for routing cost is between the centralized and fully distributed strategies.

From the above comparison, it can be revealed that the overall communication cost relies on the real scenario parameters. Each strategy has its advantages and disadvantages. There must be a tradeoff between different parts of cost.

4.4. Location Based Aggregation Protocol

The above sections give a conceptual study on the aggregation strategies. Through the analysis and mathematical computation, it shows that location based proxy strategy suits well for the high mobility environment. Thus we develop our aggregation protocol based on this strategy. It is called Location Based Aggregation (LBAG). This protocol is responsible for data aggregation for query tasks. The following section describes the possible approaches to make the protocol valid in

high mobility environment and scalable for large-scale network.

4.4.1. Obtain Aggregated data

The objective of the protocol is to coordinate many nodes and get aggregated result from many raw data from individual nodes. In order to get it, following logic entities are defined

Logic Entities and Message Flow

According to location based proxy strategy, two areas are defined: query area and aggregation area. We introduce the third area: sender area, which denotes the area where the Query Sender locates. Nodes in each area have different logic. In order to include them in one protocol, three logic entities are defined accordingly: Query Sender, Query Responder and Aggregator. Query Sender has the logic of a query sender, which sends out the query messages and waits for the aggregated data. Query Responder has the logic of a query responder, which receives the query and sends out response messages. Aggregator has the logic of an aggregator, which collects, aggregates data and sends the aggregated result to the Query Sender. The message flow and the relationships between three logic entities are shown in Figure 4-10

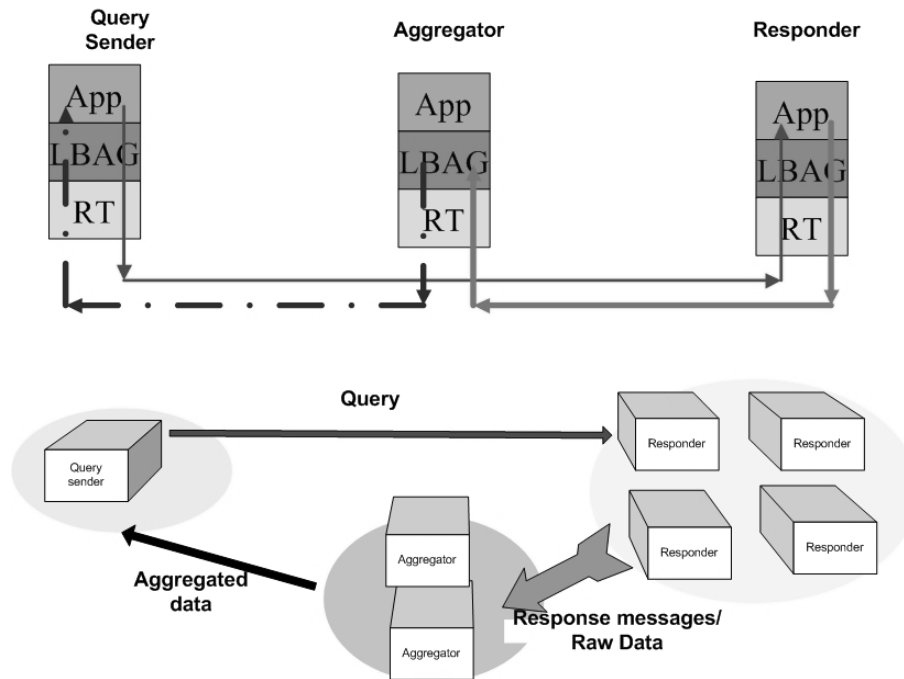


Figure 4-9 Message Flow of LBAG

Each mobile node plays role or roles according to the location based proxy strategy. Query Sender must be played by the Query Sender, which initiates the query operation. Query Responder should be played by all the nodes that send responds messages inside the query area. The role of Aggregator should be played by all nodes inside the aggregation area. A node can play different roles for different query tasks. For example, if node A is located in the query area for query 1, then

it plays the role of Query Responder for query 1. In addition, if it is also located in the aggregation area for query 2, then it plays the role of Aggregator for query 2 simultaneously.

Collect Messages by Aggregator

According to LBAG, Aggregator collects message from Query Responders. Note that Query Responders send response messages to the aggregation area by ad hoc routing. The nodes on the path do not inspect packets data. They only check packet headers to search for routing information. When response messages are sent to the defined aggregation area, all aggregators collect those messages. Each aggregator starts an Aggregation Timer when it receives the first response message. The duration of the timer is T_{AGG} , which is defined by the Query Sender. When the Aggregation Timer expires, the aggregator stops collecting response messages for that query and begins to do aggregation. The aggregation is based on the received raw data that are contained inside response messages. The result of the aggregation should include the value, the context of the data and an indication of completeness.

Define Aggregation Area

The above section describes how the protocol works according to the location based proxy strategy. The most important idea is to define an aggregation area, where nodes can collect raw data and do aggregation. However, we haven't given a concrete solution how to define the aggregation area yet. Location based proxy strategy doesn't give any constraint on the selection of aggregation area, but according to the vehicular network scenario, the aggregation area(s) must be part of road networks, because it constrain the distribution of mobile nodes. The aggregation area can be either defined by the Query Sender or by a default area.

Support Hierarchical Aggregation

When the query area is not an interlinking area or the query area is very large, it's desirable to finish the data aggregation in a hierarchical approach. For the hierarchical approach, the whole aggregation area is divided into several sub-areas. Raw data from sub-area are aggregated first to get partial aggregate. Then the partial aggregates from different sub-areas are combined together to get next level aggregate.

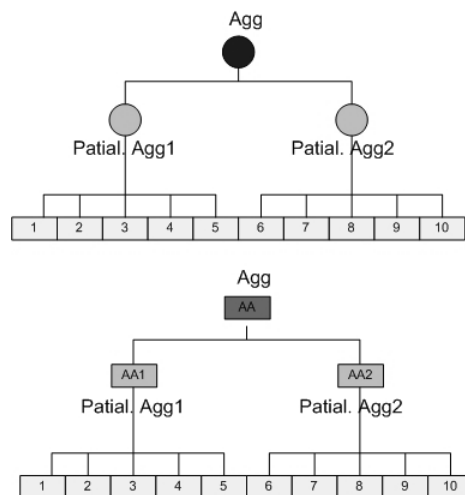


Figure 4-10 Hierarchical Aggregation

Figure 4-11 illustrates a hierarchical aggregation. As shown in the figure, the complete query area is divided into 10 sub-areas, raw data from 1-5 are first aggregated to get partial result 1, and raw data from 6-10 are aggregated to get partial result 2. Those two partial results are combined to final aggregated value. When the location based proxy aggregation is used, then the Query Responders in 1-5 send response message to Aggregation Area 1, Query Responders in 6-10 send message to Area 2. Nodes in Aggregation Area 1 collect message and do aggregation to get partial result1. Similarly, nodes in area 2 get partial result 2. The results are sent to Aggregation Area 3, nodes there will combine them together. From the above illustration, it is noticeable that the aggregation relies on a hierarchical tree of locations. LBAG defines an aggregation hierarchy according to the spatial model. The hierarchy structure embedded in the semantic names provides a good choice for it.

4.4.2. Overcome High Mobility

Support from Location-based Proxy Strategy

A distinguished characteristic of vehicular networks is the high mobility of nodes. High mobility causes the rapid change of network topology and makes it nearly impossible to get a node's exact position. They are the reasons why many aggregation protocols are not applicable to vehicular networks. In order to overcome those two challenges, LBAG uses two methods. The first one is to use location proxy based aggregation, so that the data aggregation does not rely on a tree structure of nodes, but a hierarchy of locations, which is insensitive to the node's mobility. Secondly, LBAG uses Geocast as its routing protocol. Target of routing is now an area instead of individual nodes.

Support from Geocast

We use Geocast as the Ad hoc routing strategy to support LBAG, because Geocast allows a node send message to all the nodes inside a geographic region. Such an operation is necessary for the location based proxy strategy. Furthermore, it is also used to replace unicast, because in vehicular networks, there is no global location service to tell the exact position of the target node. In such a case, a node can geocast message to the area where the target node is possibly located. Since the message is broadcasted inside the area, the target node can get it if it is inside the area. It is obvious that the larger the area is, the higher access rate will be. But the larger the area is, the more communication overhead will be. Thus there is a compromise to select the range of the area. Considering our system model and project objective, the Geocast protocol should be spatial information aware. That means it should support the representation of destination area by geometric definitions, semantic names or both. Here we give the basic requirements of Geocast, the details and enhancements in the protocol are illustrated in section5.2.

4.4.3. Reduce Communication Cost

Wireless links in vehicular networks are bandwidth limited. When the range of radio bandwidth and the modulation scheme are fixed, there is no ways to increase link capacity. Therefore in order to make the protocol scalable, overall communication for a single query task must be reduced.

Reduce by Aggregation Strategy

In the model of location-based proxy that we studied before, every response message is sent to the

aggregation area. The result shows that this strategy needs less communication cost than the fully distributed strategy. However, fully distributed aggregation also has its advantages. It allows every node inside the query area get information from other Query Responders. Therefore, we use a combination of both strategies to enhance the performance of our LBAG protocol. LBAG uses a limited fully distribution. It means that the complete query area is divided into several sub-areas. In each sub-area, the response messages are distributed to other nodes. Thus the complete aggregation is divided into two phases. The first phase is limited distribution of raw data; the second is hierarchical aggregation of partial aggregates. The partial aggregates from the first phase are sent to the aggregation area, where higher level of aggregation is performed. Through such a strategy, the distribution of raw data is limited within a certain range. The mathematic model then is changed into what Figure 4-13 depicts.

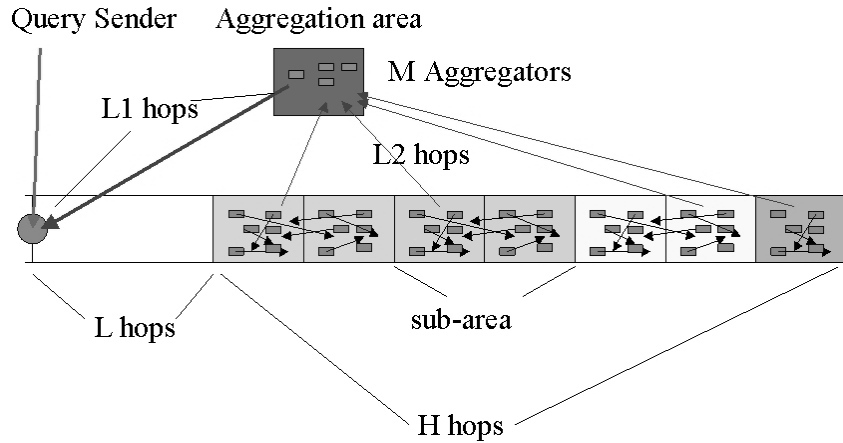


Figure 4 - 11 Location Based Aggregation

In Figure 4-13, the partial result from sub-areas are gathered at an aggregation area of size A hops, M nodes. If there are G sub-areas, then the total communication cost is:

The communication cost for each sub-area $C_{sub-area}$ is:

$$C_{sub-area} = \left(\frac{N}{G}\right)^2 \quad (G = \text{number of sub-areas}); \quad C_{sum} = G \cdot C_{sub-area} = \frac{N^2}{G}$$

The messages needed to forwarded partial results is: $C_{Fpartialagg} = G \cdot L_2$

The cost inside aggregation area is $C_{agg} = G \cdot M = GM$

The forwarding of total aggregation to the Query Sender is L_1 .

	Inside query area	Routing	Aggregation Area
Centralized	$N^*H/2$	NL	
Fully distributed	$N^2+H/2$	L	
Proxy	$N^*H/2$	$N^*L_2+L_1$	N^*M
LBAG	$N^2/G + G^*H/2$	$G^*L_2+L_1$	G^*M

Table 4-2Communication Cost of LBAG

It could be noticed that, LBAG reduces the communication in the aggregation area and in the

network, but increases inside query area.

In our system, many elements are location dependent: including the raw data, the query area and the property of aggregates. When the query is about the network property, then the minimal unit is an edge in the graph, a segment in the digital map. Thus it's reasonable to choose road segment as the sub-area unit.

To use the road segment as unit of hierarchy of locations has following advantages. 1) The road segment matches the edge of geographic topology graph. 2) It is the minimal unit of a semantic hierarchy tree. Therefore, it acts as a bridge between the geographic graph and the logic semantic hierarchy. The queries from applications are normally based on logic model, while the routing packets require the geographic graph. The use of segment as a unit gives a convenient mapping between two models.

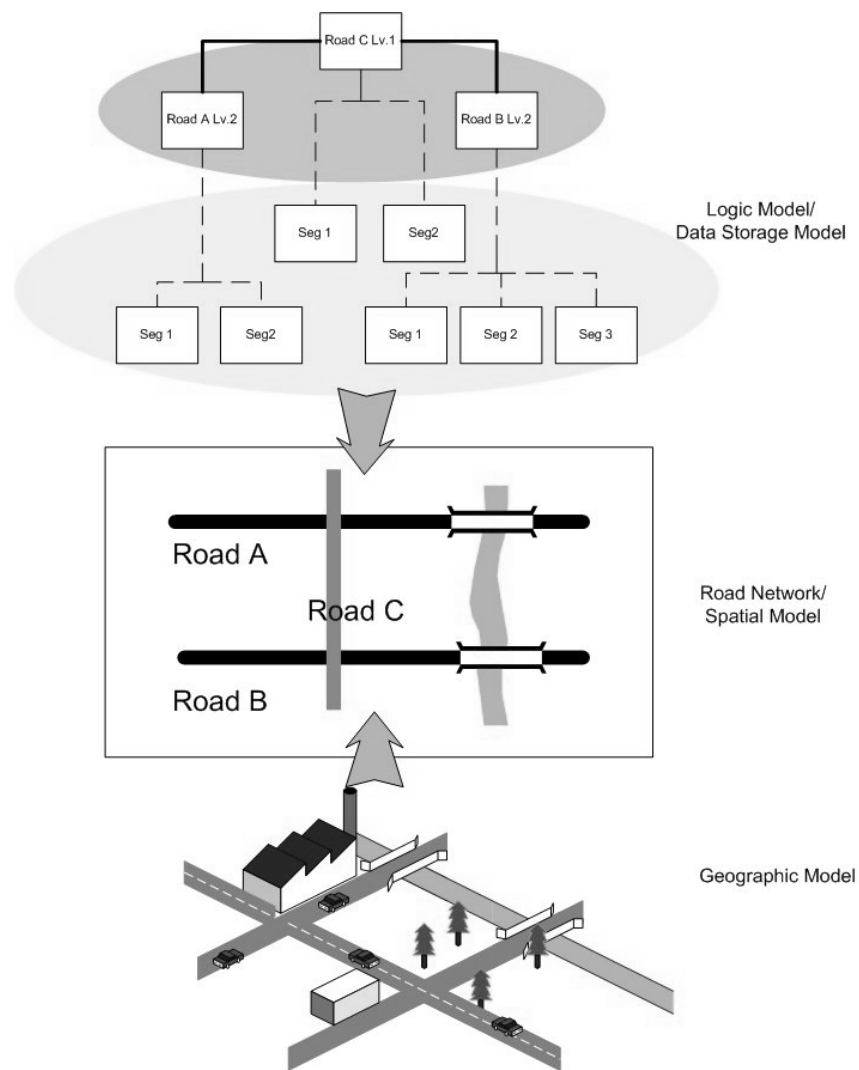


Figure 4-12 Models Mapping

With this idea, the behavior of LBAG can be illustrated as below:

- 1) The query area is divided into several sub-areas. Each sub-area corresponds to a road segment. Fully distributed aggregation is done in each sub-area, namely on each road segment. The flooding of response messages is limited in the local segment.
- 2) After the Query Responders have exchanged data within their segment, all nodes aggregate the collected information on their own node. One of the aggregates is sent to the higher level aggregation area, where nodes in that area collect the partial aggregates from many segments and do further aggregation.
- 3) The aggregate result is sent back to the Query Sender or higher aggregation area if exists.

Suppress Duplicated Aggregates

According to LBAG algorithm, there might be often more than one node inside the aggregation area. When every aggregator collects messages and does aggregation after certain period of waiting time, then more than one aggregated message will be sent out from the aggregation area. Obviously there are many redundant messages. This phenomenon will be worse if there are many nodes inside the aggregation area. Hence a method is proposed to suppress the duplicated aggregates.

Each aggregator set the Aggregation Timer with a random time, so that many timers of aggregators won't expire simultaneously. When an Aggregation Timer expires, the aggregator sends out its aggregated message to the Query Sender or its higher-level aggregation area. Then it geocasts a SUPPRESSION message in the aggregation area to inform the other aggregator about its existence and its aggregated result. When a node of other aggregators hears the SUPPRESSION message, it will stop their aggregation timer and won't send their aggregates. Thus the duplicated aggregation messages are suppressed. The SUPPRESSION message includes the query ID, aggregated value, and time stamps.

However, it is not guaranteed that one and only one aggregated data is sent out from the aggregation area. Because Geocast of a message also takes time, especially when the area is larger than a radio hop. When the timer of two non-neighboring aggregator expire close enough, none of them can suppress the other. Then two aggregated result might be both sent from the same aggregation area. But if it happens, the duplication is not suppressed further. We notice that the number of such duplicates are limited and will be useful to increase the success arrival rate or can be used to check the completeness of aggregates.

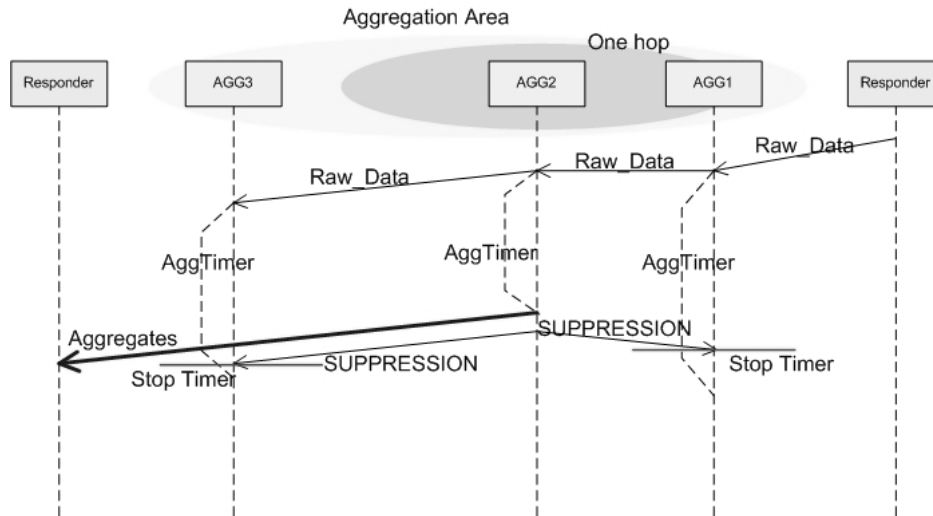


Figure 4-13 AggTimer Sequence Chart

Data Storage

To prepare for the aggregation, each aggregator collects response messages. The aggregator stores the data value together with its context, which includes a unique query ID, which is assigned to each query when the message is initiated. All the messages that relates to that querying task must carry the corresponding ID. Query ID is also used as the index of accessing query related information.

4.4.4. Scalability

According to LBAG, all nodes related to a query task can be classified into two categories. The first category is the nodes inside query related areas, which includes the query area, the aggregation area and the sender area. The data aggregation layers of those nodes process the LBAG protocol messages. The other category is the nodes that are not inside above areas. The aggregation layers of those nodes do not need to process the protocol messages. Based on this classification, the overall communication cost can be divided into two parts: aggregation cost and routing cost. Such a division has important meanings to the scalability of the network.

In vehicular networks, multiple tasks are allowed to coexist in the whole area. However, their routing path might overlap at some road segments. The communication traffics in the overlapping areas are accumulated. When the accumulated traffic load is close to or greater than the limited bandwidth of wireless link, large number of packets will be lost. Thus a bottleneck occurs. In order to prevent from such a situation happening in the network, not only the reduction of communication cost is mandatory, the balance of traffic load is also necessary. A bottleneck example is shown in Figure 4-15. Several query tasks are active in the network. All those routing paths include the central part of the network, due to the constraint of road network. In such a case, the number of routing cost must be controlled to prevent from creating a bottleneck in the middle. The amount of aggregation cost becomes unimportant. Of course there are examples emphasis on the aggregation cost instead of the routing cost. So generally, a scalable protocol must be able to

flexibly distribute the traffic load according to the real situation of the scenario.

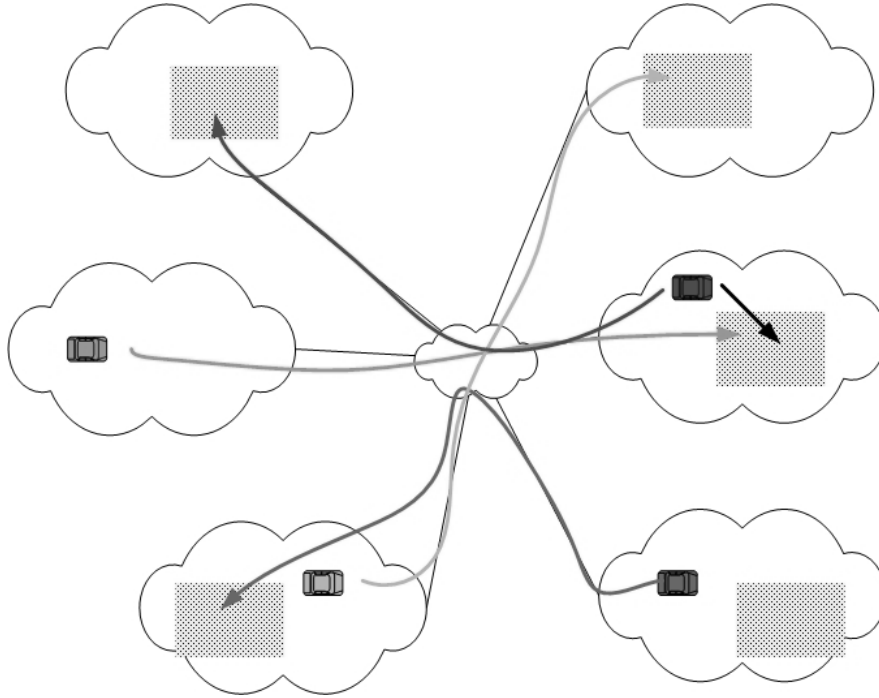


Figure 4-14 Possible Bottleneck in the network

LBAG has this capability, through the flexible definition of Aggregation areas. The details of it will be described in section 5.4.4.

4.4.5. Performance Enhancement

The above sections presents the working behaviors of LBAG and how to solve the challenge from high mobility and scalability. In this section, we further improve the conception to enhance the performance of LBAG in data aggregation.

Compare Completeness at Suppression

If there is no link failure, data collisions or network partitions between the query area and aggregation area, all response messages should be received by every aggregator. However, the reality is opposite to the above assumption. It is highly probable that the number of raw data received by aggregators is not the same, which means the aggregates will be inconsistent to each other. In such a case, the Query Sender hopes to get the best aggregated data. To achieve this, additional mechanism is needed to extend the basic version of LBAG.

As mentioned in 4.3.2, SUPPRESSION message is used to stop the aggregation timer on other aggregators. We can extend the SUPPRESSION message by appending aggregated data and an indication of data completeness in it. Thus every aggregator receives the SUPPRESSION message can compare it with its own result. Note: an aggregated message contains completeness indication,

but the aggregated message is routed to the sender area. The neighboring nodes can not received the aggregated message. If its own completeness is higher than that of the received aggregate message, this node can send its aggregated data to the Query Sender and geocasts a new SUPPRESSION message to inform the others about its result. In order to prevent from large number of nodes sends out their objections to the first aggregated data, a random back off timer is needed.

Introduce Limited Redundancy

With the above correction of an aggregated result and the redundant of aggregated messages mentioned in section 4.43, there might be more than one aggregated message arrive the Query Sender. Such kind of limited redundancy can actually help to increase the arrival rate and credibility of aggregated result, therefore, they are not suppressed. In addition, in cases that the Query Sender needs a result with higher credibility; certain degree of duplicated messages can be introduced on purpose.

1.1.1.6. Data Handover

Another challenge for the completeness of the aggregated data comes from the high mobility of nodes. If during the collecting period of a data aggregation, some nodes might leave the aggregation area and some might come in. Thus, it's possible that none of the aggregator has all the raw data, either miss the beginning or miss the end. Then it is expected that the aggregators that leave the aggregation area can pass their knowledge to the new aggregators.

We proposed a data handover scheme, which takes consideration of geographic semantic information. It works as follows:

When a node enters a new segment, it will be ready to be the aggregator for this segment. When it receives a piece of raw data, it will store the data according to the query ID. When it leaves the segment, it checks if there are any active aggregation, if yes, then it will do aggregation and sends out its aggregates to the leaving segment. All nodes inside that segment can hear the aggregate and update their own data list if necessary. The handover message must contain query ID, aggregated value, completeness, starting and ending time of the collection. However, the handover of knowledge needs a signal from spatial mode to notify node the changing of segment.

5. Design

In chapter 4 we completed the conceptual study of data aggregation strategies. A protocol called Location Based Aggregation is proposed for vehicular networks. As shown in the software architecture of a node (Figure 5-1), LBAG uses underlying Geocast to route packets, obtains spatial information from Spatial Model and provides aggregation service to the upper Application layer. In this chapter, the LBAG protocol is formalized. But before that, all interfaces to the neighboring modules must be defined.

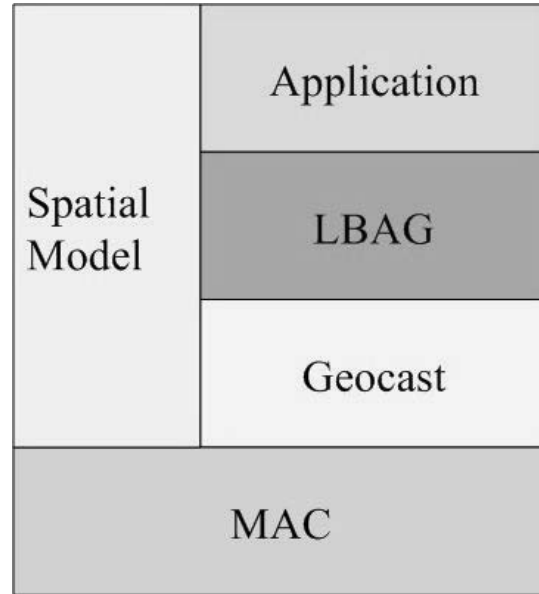


Figure 5-1 Software architecture

5.1. Spatial Model

Spatial model is a key element in the software architecture. It is responsible for the representation and management of geographic and semantic information.

5.1.1. Interface

A common interface is provided by spatial model to other modules. The interface consists of three sets of functions: 1) access functions 2) mapping functions 3) computation functions.

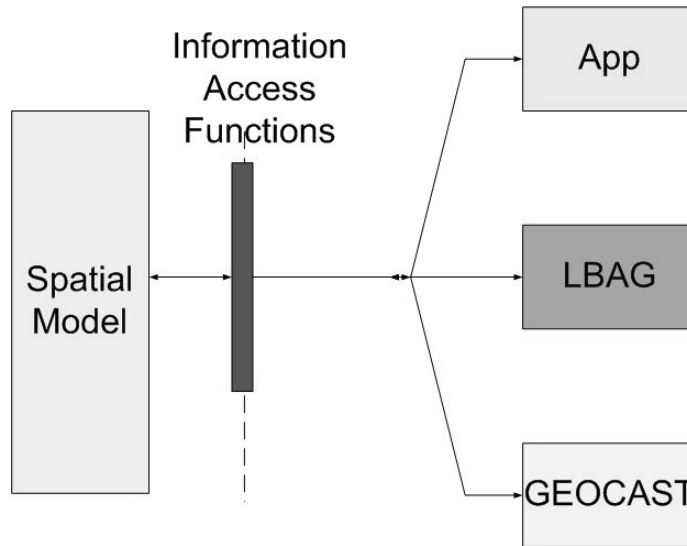


Figure 5-2 Common interface of Spatial Model

Access Functions: Those functions return the geographic or semantic information of the current position.

- Access the coordinate of the current position
- Access the semantic information of the current position

Mapping Functions: Those functions return the geographic or semantic information of a given position or semantic name.

- Map a position to its semantic names
- Map a semantic name to its positions

Computation Functions: Those functions do mathematical or logic computation on the geographic or semantic information.

- Get the distance between two positions
- Get the shortest path between two points
- Get the closest entrance point to the given area
- Test whether a position is inside the given area

5.1.2. Representation of spatial information

How a spatial model is implemented is not relevant to the development of rest modules, as long as the functions of the common interface are implemented correctly. However, semantic information must be included in our spatial model. Therefore, a hierarchical model is adopted in our development. In order to distinguish it from the other spatial models, Spatial Model is used to refer to our implementation.

In the Spatial Model, there are two parts of information: geographic and semantic information. The geographic information includes the road network topology and location coordinates. The semantic information includes 1) semantic name of geographic objects, such as roads and crossings 2) hierarchy of semantic information, such like a road-section-segment hierarchy. Those

two parts of information needs reciprocal mapping.

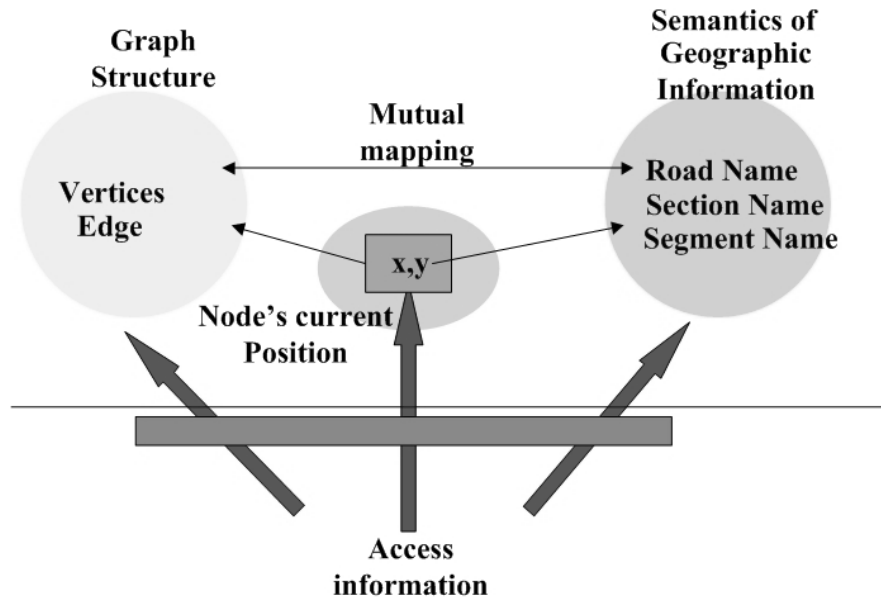


Figure 5-3 Spatial Model

Figure 5-3 outlines the interface and structure of the Spatial Model. The details of the implementation are given in section 6.2.

5.2. GEOCAST

LBAG relies on the underlying routing protocol for message exchange between the nodes. In addition, an efficient implementation of routing service will enhance the overall performance of data aggregation. However, none of the proposed geocast protocols in the related work Section 3.1.1 is suitable for our system model. Hence a new geocast protocol is developed, which uses position based routing and optimized flooding. We name it GEOCAST to distinguish this new protocol from the other geocast protocols.

5.2.1. API

The API of GEOCAST defines the service that the routing protocol provides to the upper layer. The service is to route a given message to every node inside the given destination area. Similar to other routing protocols, the API of GEOAST includes two functions: **SEND** <message, destination area> and **DELIVER** <message>.

According to this API, an upper layer passes its protocol message to GOECAST. Upon receiving a message from the upper layer, GEOCAST encapsulates it in a GEOCAST data packet and route it to the destination area. When a node inside the destination area receives a GEOCAST data packet, it rebroadcasts the message and delivers the message data to its upper layer.

Protocol Message Format

There are three types of GEOCAST protocol messages: 1) DATA BROADCAST 2) DATA FORWARD 3) BEACON. The first two are called GEOCAST data packets, which have a header and a data body, while the BEACON message only has a header.

GEOCAST header includes following fields

Fields	Description
Destination area	Geocasting Region
GEOCAST Message Type	GEOCAST message type identifier. DATA FORWARD/DATA BROADCAST/BEACON
Target Position of Forwarding	Position of reference forwarding point
Position of Origination	Position of packet origination

Table 5-1 GEOCAST Header Fields

Destination area is a very important part of GEOCAST header, which corresponds to the destination address in unicast packet headers. There are many options for the representation of a destination area. 1) A rectangular can be used to define a destination area by two coordinates of the diagonal $[x1, y1, x2, y2]$. 2) A circle can be used to define the destination area by the centre and the radius $[x1, y1, r]$. When the radius is zero, then the circle shrinks to a point. 3) Apart from using geometric shapes to define destination area, one semantic name or a list of semantic names are also allowed to define the destination area. Spatial Model on each node is responsible for the interpretation of semantic names to real positions. With those three options, we can define the destination area in a precise and neat way.

Rest of the header fields such as the target position of forwarding and the position of origination will be discussed later in the protocol operations.

5.2.2. Protocol Operations

The objective of GEOAST is to route a packet to all the nodes in a given area. The entire routing process includes actually two phases: 1) forwarding a packet from the sender to the destination area 2) broadcast a packet inside the destination area.

Each protocol message records the state of the routing phase in its header's field: *GEOCAST Message Type*. The switching from forwarding phase to broadcast phase depends on the location relationship between a message's current position and the destination area. A test function called *isInsideDestArea()*, which is defined in the Spatial Model's common interface, is called to test that relationship. *isInsideDestArea()* requires the destination area as the input parameter, and returns a Boolean result. The complete routing algorithm is depicted in Figure 5-4.

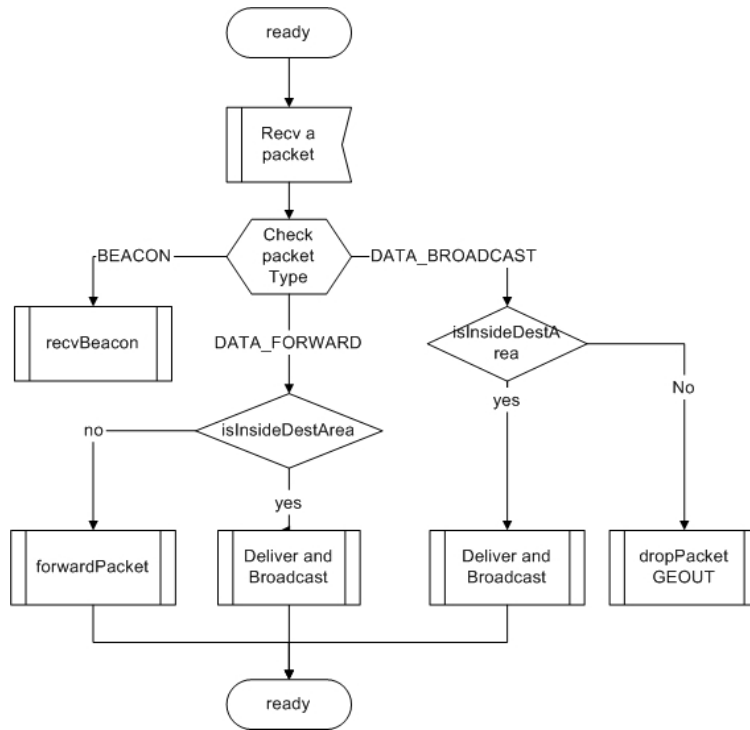


Figure 5-4 GEOCAST Algorithm

Upon receiving a message, either from upper layer or lower layer, GEOCAST first checks the message type. If the type is DATA FORWARD, this message is in forwarding phase. If it is DATA BROADCAST type, this message is in broadcast phase. After the check of message type, test function *isInsideDestArea()* is called to see whether phase switching from forward to broadcast is needed. Function *deliver()* and *broadcast()* is called when the position of the message is inside the destination area, otherwise function *forward()* is called. Those procedures are illustrated below.

1.1.1.7. Function deliver and broadcast

When a GEOCAST data message arrives a node that is inside the destination area, it will be duplicated once. The duplicated message will be delivered to the upper layer by *deliver()* and the original message will be broadcasted by *broadcast()*.

Function *broadcast()* is used to do local broadcast of the received message. It sets the MAC_BROADCAST as the MAC address, so that all neighboring nodes receive the message.

When plain flooding is used as the broadcast algorithm, all nodes that hear the message for the first time will rebroadcast it once again. If a node receives a GEOCAST data message that is already known, it drops it. The plain flooding ensures the delivery of a packet to each node that currently is located in the destination area, if no network partition exists. However, it introduces large number of messages to the network. The number of rebroadcast messages is proportional to the square of node number. Therefore, we propose a better algorithm to reduce the rebroadcast messages.

Figure 5-5 shows a typical scenario of vehicular networks. Nodes are distributed along the roads. When node1 broadcasts a message, all neighboring nodes within its radio range can hear the message.

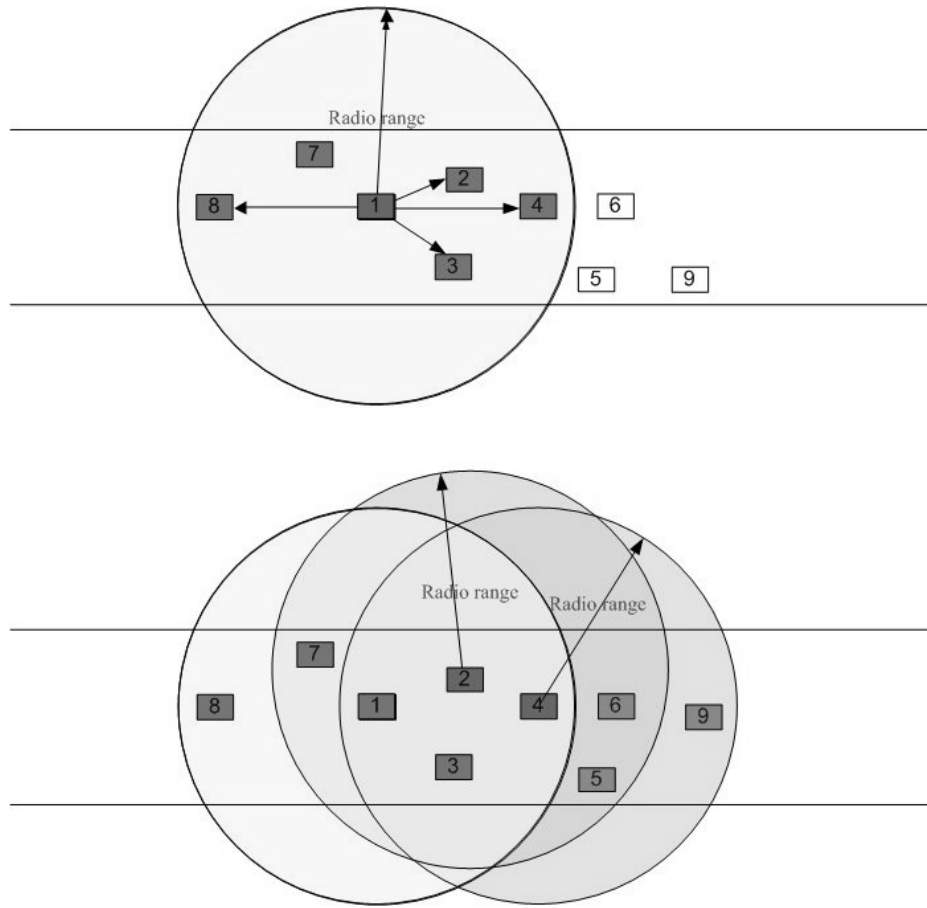


Figure 5-5 Broadcast

According to plain flooding algorithm, node 2,3,4,7,8 will rebroadcast the message after they receive it, leading to many duplications and collisions on the MAC layer. But we noticed that the neighboring nodes of node 2 are also neighbors of node 1 and node 4, because of the linear distribution. That means, the rebroadcast at node 1 and 4 give the message to all the neighbors of node 2. Hence node 2 does not need to do rebroadcast. In order to suppress the in-between nodes from rebroadcasting, we use the following mechanism to let the farthest neighbor rebroadcasts the message first. When node1 rebroadcast a message, all its neighbors 2,3,4,7,8 will hear the message. Before doing rebroadcast, they will wait a period of time T_x .

$$T_x = \frac{k}{Dist(x, node1)}$$

The x in the equation denotes the ID of the node. k denotes rebroadcast modifier coefficient.

That means the delay is reversely proportional to the distance between the node and the sender. The farther a neighbor is, the shorter it needs to wait. When the delay timer expires, it will

rebroadcast the message to its neighbors. In the above scenario, node 4 and node 8 are expected to experience a shorter waiting time than node 2,3,7. When nodes 2,3,7 hear the rebroadcast from 4 or 8, they will stop their rebroadcast timer and cancel the rebroadcast.

1.1.1.8. Function Forward

Unlike *broadcast()*, GEOCAST uses unicast to forward a packet to its destination area. The forwarding algorithm uses the position based routing, which is discussed in section 3.1.2. Position based routing makes the forwarding decision based on three parts of spatial information 1) the node's current position 2) the destination area 3) the location of the neighboring nodes. The node's current position can be obtained from Spatial Model, and the destination area is already available in each GEOCAST data packet header. The location of the neighboring nodes can be obtained by BEACON messages (illustrated in next section). Thus a node can make stateless forwarding decision based on local spatial information.

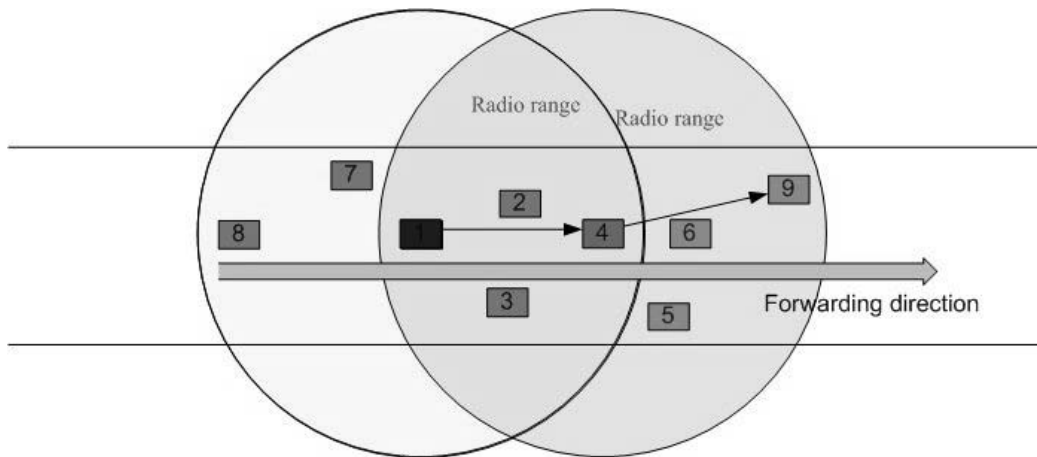


Figure 5-6 Forwarding a Packet

When a message is not inside the destination area, function *forward()* is called to process it. It first tests if the node has a neighbor locating inside the destination area. If yes, it means this message is already close to the destination area. This node can broadcast it. Thus all neighbors inside the destination area get the message and start to rebroadcast it. If none of its neighbor is inside the destination area, the node starts to find a neighbor with the shortest distance to the destination area. However, the target of a packet is now an area instead of a position. A proper reference point is needed to enable the calculation of Euclidian distance. There are many options to select the reference point. For example: 1) the geometric middle of the area 2) the crossing on the border of the area that has the shortest Euclidian distance to the current position 3) the crossing on the border to the area, with which the weight of the shortest path is minimum. We take the second option as the default choice because computing Euclidian distance is less time-consuming than computing the shortest path. Then the reference point will be the entry point for the message to the destination area. A function in the common interface *findTargetIS()* is called to do such reference point searching.

Such a forwarding mechanism belongs to geographic greedy forwarding. Apart from it, there are enhanced position based forwarding implantations like GPSR and SAR. Since LBAG does not

rely on any specific implementation of GEOCAST, their forwarding techniques can be used to improve Geocast protocol. However, to improve the performance of geocast routing is beyond the scope of this work. In this thesis, a simple greedy forwarding is used for GEOCAST.

1.1.1.9. Beacon and Neighbor Table

Beacons are used to inform the neighboring nodes about a node's position. Every node sends beacon periodically. The period of beaconing is $T_b = T_{b0} + T_{jitter}$. To add a jitter to the beacon interval is to avoid the collision of beacons of different nodes on the MAC layer.

Each node has a data structure called *Neighbor Table*. When a node receives a beacon from another node, it stores the location information and the node's ID in the *Neighbor Table*. *Neighbor Table* contains many entries, each of which corresponds to a neighboring node record. The valid duration of each entry is controlled by a related timer. Upon receiving a beacon from a new neighboring node, a new entry is added to the *Neighbor Table*. A beacon from a known neighbor refreshes the corresponding entry for this timer. An entry will be removed from *Neighbor Table* after the expiration of the entry timer, which means that the node is not its neighbor any more.

With a short beacon interval, a node can let other nodes find it more quickly and more precisely. With a shorter entry valid time, a node can forget a node more quickly when it is not reachable anymore. The value of the beacon interval and valid time depends on the mobility of nodes. The shorter the beacon interval and the entry valid time is, the more precisely the *Neighbor Table* can reflect the current neighborhood situation. But this also costs more communication overhead. In order to reduce the occupied bandwidth, implicit beaconing is used, which piggybacks beaconing messages in every normal GEOCAST data packet. When all nodes work in promiscuous mode, every message sent out can act as a beacon message. Thus node can exchange location information without blocking the link by large amount of beacons.

1.1.1.10. Constrained Greedy Forwarding

As mentioned in section 5.2.2.2, greedy forwarding is used in *forward()*. The neighboring node that is most close to the destination is selected as the next hop node. However, this approach has problem in the high mobility scenarios. The problem is caused by outdated information in the *Neighbor Tables*. Note that a node records a neighbor's position when it receives a beacon. The position of the neighbor is not updated until a refresh beacon comes. During the interval, the node assumes its neighbors are static. If the node's movement speed is slow, then if the beacon interval is 0.5sec, the change of location of neighbors is very limited. But when the movement speed is high, for example 180kmh, then a neighbor can move 50m with 0.5sec. It's highly possible that a node hears a beacon from another node but that node soon leave out its radio range within 0.5sec. Figure 4-7 shows a scenario, in which node1 tries to forward a packet greedily. By checking its *Neighbor Table*, the node finds node4 is most close to the destination. Therefore, it set the MAC address of node4 as the receiver of the packet. However, the node4 has already leave the radio coverage range, and thus unable to receive the packet. The packet becomes lost.

Several approaches are taken to solve this problem. First beacon interval is shortened, so that a node can get more fresh position reports. Secondly, the valid time for a neighbor entry is shortened. An entry becomes obsolete when no refresh beacon is obtained. Third, a constraint is set in selecting next hop node in greedy forwarding. The next hop nominee should not be very close to the border of its radio coverage, because they are possibly unreachable. Then the algorithm of greedy forwarding becomes: find a node in the *Neighbor Table*, which is closest to the destination and our interval distance is no larger than (Radio Range – Guard Distance). The value of *Guard Distance* can be set as: $Guard\ Distance = T_{beacon\ interval} * Max\ Velocity\ Limit$

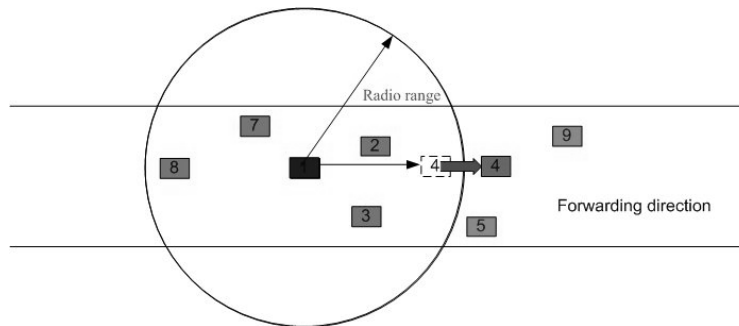


Figure 5-7 Message Lost due to outdated Neighbor Information

5.2.3. Summary of GEOCAST

A Geocast routing protocol GEOCAST is developed. GEOAST uses spatial information and supports geocast in the destination area. It provides an API to the upper layer, which includes a function: **send<message, destination area>**. LBAG uses this API to forward its protocol messages. An efficient implementation of Geocast will enhance the overall performance of data aggregation.

5.3. Query Application

We have illustrated the development of Spatial Model and GEOCAST routing protocol. In this section, module: Query application is defined before we start to develop LBAG. It would be easier to explain the contents of LBAG messages and the LBAG protocol operations with the clearly defined application data (query messages).

Query application only concerns about the application data and semantics of spatial information. The Query application is responsible for defining the query and generating response data. Query application calls LBAG to process the query. LBAG collects the data according to the specifications in the query and computes the aggregated result. Next section describes how a query is defined and how a response is generated according to a received query.

5.3.1. Format of Query and Response

Applications of vehicular networks can send queries to ask for network properties or to initiate cooperative tasks. The definition of a query includes the query command and related spatial areas: query area and aggregation areas. Aggregation areas can be defined by application or by LBAG. The definition from application overloads the default definition from LBAG, so that the application is able to determine aggregation area with the help of other application contexts.

The definition of query command includes the data type, data name, query operation and query area. The query operation is a set of functions that is a subset of SQL commands, so that all kinds of applications can use the data aggregation services in a general way. The operations we support in our simulation are of five types: MIN, MAX, SUM, COUNT and AVG.

Fields	Description
Query Type	Snapshot or continuous query
Query area	Query area
Aggregation Area	Aggregation Area
Data Name	
Data Type	
Query Operation	MAX/MIN/SUM/AVG/COUNT

Table 5-2 Format of Query

Fields	Description
Data value	Value of a Response

Table 5-3 Format of Response

5.3.2. Message Flow of a Query Operations

The message sequence chart of a query operation illustrates how a query operation is processed. (Figure 5-8)

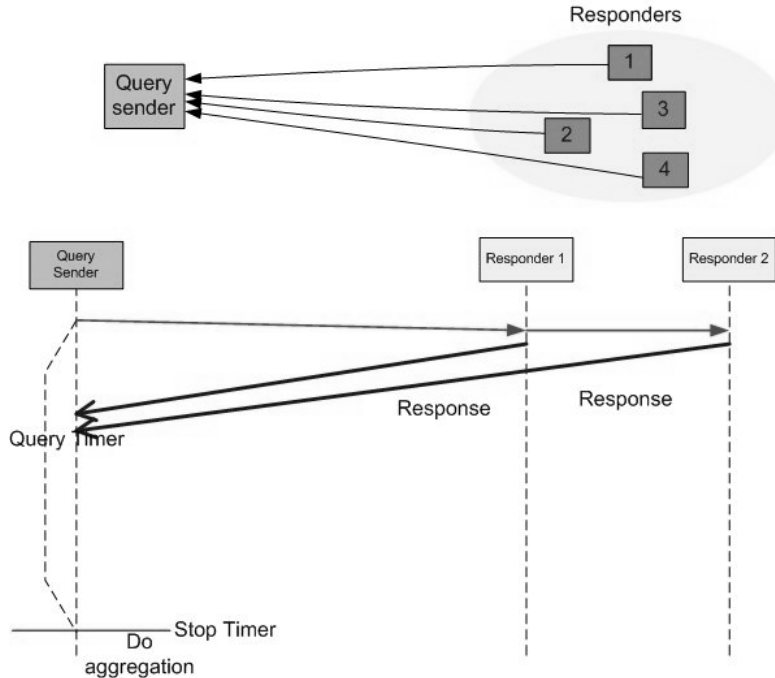


Figure 5-8 Message Sequence Chart of Query Application

5.3.3. Operations

Query Application is responsible for initiating a query, resending a query and receiving results. Those operations are performed by the following functions respectively.

- **Send a Query:** Query application creates a query message and calls underlying layer to send the query to the query area.
- **Respond a Query:** Each node inside the query area generates raw data and encapsulates it in response message.
- **Send Response:** Response message is routed by underlying protocols to the aggregation area.
- **Receive Result:** Query application retrieves the query result from its underlying LBAG protocol.

5.3.4. Time control

For each query, the Query Sender creates a Query Timer. It sets up the timer when it sends out a query message. When the timer expires, the Query Sender retrieve aggregated data from underlying LBAG protocol.

5.4. Location Based Aggregation

5.4.1. Protocol Overview

According to the software structure of a node, LBAG provides aggregation services to applications. LBAG defines three logic entities: 1) Query Sender, 2) Query Responder and 3) Aggregator. Each logic entity plays a special role in the algorithm. Query sender is responsible for sending the query and receiving aggregated result. Query Responder receives a query and sends out its response message. Aggregator collects response messages and does aggregation.

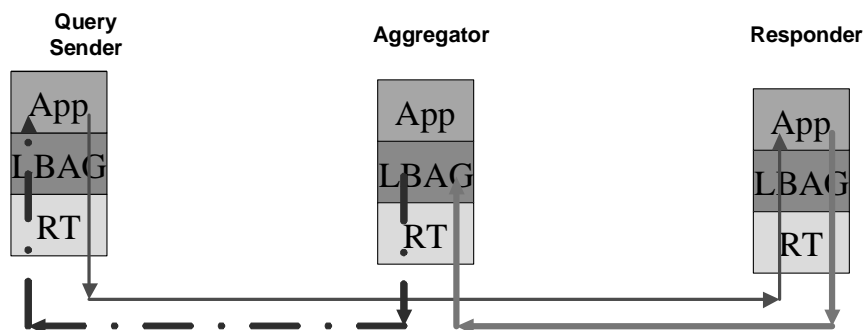


Figure 5-9 LBAG Message Flow

5.4.2. Protocol Messages Format

There are five kinds of protocol messages defined in LBAG. 1) QUERY MESSAGE, 2) RAW DATA, 3) AGG DATA, 4) SUPPRESSION. Each protocol message consists a protocol header and message body. The fields of LBAG header are shown in Table 5-4:

Fields	Description
LBAG Message Type	Identify the content of data body
Query ID	Distinguish message of different querying tasks
Expire Time	Valid duration of the querying task
Generation Time	Distinguish duplications if existing

Table 5-4 LBAG Message Header

The LBAG Message Type inside the query header indicates the type of messages.

Format of Query Message:

	Fields	Description
Aggregation Control Flags	Query Type	Snapshot or continuous query
	Query area	Query area
	Aggregation Area	Aggregation Area
Query Definition	Data Name	
	Data Type	
	Query Operation	MAX/MIN/SUM/AVG/COUNT

Table 5-5 QUERY MESSAGE

Format of Raw Data:

	Fields	Description

	Query Message	Context of the raw data
Raw Data	Raw data value	Value of the raw data

Table 5-6 Format of Raw Data

A complete query message is included in the RAW DATA message. Thus a node doesn't know about the QUERY MESSAGE can get the context of the raw data.

Format of AGG DATA:

Fields	Description
Query Message	Context of the raw data
Agg data value	Value of the aggregate
Coverage	The description of the source of aggregation

Table 5-7 Format of Agg Data

Similar to RAW DATA, AGG DATA also includes a complete query message to provide the context of an aggregated data. In addition, a description of the group, on which the aggregation operation is done, is given.

Format of SUPPRESSION:

Fields	Description
Query ID	Distinguish message of different querying tasks
Agg data value	Value of the aggregate
Completeness	An indication of the completeness of the aggregate

Table 5-8 Format of SUPPRESSION

The use of SUPPRESSION message will be illustrated in next section.

5.4.3. Protocol Operations

1.1.1.11. Query Sender Operation

Query Sender is responsible for processing new queries from upper layer and AGGREGATED DATA from low layer. It is a query's initiator.

Arrival of a new Query from upper layer

When a new query arrives, the Query Sender should assign a unique query ID for the query. The query ID will be used in all the messages that relate to this querying task. Query sender must record all the important information of a query, such as query ID, query area, and the fields of Query message. In addition, query sender starts a Query Timer, which will expire after the lifetime of a query. A buffer is created to store the incoming aggregates. When the recording is done, query sender will create a QUERY MESSAGE based on the new query, and call GEOCAST's API function send<Query Message, query area> to send out the query. Then query sender is free again and ready for the next query.

Arrival of an AGG DATA

According to the logic entity model, query sender only process aggregated data. So only an incoming aggregated data will trigger the Query Sender once again.

When an AGG DATA arrives, query sender will check whether it is the real query sender (not an aggregated data of another querying task). If the query ID is known, it will extract the aggregated data from the message and store it as the final result of the query.

Expiration of Query Timer

When a query timer expires, aggregated data should have already arrived the Query Sender. The Query Sender will check whether the final result is available. If it exists, it delivers the result to its upper layer. If not, it informs its upper layer with an error exception.

1.1.1.12. Query Responder Operation

Query Responder is responsible for processing a QUERY MESSAGE from the lower layer and a query response from the higher layer.

Arrival of a QUERY MESSAGE

When a new query message arrives from the lower layer, the Query Responder records the query's ID and other fields. Space for result buffer is assigned, but no timer is set up. Then it extracts the query from QUERY MESSAGE and sends it to upper layer: Application. Application analyses the query and generates a response.

Arrival of a Response

When a response from upper layer arrives, the Query Responder extracts the data value from the response and records it to the local result buffer. Then a new RAW DATA is created to encapsulate the response. The query ID of the response message is the same as the query ID of the query. Finally, the Query Responder calls GEOCAST's API function `send<RAW DATA, aggregation area>` to send out the raw data to aggregation area. Then Query Responder is free again and ready for the next message.

1.1.1.13. Aggregator Operation

Aggregator only receives protocol message from lower layers, because it is only visible in aggregation layer. It is responsible for processing a RAW DATA, SUPPRESSION messages.

Arrival of a RAW DATA

When a raw data arrives, aggregator will check whether it is from a new querying task. If the query ID is unknown, then this RAW DATA is the first one for this querying task. The aggregator will record the fields of this query and assign spaces for result buffer. An AggTimer will be started to count the time for collecting RAW DATA messages. Then the RAW DATA will be dropped. If the arriving RAW DATA is not the first one for this querying task, the aggregator only needs to store the

data value and the Query Responder's context into the result buffer. If the AggTimer is already started, it will not reset it. After those actions, the aggregator is free again and ready for the next RAW DATA.

Expiration of AggTimer

When an AggTimer expires, the aggregator stops collecting raw data and starts to do aggregation of the data in the result buffer. When the aggregate and coverage is computed, a new AGG DATA is created. The aggregator will call GEOCAST's API function send<AGG DATA, Sender Location> to send out the message. The sender location is from the information in QUERY MESSAGE, which is included in RAW DATA. After finishing those actions, the aggregator will mark the state of the querying task as finished. Finally, the aggregator must form a SUPPRESSION message to reduce the possible duplicated aggregates from its neighboring aggregators. Then the aggregator is free again.

Arrival of SUPPRESSION

When an aggregator receives a SUPPRESSION message, it checks the querying task it belongs to. If its own AggTimer for that querying task is still pending, i.e. it is going to expire, it stops the AggTimer to prevent from generating a duplicated AggData. When completeness enhancement is used, the aggregator stops its timer and does aggregation right away. It then compares its aggregate value and completeness with the received result. If its own result is much better, it sends out its own AGG DATA after a random waiting (waiting to hear corrections from other aggregators), no matter whether there is already one being sent to the Query Sender. Such duplication can increase the completeness of the final result at the Query Sender.

When an aggregator receives a SUPPRESSION, if the AggTimer is inactive and the random Timer is set, it compare the completeness of this version of aggregated data with its own aggregation result. When the received result is better, it cancels its random Timer, otherwise it sends out its aggregated data and SUPPRESSION message.

1.1.1.14. Mapping physical nodes to logic entities

Now we have illustrated the behavior of logic entities. The mapping of real nodes to logic entities is simple. The node, which initiates a querying task, plays the role of Query Sender. If a node receives a QUERY MESSAGE, it plays the role of Query Responder. If a node receives a RAW DATA, it plays the role of aggregator. Note that different logic entities must not locate on different real nodes. A node can play several roles for a querying task simultaneously. Each node records the role(s) that it plays for each querying task. Thus a node can play different role(s) for different querying tasks.

5.4.4. Framework for Different Strategies

When we develop the protocol LBAG, we found LBAG can work as a framework of data aggregation for vehicular networks. In section 5.4.3.4, we have given the rule of mapping real nodes to logic entities. The role of aggregator is actually determined by aggregation area. When the aggregation area overlaps the query area, then all Query Responders also take the role of aggregator. RAW DATA are actually sent to the query area, so that all nodes inside will receive the

message and start collecting raw data. This behavior is the same as that of fully distributed aggregation.

Similarly, when the aggregation area overlaps the sender location, the raw data are sent to the sender's location directly. The sender now has two roles to play: query sender and aggregator. It collects the raw data and does aggregation when its Agg timer expires.

Therefore, the protocol of LBAG can emulate the behavior of centralized and fully distributed strategies. Including its own location based proxy strategy. LBAG is able to provide different aggregation strategy to its applications. Such a framework has a big advantage, because the data aggregation can be adapted to the scenario parameters and requirements flexibly.

6. Implementation

In order to evaluate the performance of Location Based Aggregation, the protocol is implemented for the network simulator ns-2.

6.1.NS-2 Class Overview

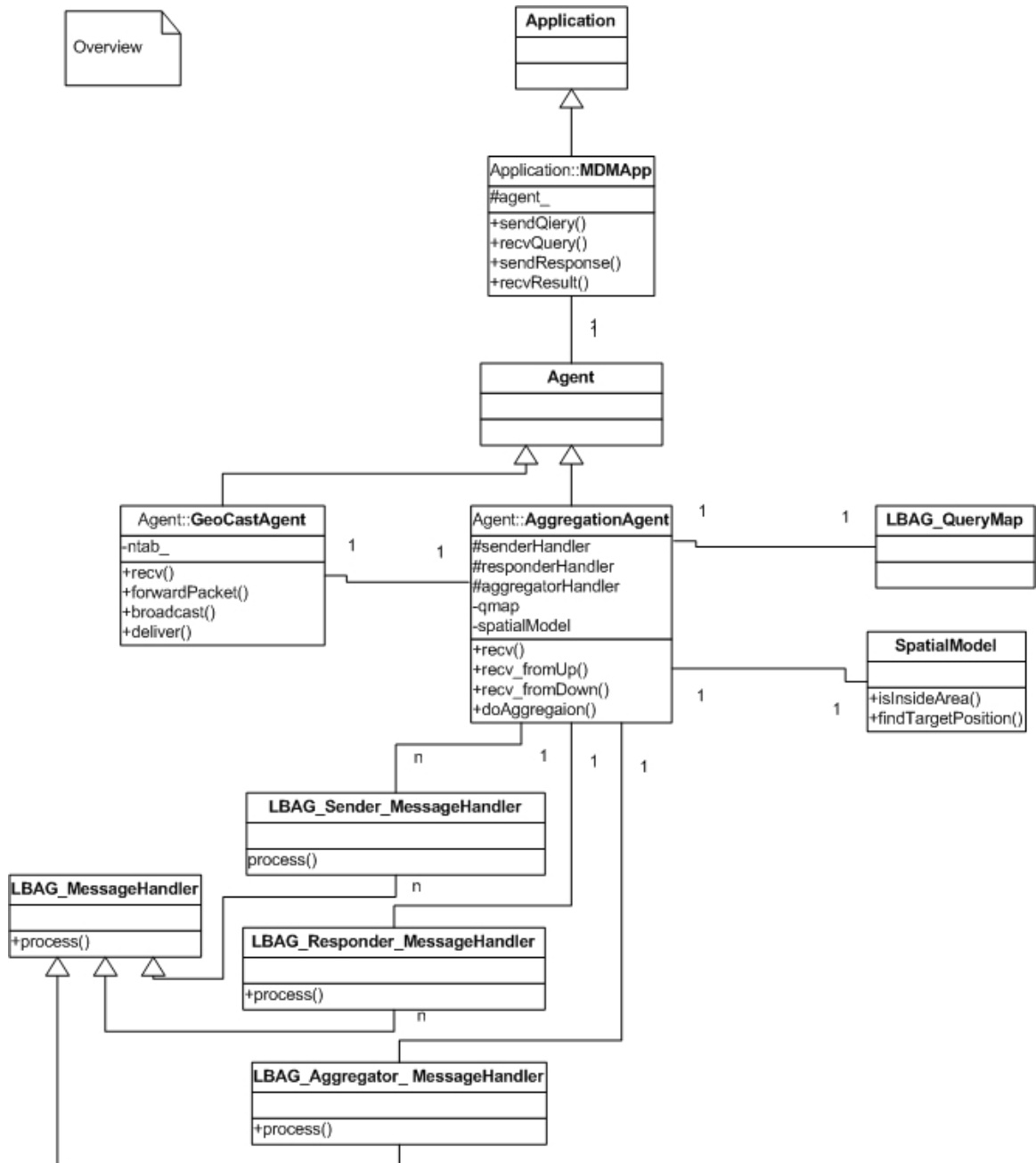


Figure 6-11 Overview of UML Class Diagram1

The implementation of Location Based Aggregation is for ns2 simulator. Apart from LBAG, Spatial Model, GEOCAST and an application MDMAApp are implemented, so that the complete architecture is constructed.

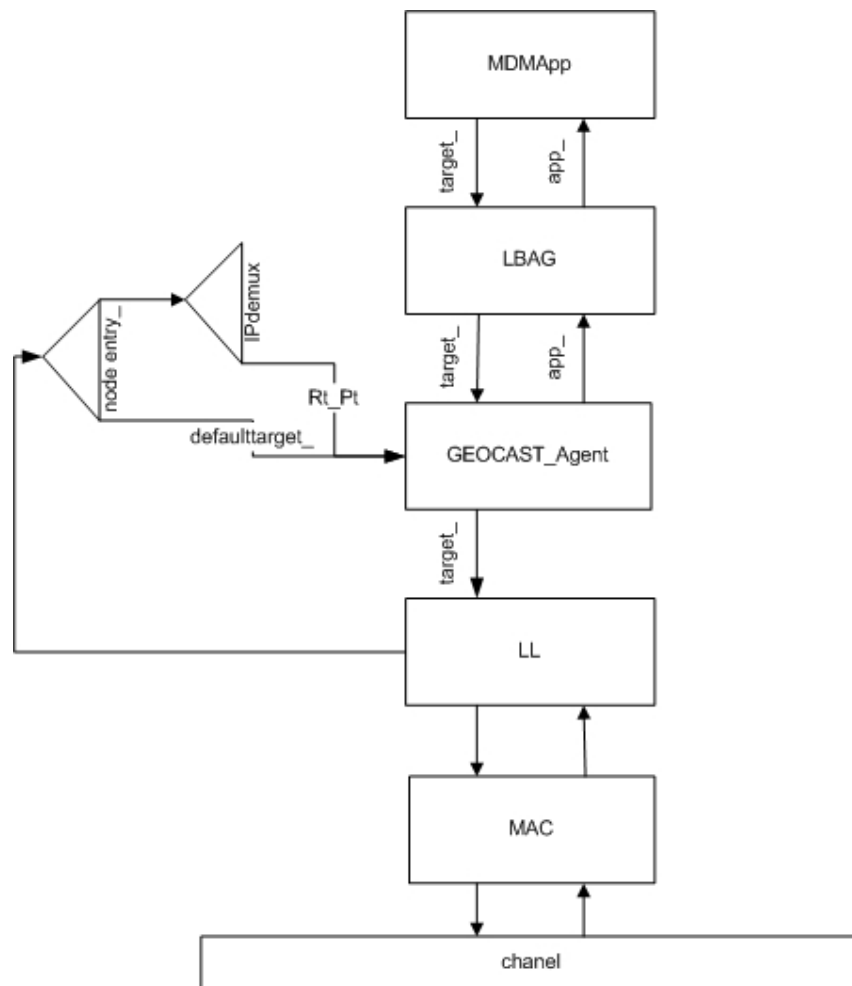


Figure 6-2 ns-2 Node Architecture

1.1.1.15. Packet Structure

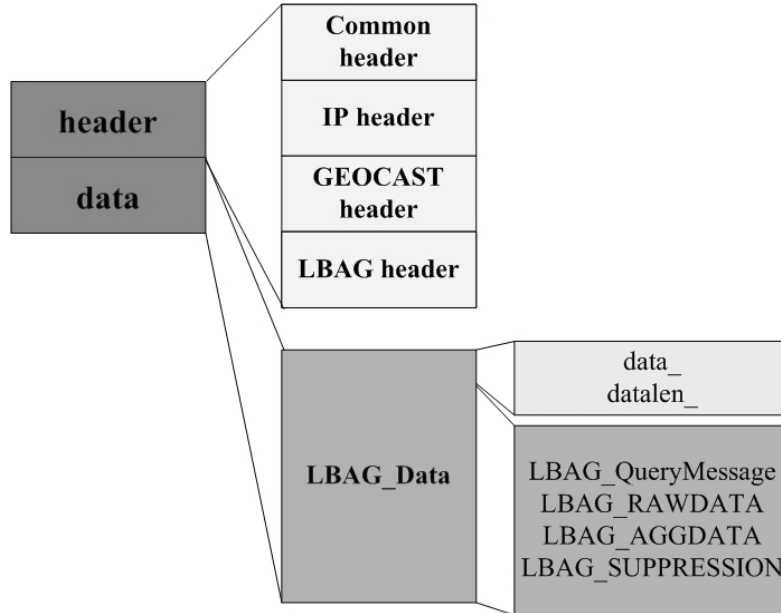


Figure 6-3 ns-2 Packet Format

The format of a packet in ns-2 simulator is shown in Figure 6-3. According to ns-2 convention, each packet has a header and a data. The header part is a list of headers of different protocols. We remove the headers that are not useful to our simulation, so that each packet occupies less memory space. The remaining headers are common header, IP header, GEOCAST header and LBAG header. Fields of these headers are listed in the Appendix1.

1.1.1.16. Class Hierarchy

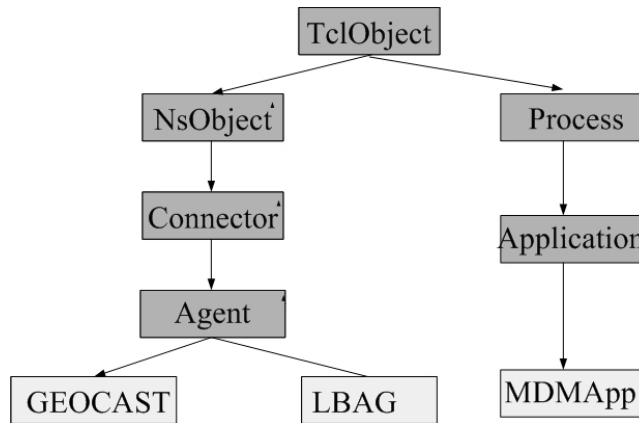


Figure 6-4 NS-2 Class Hierarchy

Figure 6-4 shows the class hierarchy of ns-2 objects. The GEOCAST_Agent and LBAG_Agent are both inherited from Agent. MDMApp is inherited from Application.

6.2. Class SpatialModel

6.2.1. Class Diagram

The spatial model we implemented here is a hierarchical model, which includes semantic information. It is based on a graph structure. We provide an interface to access geographic and semantic information by other modules. We use boost graph library to represent the road network. Semantic information is stored with the edge and vertices. Many maps are created to show the relationship between geographic and semantic information. Road network topology is read from a graph file.

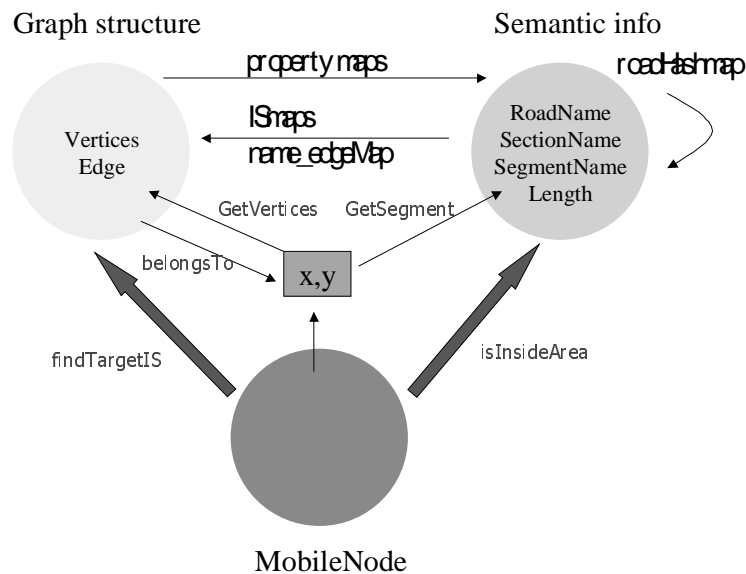


Figure 6-5 Spatial Model

6.2.2. Data Structure

There are two parts of information in the Spatial Model: geographic information and semantic information. Hash maps are used to construct the connection between them.

We use the adjacency graph to store the geographic information (network topology). The semantic information is stored as the edge or vertex properties. Boost Graph Library provides property maps, which allows to access property value by the edge or vertex ID.

Categories	Name	Meaning
Geographic	<i>h_graph</i>	Graph, according to boost graph library ⁵

⁵ boost graph library

information		
Semantic information	<i>roadHashmap</i>	A tree-form road network
Mutual mapping between geographic and semantic information	<i>vtx_pm</i>	map with key vertexID and value vertex properties
	<i>seg_map</i>	map with key edgeID and value segment name
	<i>sec_map</i>	map with key edgeID and value section name
	<i>road_map</i>	map with key edgeID and value road name
	<i>weight_map</i>	map with key edgeID and value distance
	<i>secName2IS_map</i>	map with key section name and value list of included intersections
	<i>roadName2IS_map</i>	map with key road name and value list of included intersections

Table 6-1 Data Structure of Spatial Model

The roadHashmap stores the hierarchical information of semantic names. The hierarchy is in tree form: Road-Section-Segment. Through this hash map, sub-areas of a region that defined by semantic name can be obtained.

6.2.3. Common Interface

The functions provided by the common interface are listed in Table 6-6. Other modules can obtain spatial information by these functions.

Functions	Description
<i>getLocation</i>	Get coordinate of current position
<i>GetSegment</i>	Get the semantic information of a segment which includes the given point or the given pair of intersections
<i>GetVertices</i>	Get the intersections that enclose the edge on which the given position lies
<i>findTargetIS</i>	Determine the closest Intersection to on the border of the given area from the given point
<i>GetDistance</i>	Compute the geometric distance between two given points
<i>isInsideDestArea</i>	test whether the given point is inside the given area

Table 6-2 Common Interface

6.3. Class GEOCAST_Agent

6.3.1. Class Diagram

Class GEOCAST_Agent implements the GEOCAST routing protocol. Its structure is shown in UML diagram Figure 6-6.

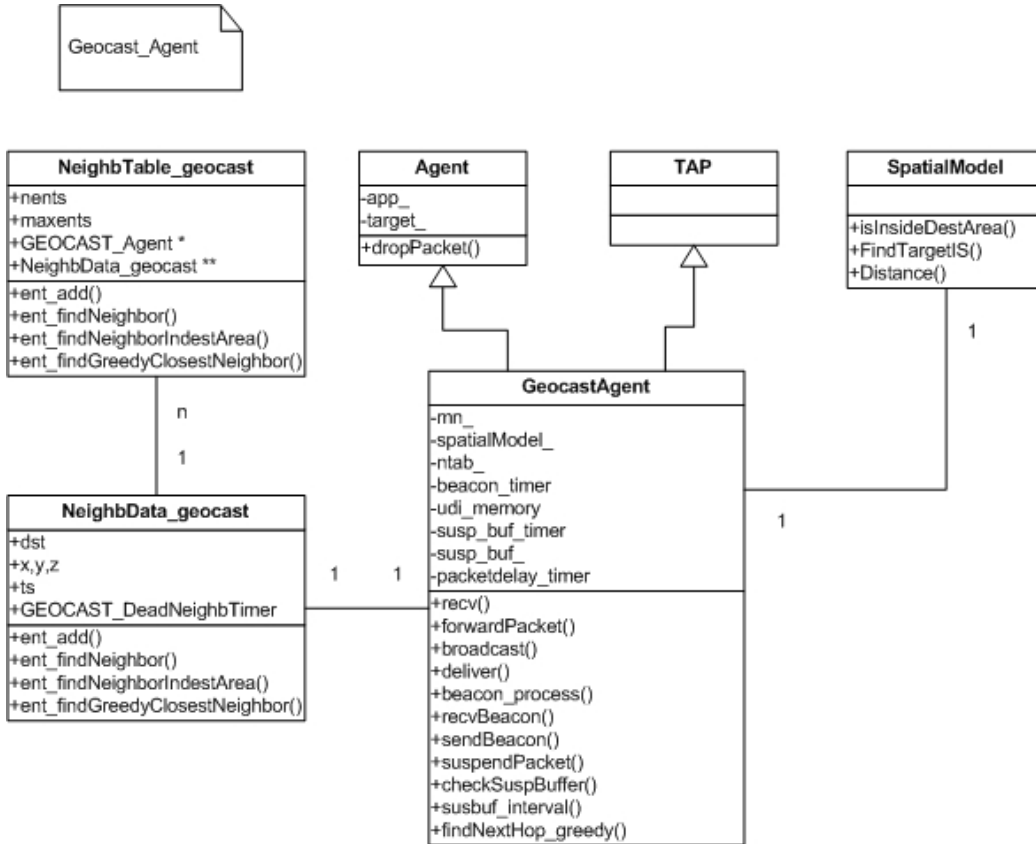


Figure 6-6 ULM diagram of GEOCAST_Agent

6.3.2. Data Structure

Type	Name	Value	Meaning
<i>DestArea</i>	<i>destArea</i>		GEOCAST Area
<i>int</i>	<i>mode_</i>	GEOCASTH_DATA_BROADCAST GEOCASTH_DATA_FORWARD	GEOCAST message type
<i>double</i>	<i>ox, oy</i>	Filled by GEOCAST when it receives a fresh packet	Position of packet origination

Table 6-3 GEOCAST Header

Geocast header contains an important parameter: destination area. The format of *DestArea* is a structure, which can contain a list of semantic names. The current implementation supports the semantic names in segment level, section level and road level.

6.3.3. Functions

The most important functions of GEOCAST are its routing decision functions.

- *recv()*
- *broadcast()*
- *forwardPacket()*

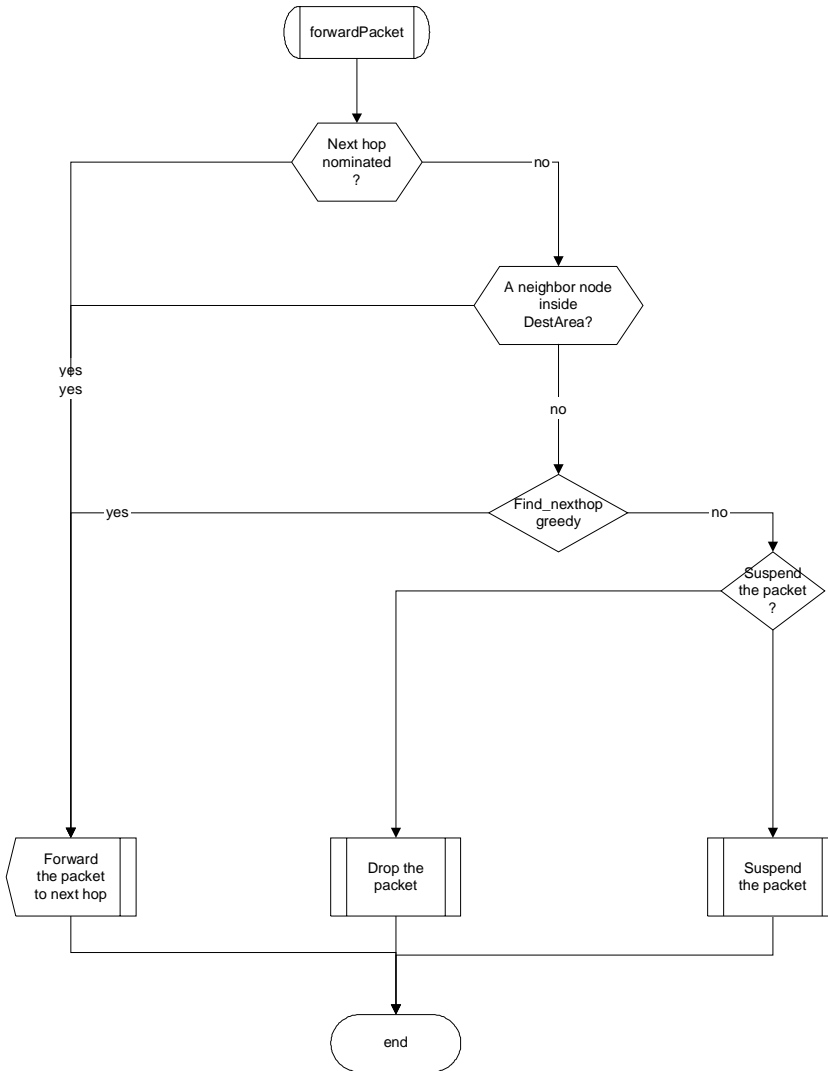


Figure 6-7 Function *recv()*

Member function *recv()* is in charge of checking the type of incoming packets and call corresponding functions to handle the packet. The logic of this function is shown in Figure 6-7.

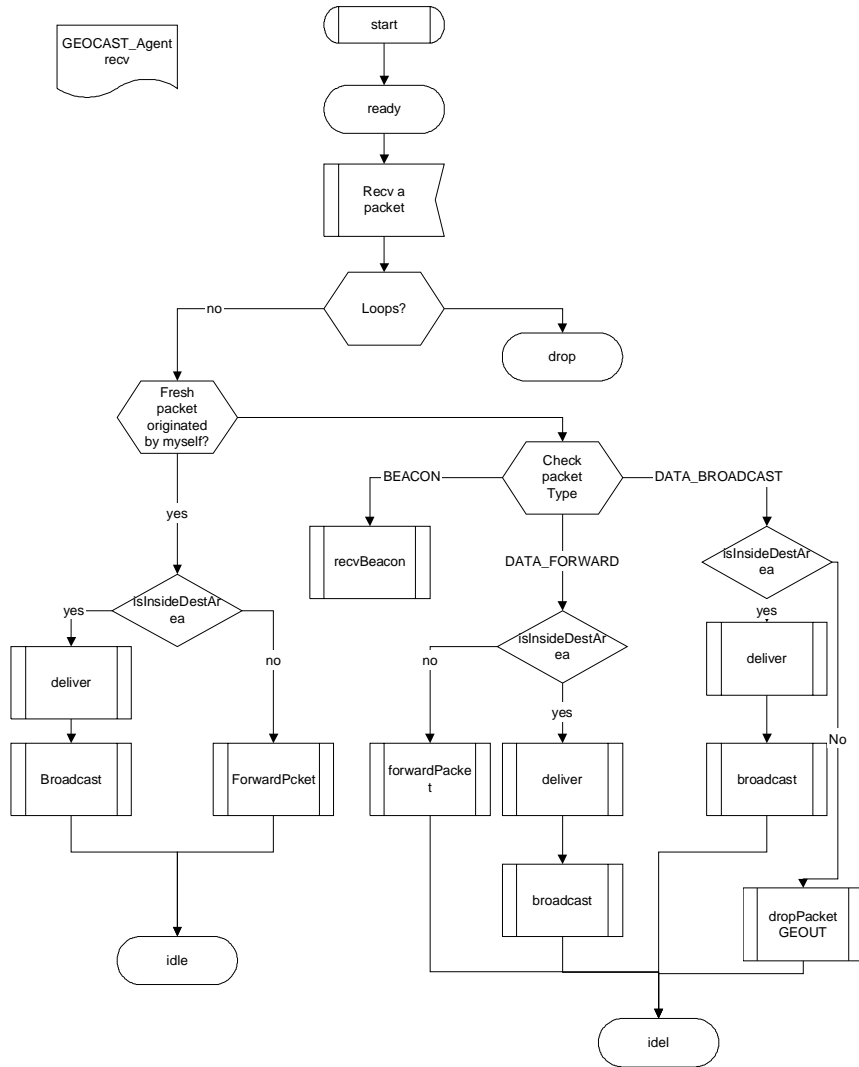


Figure 6-8 Function forward()

Member function forward handles the packet that is not yet in the destination area. *forward()* uses information in *Neighbor Table* to make decisions on how a packet is further handled: broadcast, forward or drop. Figure 6-8 shows the logic of *forward()*

6.4. Class MDMApp

Class MDMApp is used as a query application. MDMApp donates the Mobile Data Management. Its name is actually a much more broad category than the information querying serves, which we are using. But since it is used in the implementation, we still document it here with this name. But in other part of the thesis, QueryApp is used to make it more understandable.

6.4.1. Class Diagram

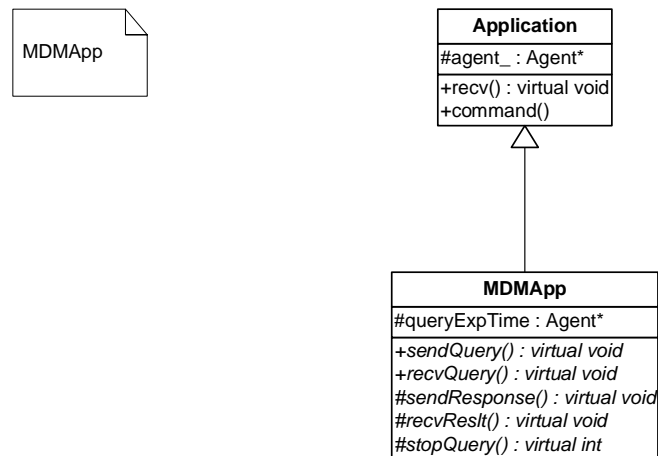


Figure 6-9 Class Diagram of Query Application

Class MDMApp is derived from Class Application. It contains a pointer to the lower agent and has functions to generate and process queries.

6.4.2. Protocol Operations

MDMApp is developed to generate and process queries. It provides four member functions to do the task.

- *sendQuery* Generate a query, which is defined by Tcl command
- *recvQuery* Receive a query from lower layer
- *sendResponse* Generate a response message upon receiving a query
- *recvResult* Retrieve the query result

The state machine chart is shown in Figure 6-10, which shows how MDMApp works on a query sender and a Query Responder.

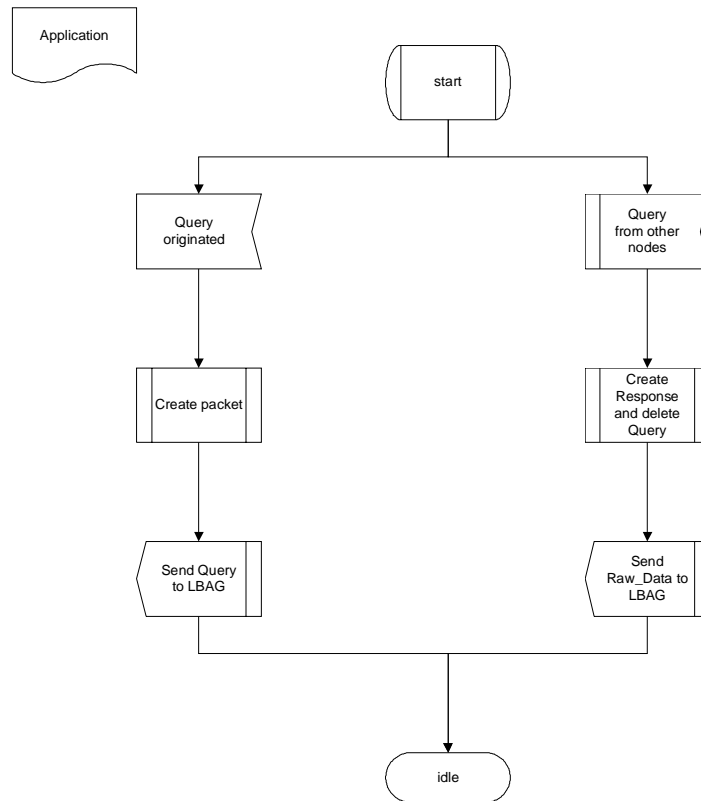


Figure 6-10 State Chart Diagram of MDMApp

1.1.1.17. Commands to define a query in Tcl script

Function Name	Value
<i>setQueryType</i>	SNAPSHOT
<i>setAggStrategy</i>	PROXY_AGG
<i>setDataName</i>	“speed”
<i>setDataType</i>	LBAG_DOUBLE
<i>setQueryOperation</i>	MAX
<i>adddest</i>	ROAD C
<i>setAGGArea</i>	B_3

Table 6-4 Tcl Functions to define a query

1.1.1.18. Command to call *sendQuery* in TCL script

Tcl command *sendQuery* is used to start the query operation.

6.5. Class LBAG_Agent

Class LBAG_Agent implements the respective data aggregation algorithm. It is derived from the super class Agent. Its data elements includes a Query Map and

6.5.1. Class diagram

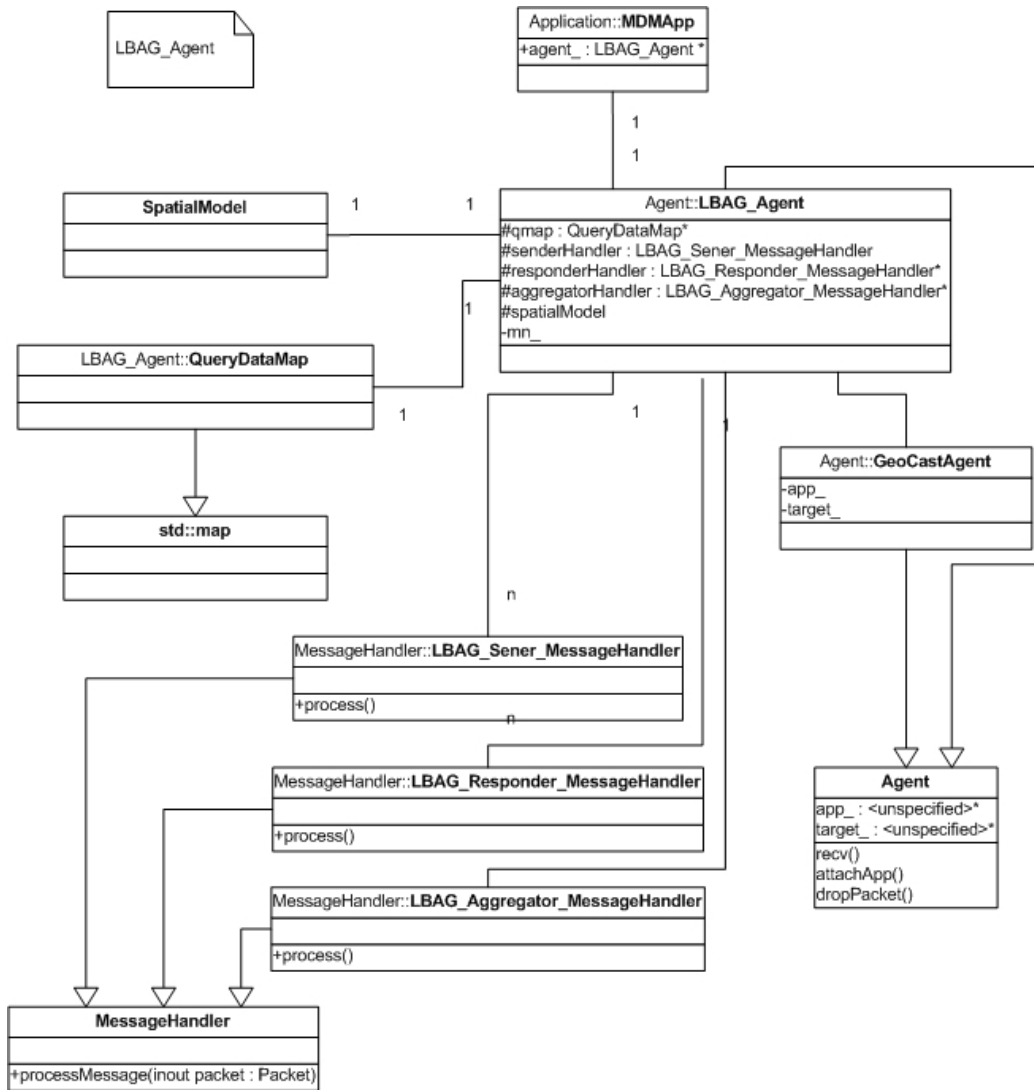


Figure 6-11 LBAG UML Class Diagram

6.5.2. Data Structure

1.1.1.19. Protocol Messages

LBAG Header

Type	Name	Value	Meaning
<i>static</i>	<i>offset_</i>		ns 2 requirement
<i>LBAG_PKT_TYPES_t</i>	<i>pktype</i>	LBAGH_QUERY=0 LBAGH_RAWDATA=1 LBAGH_AGGDATA=2 LBAGH_SUPPRESSION=3	Through this indicator, protocol is able to know how to interpret the packet body
<i>LBAG_QueryID_t</i>	<i>qid</i>	Initiated at <i>allocLBAGpacket()</i> <i>qidcnt++</i>	Query ID
<i>LBAG_OP_t</i>	<i>op</i>	MAX/MIN/SUM/AVG/COUNT	Determines which operation to use in data aggregation
<i>ns_addr_t</i>	<i>nid</i>		Generated at initialisation
<i>LBAG_DataName_t</i> (<i>char [16]</i>)	<i>dataName</i>		Filled according to query message
<i>LBAG_DataType_t</i>	<i>dataType</i>	DOUBLE INT	
<i>double</i>	<i>expireTime</i>		System time
<i>double</i>	<i>generationTime</i>		

Table 6-5 LBAG Header

In the implementation, some common fields about a query are moved to the header, only in order to avoid checking packet body frequently.

LBAG_QueryMessage

Type	Name	Meaning
<i>LBAG_Query_t</i>	<i>query_type</i>	Snapshot or continuous
<i>LBAG_Strategy_t</i>	<i>agg_strategy</i>	Specify the aggregation strategy: centralized / fully distributed / LBAG
<i>struct DestArea</i>	<i>query Range</i>	Query Area
<i>bool</i>	<i>useAGG</i>	Do fully distributed aggregation in segment level or not
<i>Name_t</i>	<i>aggregationarea</i>	Aggregation area

Table 6-6 LBAG_QueryMessage

LBAG_RawData

Type	Name	Meaning
<i>LBAG_QueryMessage_t</i>	<i>query</i>	The complete query message
<i>LBAG_DataValue_t</i>	<i>dataValue</i>	Value

Table 6-7 LBAG_RawData

LBAG_AGGData

Type	Name	Meaning
<i>LBAG_QueryMessage_t</i>	<i>query</i>	The complete query message
<i>DestArea</i>	<i>coverage_area</i>	List of covered segments
<i>LBAG_DataValue_t</i>	<i>dataValue</i>	Value
<i>int</i>	<i>completeness</i>	Number of raw data

Table 6-8 LBAG_AggData

Raw Data stores the data value. The ID of the responder can be found in the header. In an Agg Data, the coverage area includes a list of segments, where its raw data are from. The completeness is the number of raw data included in computing that aggregated data.

1.1.1.20. Class LBAG_QueryMap

QueryMap is a data member of class *LBAG_Agent*. It stores information about all known queries. `<map>`¹ is used to implement it. The key of the map is *qid* (unique query ID) and the value is of type *LBAG_QueryMapEntry*, which stores information about one query as shown in the following table.

Each node should have a data structure to store information related to every querying tasks.

Type	Variable name	Description
<i>LBAG_QueryID_t</i>	<i>qid</i>	Unique ID
<i>LBAG_QueryType_t</i>	<i>queryType</i>	Type of Query and strategy for aggregation
<i>LBAG_Strategy_t</i>	<i>aggStrategy</i>	
<i>LBAG_OpType_t</i>	<i>op_</i>	Description of a query operation
<i>LBAG_DataName_t</i>	<i>dataName</i>	
<i>LBAG_DataType_t</i>	<i>dataType</i>	
<i>double</i>	<i>senderx</i>	Position of query sender
<i>double</i>	<i>sendery</i>	
<i>LBAG_QueryExpireTimer*</i>	<i>expireTimer</i>	Related timers
<i>LBAG_QueryAGGTimer*</i>	<i>aggTimer</i>	
<i>LBAG_QueryResultList*</i>	<i>resultList</i>	Buffer to store received data
<i>LBAG_DataValue_t</i>	<i>summaryResult</i>	Aggregate value
<i>bool</i>	<i>Sender</i>	The role played by this node for this query
<i>bool</i>	<i>Aggregator</i>	
<i>bool</i>	<i>Responder</i>	
<i>struct DestArea</i>	<i>queryRange</i>	Query area and aggregation area
<i>Name_t</i>	<i>aggregationArea</i>	
<i>LBAG_MessageHandler*</i>	<i>msgHandler</i>	Descriptor of message handler
<i>queryState</i>	<i>Lbag_queryState</i>	

Table 6-9 Query Map Entry

Result List is a vector, which is used to store the received raw-data or partial aggregated data. The node performs aggregation based on it. When LBAG of a node receives a message, it knows how to process the message by checking the message type and information in the querymap.

6.5.3. MessageHandler

MessageHandler processes the given message according to defined logic. Three logical entities in

¹ `<map>` from STL (C++ standard template library)

LBAG: Query Sender, Query Responder and Aggregator have different logics to process the LBAG messages. Each logic entity is implemented by a message handler. The class *MessageHandler* is a base class, which defines a virtual function *process()*.

1.1.1.21. LBAG_Sender_MessageHandler

The Query Sender is in charge of two types of messages: query message from application layer and AGG DATA from lower layer.¹

After the arrival of a query message from application, the Sender_Handler registers the qid if it is new to this node and passes it to GEOCAST to send it. In addition, it set the Agg Timer.

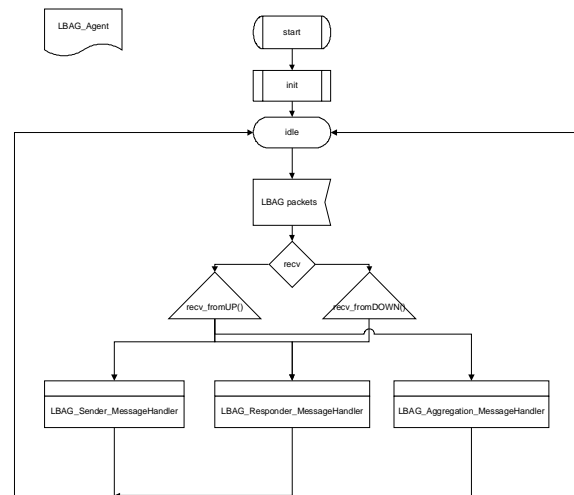


Figure 6 12 LBAG recv function

¹ For centralized strategy, the sender and aggregator are both on the Query Sender node.

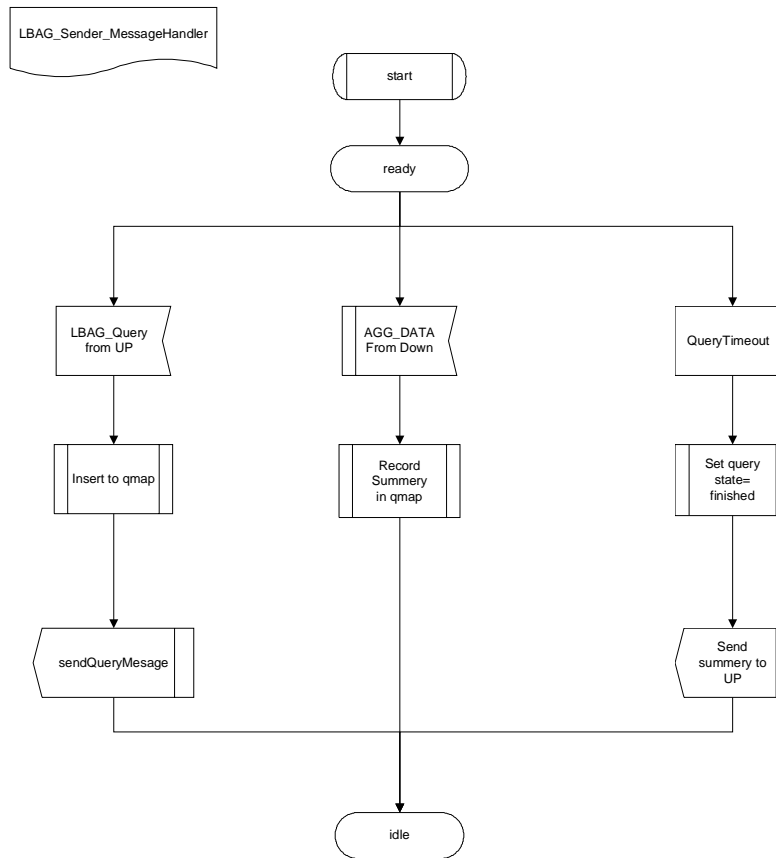


Figure 6-13 SDL of Sender Handler

1.1.1.22. LBAG_Responder_MessageHandler

When a Query Responder receives a QUERY MESSAGE, it checks whether it is responsible for the query. If yes, then it passes the query to its upper MDMAApp layer. The MDMAApp processes the query and return a raw data to the Responder handler. The Responder Handler will record the value in its *qmap* and passes it to the underlying GEOCAST_Agent.

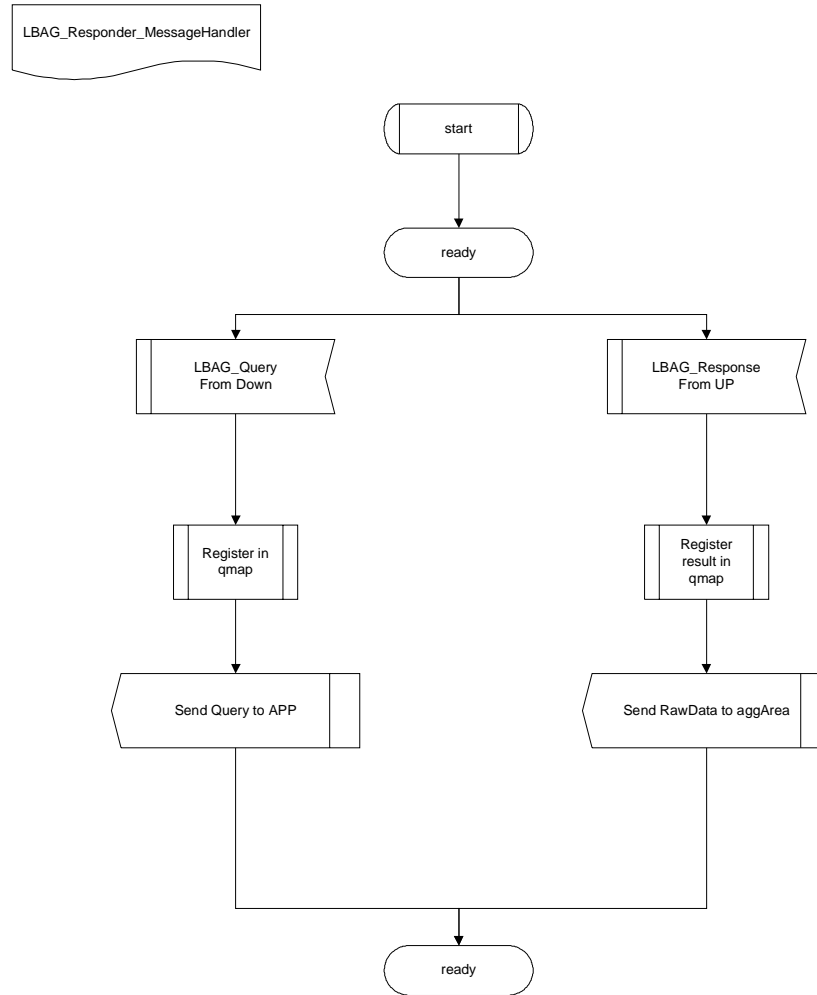


Figure 6-14 Query Responder SDL Diagram

1.1.1.23. LBAG_Aggregator_MessageHandler

RAWDATA for a query is sent to Aggregator. When aggregator receives a RAW DATA, it first checks whether there is an entry existing in the *qmap* for that query. If not, then it creates a new one and fills information about the query into it. Then the aggregator takes the value and its context from the RAWDATA and stores them in the result list. When the timer expires, the *doAggregation()* is called to generate aggregated data and send out AGGDATA message and SUPPRESSION.

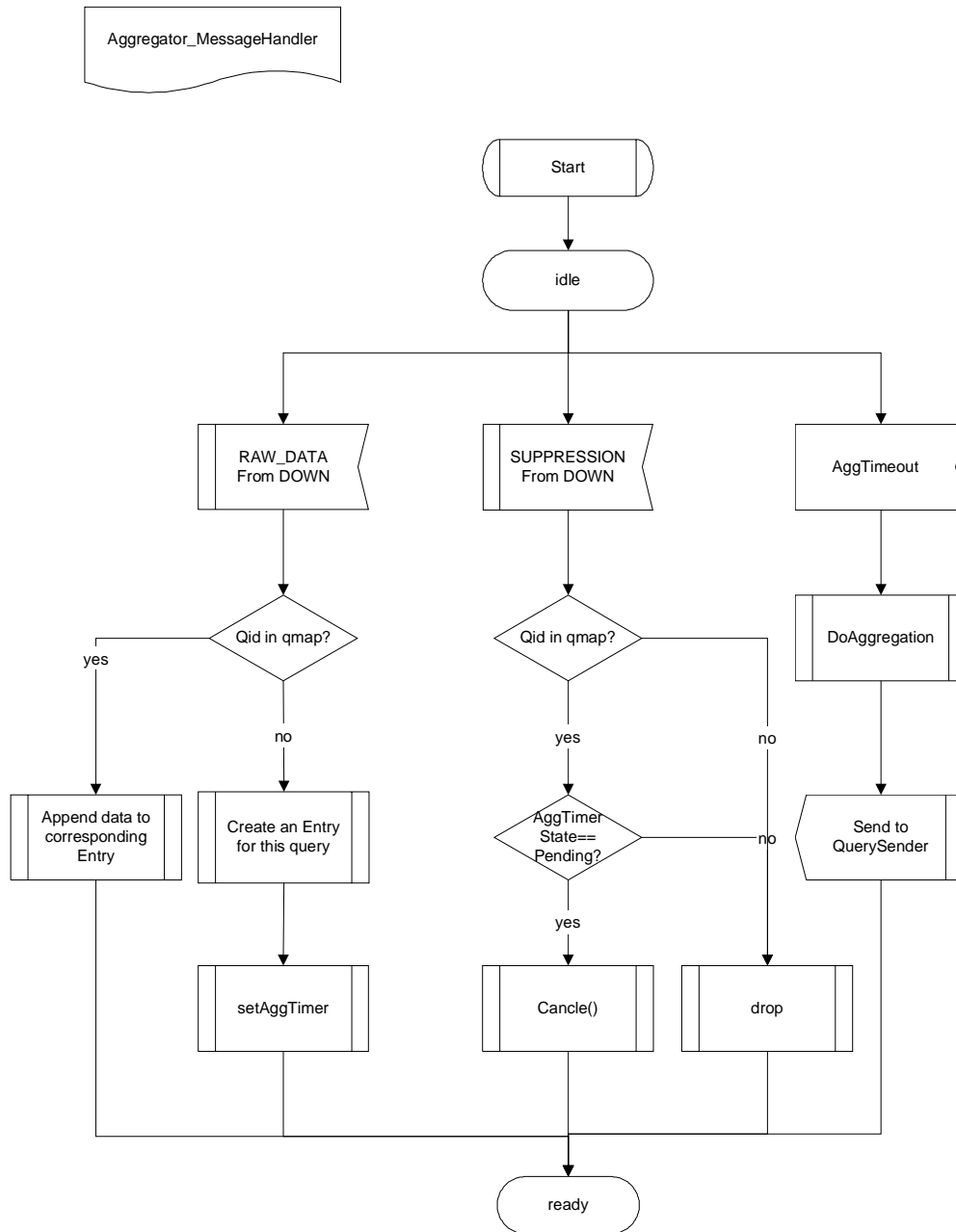


Figure 6-15-Aggregator SDL Diagram

6.5.4. Summary of Implementation

The implementation of LBAG protocol is especially for ns-2 simulator, so the structure of a packet and the software structure for a node is different from what we described in the previous chapters.

7. Simulations

Our implementation LBAG tries best to overcome the challenges from high mobility and scalability. LBAG do aggregation relies on a hierarchy of locations. By adding semantics to the spatial model, LBAG cooperates with GEOCAST ad hoc routing and provides semantic meanings to the aggregated result. In addition, it is able to emulate the behaviors of other aggregation strategies as well, thus provides a framework for data aggregation in vehicular networks. In this chapter LBAG is simulated in ns-2 simulator. The performance of location based proxy strategy is compared with that of centralized and fully distributed strategies.

7.1. Simulation Environment

The simulations are done with the network simulator ns-2. Simulation scenario is based on road networks, and the mobile nodes are in high mobility. Data aggregation is triggered by Query message.

7.1.1. Mobile Node Setup

During the setup of a mobile node, IEEE 802.11 is used as MAC layer; GEOCAST is selected as the default routing protocol; LBAG is attached to provide data aggregation service and MDMAApp is attached as an application. By using Tcl script commands, LBAG and GEOASAT are linked mutually, so are the MDMAApp and LBAG. Spatial Model is initialized as a member element in LBAG and GEOCAST. The following Figure 7-1 shows the original tcl script to setup a node.

```
for {set i 0} {$i < $val(nn) } {incr i} {
#setup mobile nodes
  set node_($i) [$ns_ node]
  $node_($i) random-motion 0;      # disable random motion
#attach LL and GEOCAST to node
  if { $val(adhocRouting) == "GEOCAST" } {
    set ragent [$node_($i) set ragent_]
    $ragent install-tap [$node_($i) set mac_(0)];
    # prepare promiscuous mode
  }
# Set GEOCAST as the default routing agent
  set geocast_($i) $ragent

# Create LBAG, attach it to Node and link it with GEOCAST
  set lbag_($i) [new Agent/LBAG]
  $ns_ attach-agent $node_($i) $lbag_($i)
  $lbag_($i) mobilenode $node_($i)
  $lbag_($i) attach-ragent $ragent
# Create MDMAApp and attach to LBAG
  set mdmApp_($i) [new MDMAApplication]
  $mdmApp_($i) attach-agent $lbag_($i)
$god_ new_node $node_($i);      # bring nodes to God's attention
}
```

Figure 7-1 Mobile Node Setup

7.1.2. Spatial Model Setup

The Spatial Model we used in our simulation is based on a road network. Figure 7-2 shows its

topology. It includes 12 edges and 13 vertices. Each edge has its semantic names.

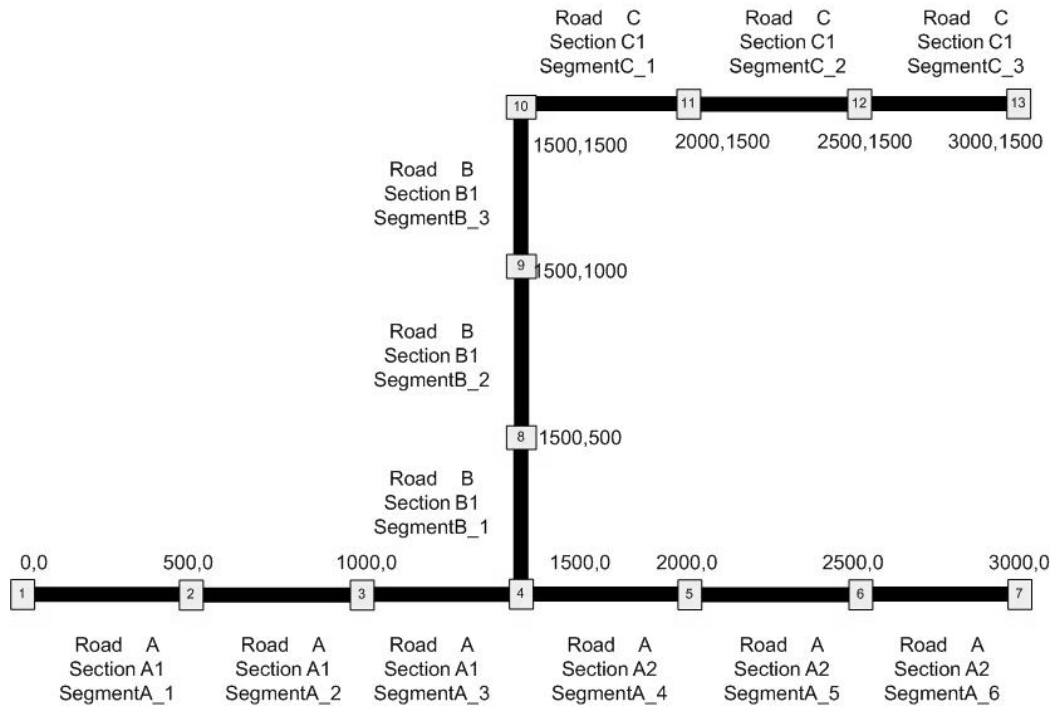


Figure 7-2 Simulation Road Graph

All topology, geographic and semantic information are stored in a file, which is required by Spatial Model for its initialization.

```

vertex
13
0      0.10      0.10
1      500.0     0.10
2      1000.0   0.10
3      1500.0   0.10
4      2000.0   0.10
. . . . .

edge
12
0      1      0      A      A1      A_1
1      2      1      A      A1      A_2
2      3      2      A      A1      A_3
. . . . .

10     11     10     C      C1      C_2
11     12     11     C      C1      C_3

```

Figure 7-3 Spatial Information File

Figure 7-3 shows the information representation format in the file. The first part of the file is the information about vertices. The second part is about the edges. Semantic names of each edge are recorded at the rear part of each line.

7.1.3. Simulation Scenarios

The network scenario for simulation is based on the same network topology which is used Spatial Model. The size of the road network is 3000×1500m. 200 mobile nodes are distributed in the road network.

Mobile nodes in our scenario move at a high speed chosen between 72 and 180kmh. A tool called “*setgraphdest*” is used to create movement patterns of mobile nodes. In order to constrain the node movement, *setgraphdest* randomly select two vertices, one as a node’s starting point and the other as the destination of its movement. Then it computes the shortest path in between, and enforces the node to travel along that defined route. In the simulation, the distribution of nodes can be assumed as uniformly distributed, after 10 seconds warm-up phase..

7.1.4. Protocol Parameters

There are several protocol parameters need to be defined for the simulation.

Parameters of GEOCAST	Value
Beacon Interval	0.3s
Beacon expire time	0.5
Rebroadcast modifier coefficient <i>k</i>	0.001
Guard distance	30m
Parameters of MAC	Value
Radio Range	250m
Parameters of LBAG	Value
Aggregation Expire Time	2s
Query Expire Time	5s

Table 7-1 Protocol Parameters

7.1.5. Query Definition

In the simulation Tcl commands are used to define a query message on a mobile node. Table 7-2 shows the set of Tcl commands to define a Query. If the values in the table are used to define a query, the obtained query message is “get max speed from nodes in Road C”. Another Tcl command *sendTestQuery* is used to trigger the sending of the defined Query, thus starting a data aggregation.

Tcl Command	Value used in simulation
<i>setQueryType</i>	SNAPSHOT
<i>setAggStrategy</i>	CENTRALIZED_AGG
	PROXY_AGG
	LOCAL_AGG
<i>setDataName</i>	“speed”
<i>setDataType</i>	LBAG_DOUBLE
<i>setQueryOperation</i>	MAX
<i>addest</i>	ROAD C
<i>setAGGArea</i>	SEGMENT B_3

Table 7-2 Define Query Message

7.1.6. Three Strategies

Three aggregation strategies are chosen to test by simulation: centralized strategy, fully distributed strategy and location based proxy strategy. The simulation result is studied to compare their performances.

Centralized Strategy

As shown in Figure 7-4, all Query Responders send their response message to the Query Sender, which will collect response messages and do aggregation by itself.

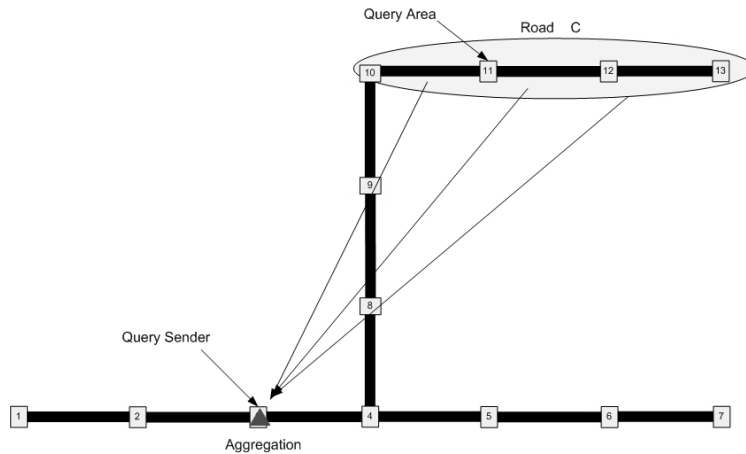


Figure 7-4 Simulation with Centralized Strategy

Fully Distributed Strategy

As shown in Figure 7-5, all Query Responders send their response message to other Query Responders. Each Query Responder collects response messages. When Aggregation Timer expires, every Query Responder does data aggregation. But only one aggregated data is sent back to the Query Sender.

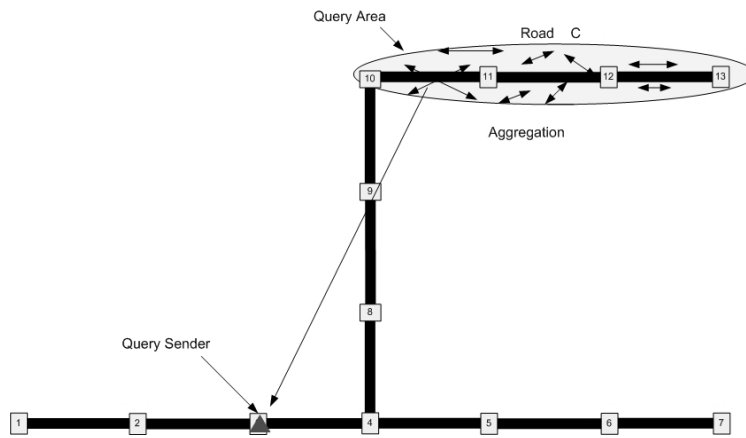


Figure 7-5 Simulation with Fully Distributed Strategy

Location based Proxy Strategy

As shown in Figure 7-6, all Query Responders send their response message to the aggregation area. The aggregation area chosen for the simulation is Segment B_3. Every node in the aggregation area becomes Aggregator for that query and collects response messages. When Aggregation Timer expires, each Aggregator does data aggregation. But only one aggregated data is sent back to the Query Sender.

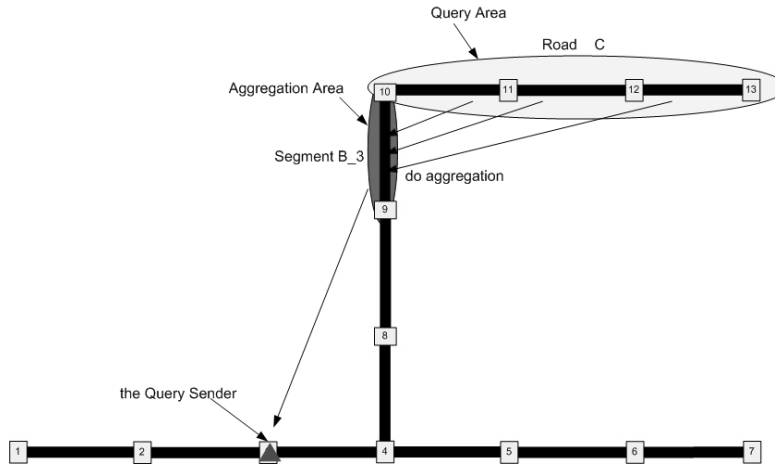


Figure 7-6 Simulation for Location based Proxy Strategy

7.1.7. Behavior Emulation by LBAG

Due to the fact that the implemented protocol LBAG is able to emulate the behaviors of centralized and fully distributed strategies, implementations for those two strategies are not developed.

To emulate the behavior of centralized strategy, LBAG sets the area, where the Query Sender is in, as the aggregation area. According to LBAG, response messages are routed to that area, thus the Query Sender can collect response messages by itself. In addition, when a control flag called *AggregationStrategy* is set to “*CentralizedAgg*”, all the neighboring nodes of the Query Sender will not do aggregation. In such a way, only the Query Sender does aggregation in the whole network. It will generate an aggregated data by itself.

To emulate the fully distributed strategy, LBAG sets the complete query area as the aggregation area. According to LBAG, every Query Responder sends response message to the query area once again. Because a message is geocasted inside the query area, all the other Query Responders get it. In addition, every Query Responder becomes Aggregator and begins to collect data. When the Aggregation Timer expires, Aggregator, i.e. the Query Sender does data aggregation. An aggregated data is sent to the Query Sender afterwards.

7.1.8. Location of the Query Sender

In order to compare the performance of strategies with different scenario settings, the Query Sender is placed at different locations. Figure 7-7 shows the 6 chosen positions.

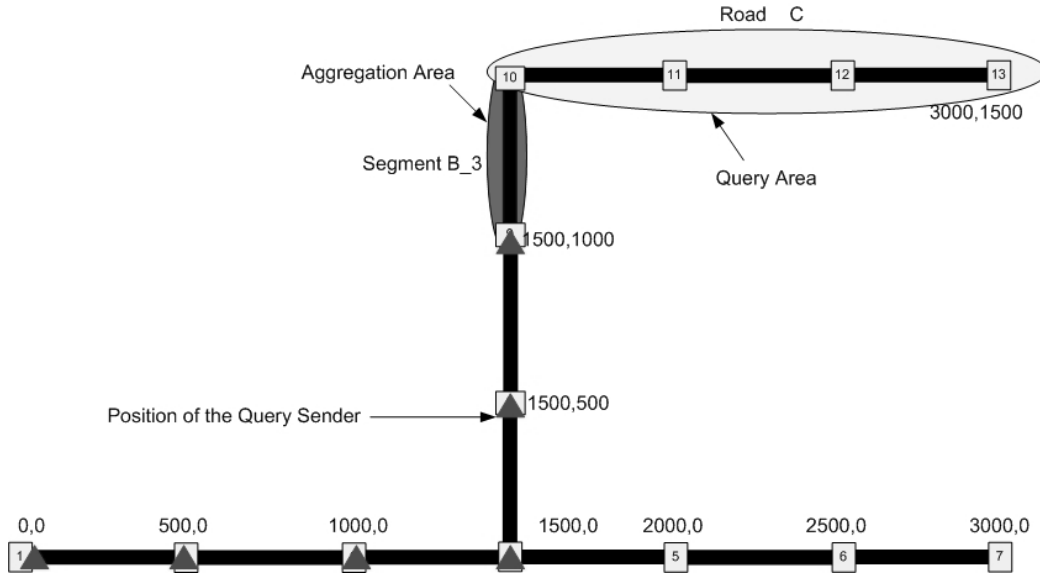


Figure 7-7 Location of the Query Sender

7.2. Simulation Results

We evaluate the protocols using four metrics: the overall communication overhead, the communication overhead from Broadcast, The communication overhead from Forwarding and the completeness of aggregated data.

7.2.1. Completeness of Aggregated Data

The completeness of the aggregated data reflects the percentage of response messages that are actually calculated in aggregated data in contrast to all the sent response messages by Query Responders. The measured results for centralized, fully distributed and location based proxy aggregation strategies are shown in Figure 7-9.

Simulation result shows the Query Sender is able to get aggregated data with different completeness. In addition, the aggregated data is also with semantic meanings. In our simulation, the aggregated result means “the maximum speed of nodes on Road C is xxx”.

However, we also noticed the difference in the completeness of aggregated data from different aggregation strategies.

1) The aggregated data from fully distributed strategy has the highest completeness, but the one from centralized strategy has the lowest completeness. This result accords with our expectation. The fully distributed strategy has a higher completeness because its aggregations are done inside the query area. Response messages reach the other Query Responder only with few hops of

rebroadcast. Therefore, the possibility of message loss is low. For location based proxy strategy, response messages need to be forwarded to the aggregation Area. Thus every response message experiences more hops than the fully distributed strategy. Because of the unreliable wireless communication, each message has a higher lost rate with more hops.

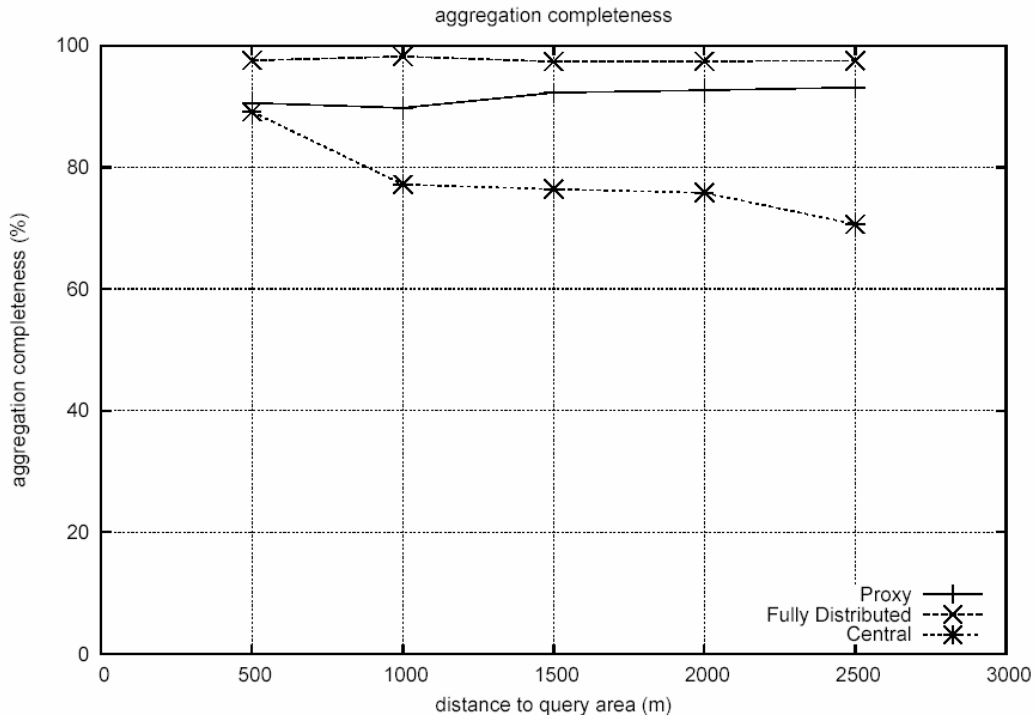


Figure 7-8 Aggregation Completeness

Therefore, the average number of response messages that arrive the aggregators are less than that in the fully distributed case. It's the reason why proxy based strategy has a lower completeness. Similarly, fewer response messages are able to reach the area where the Query Sender is located, when the distance between it and the query area increases.

2) The completeness with fully distributed and location based proxy strategy keeps almost constant, while that of centralized strategy decrease linearly. The completeness with fully distributed strategy keeps between 93-98% and with proxy strategy between 90-95%, regardless of the distance between the Query Sender and the query area. This is because for those two aggregation strategies, location of aggregation area is irrelevant to the distance between the Query Sender and query area. Only the distance from the query area and aggregation area will influence the completeness of location based proxy strategy.

From the above observation, we get the following conclusion: 1) The completeness of aggregated data is influenced by the distance between the query area and the area where data aggregation is done. 2) Completeness decreases linearly proportional to that distance.

7.2.2. Communication Overhead from Forwarding

The communication cost can be divided into two parts. One part is from Forwarding, which refers

to the packets forwarded by ad hoc routing GEOCAST. Another part is from Broadcast, which refers to the packets rebroadcasted by GEOCAST. Figure 7-9 shows the communication overhead from Forwarding. In section 4.3.8 and 4.4.4 we have mentioned about the routing and aggregation part of the communication cost. Here, the forwarding messages correspond to the routing cost and the rebroadcast messages to the aggregation cost.

1) The fully distributed strategy has lowest Forwarding cost. Location based proxy and centralized strategies start with low. The Forwarding cost of proxy strategy keeps almost constant while that of centralized one increase very fast. But this result accords with our expectation. For centralized strategy, each response message is forwarded to the Query Sender. Thus the Forwarding cost is proportional to the number of responders and the distance between the Query Sender and the query area. $Cost_F = N \times L$. (N: number of Query Responders, L distance in number of hops) In case that the number of Query Responders is fixed, the Forwarding cost is linearly proportional to the distance.

2) The Forwarding cost of proxy and fully distributed strategies increase slowly. The increasing is due to the Forwarding of aggregated data. With aggregates suppression in LBAG, limited number of aggregated data are sent to the Query Sender. Thus the increasing of length does not influence the Forwarding cost of those two strategies.

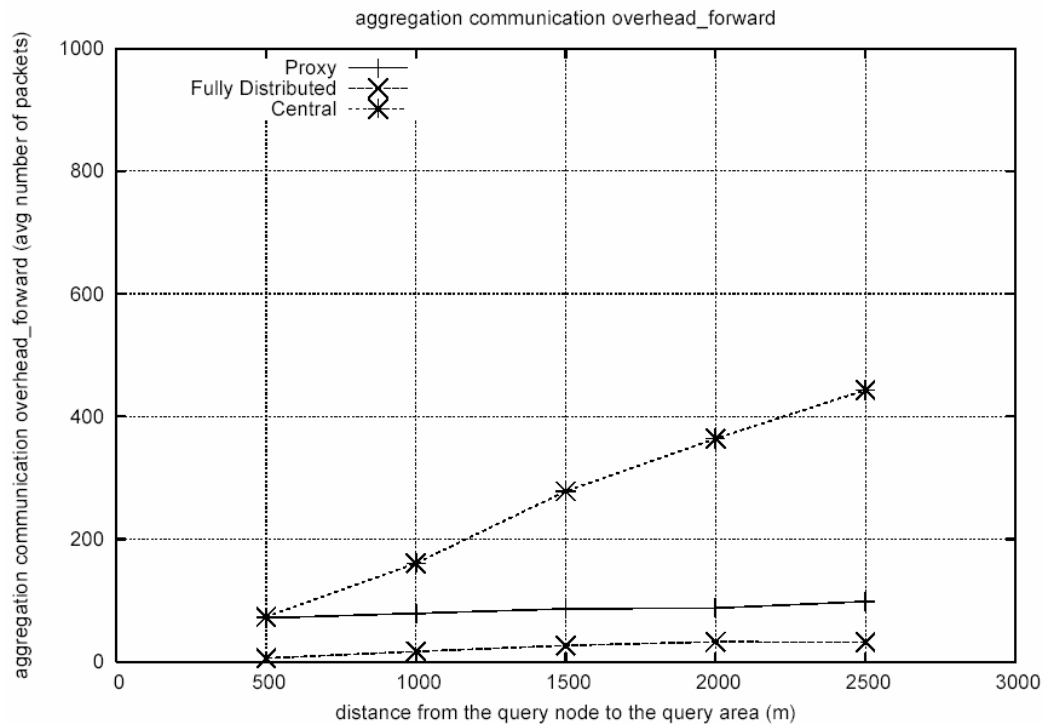


Figure 7-9 Communication Overhead from Forwarding

From the above observation, we get the conclusion that the fully distributed strategy has the minimum Forwarding cost. Centralized strategy has linearly increasing Forwarding cost. Location based proxy strategy has a low Forwarding cost.

7.2.3. Communication Overhead from Broadcast

Figure 7-10 shows the communication overhead from Broadcast for data aggregation with different strategies. We noticed that:

1) Broadcast costs for all three strategies do not increase when the distance increases. This is because the increase of distance only extends the routing path, but not enlarges any broadcast areas.

2) The fully distributed strategy has a high broadcast cost. Location based proxy and centralized strategy has nearly the same broadcast cost. For fully distributed strategy, every response message needs to be rebroadcasted in its query area. Therefore the number of broadcast is expected to be huge. The result that is showed here is already from the improved broadcast algorithm. When we use plain flooding during the early simulations, the number of broadcast is over 2000! For centralized strategy, the broadcast cost comes from the Broadcast Query Message in query area and the Broadcast of response messages in the sender area. For proxy strategy, the broadcast cost comes from the broadcast of query message at the query area and the broadcast of response messages at the aggregation area.

2) Broadcast cost for centralized strategy is fluctuant. After the analysis of raw results, it is explainable. First, the Broadcast cost of centralized strategy is much higher than proxy strategy at the point for 1500m. This is because the node density near 1500m is higher than the density in the aggregation area. (The Broadcast cost is proportional to the number of nodes in the broadcast area) The Broadcast cost in 2000m and 2500m are less than then the cost in 500m and 100m. That is due to the reason that when the distance increases, less response messages can reach the Sender Area, thus triggers less broadcast.

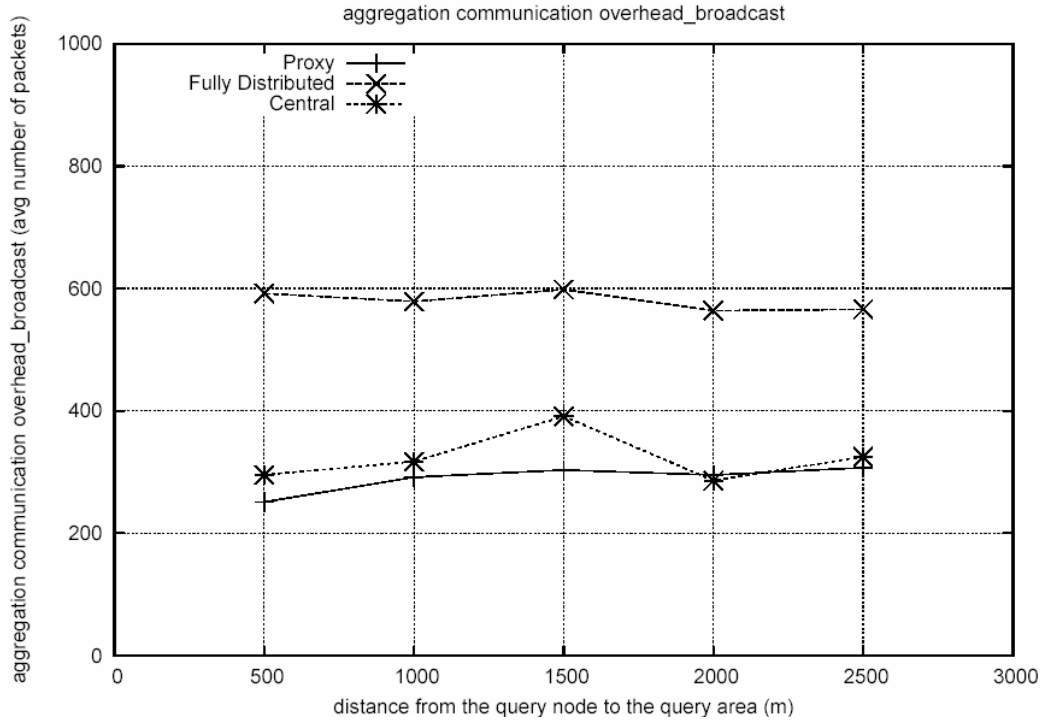


Figure 7-10 Communication cost from Broadcast

From the above observation, we get the conclusion that 1) Broadcast cost is proportional to the size of the area where the aggregation is done. 2) Fully distributed strategy has the biggest aggregation area, thus it has the highest Broadcast cost. Centralized and proxy based have much less cost. 3) Broadcast cost is proportional to the density of nodes.

7.2.4. Overall Communication Cost

The overall communication cost is the combination of Broadcast cost and Forwarding cost for data aggregation. Figure 7-11 shows the overall communication cost for query operations with different strategies. From it we noticed that:

- 1) The location based proxy strategy has a slow increasing and low overall cost. The fully distributed strategy has an almost constant but higher overall cost. And the centralized strategy has a fast increasing overall cost, though it starts with low when the Query Sender is located near the query area. It increases very fast strategy increase fast. When the Query Sender is located far away from the query area, then the overall cost become larger than that of the fully distributed strategy.
- 2) The overall costs of proxy and fully distributed strategies are not influenced by the increase of distance. But the overall cost of centralized is hard to predict precisely, because it is influence by two factors: 1) the distance 2) packet loss on the way. The first factor will increase the overall cost, but the second will decrease it. The final value is actually a compromise. However, both factors are not negative to the performance. The first one increase the communication cost, while the second one decrease the completeness of the aggregated data.

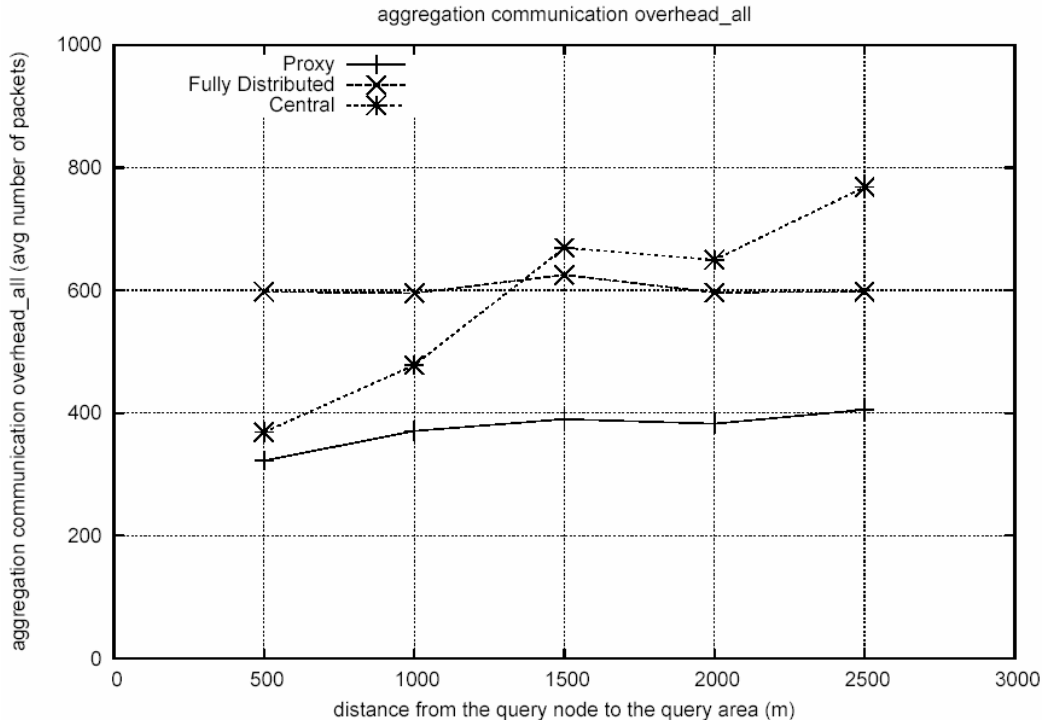


Figure 7-11 Overall Communication Cost

Therefore we can get the conclusion that the proxy strategy has the lowest and slow increasing communication overhead, and the fully distributed strategy has almost constant but higher overhead. The centralized strategy has an increasing overall cost, when the distance between query sender and query area increase.

7.2.5. Summary of the Simulation Result

From the above analysis of simulation results, we found that LBAG is able to do data aggregation in vehicle networks with high mobility. The context of geographic information is used in data aggregation. Apart from it, our implementation- LBAG is able to emulate behaviors of different aggregation strategies. The simulation results validated our conception design. With location based proxy strategy, it is able to get aggregated data with a relative high completeness and low overall communication cost. In addition, we also noticed that the centralized and fully distributed strategies have their own uniqueness. The performance of the centralized aggregation is as good as that of the proxy strategy, when the distance is short. With fully distributed strategy, though with relative high overall cost, completeness of the aggregated data is the highest

8. Conclusion

The objective of this thesis is to develop a data aggregation protocol for vehicular networks. The intention of the protocol is to aggregate raw data from individual nodes into an aggregated result, which can reflect network properties or answer a query from a node. There are several data aggregation protocols proposed for MANETs. But the strategies they are based on are not applicable for vehicular networks, because of its high mobility and network scalability.

Therefore, a new aggregation strategy, which relies on a hierarchy of locations, is proposed. We name it location based proxy strategy. The conception of “Aggregation Area” is very important to this strategy. The task of collecting data and doing aggregation is not bound to nodes with certain ID any more, instead, is bound to nodes within certain area. All nodes inside the aggregation area take up the responsibility of “aggregator”, which means a node that does data aggregation for a query operation. “Aggregation Area(s)”, together with “Query Area” and “Query Sender Area” construct a location-based hierarchy. The advantage of relying on this hierarchy is that this tree is not sensitive to high mobility of nodes.

Based on this new strategy, we developed protocol: Location Based Aggregation (LBAG). LBAG is able to get aggregated data, under the requests from higher-level applications. The API it provides to the applications is generic, which allows SQL-style definition of queries. With the help of spatial model, LBAG supports data aggregation with semantic meanings of geographic areas.

LBAG uses several methods to deal with the mobility problem of vehicular networks. First, it is based on location based proxy strategy. Secondly, it uses geocast ad hoc routing, which is very effective in mobile ad hoc networks. By fine adjustment of several parameters, geocast becomes more reliable in high mobility environments.

In order to make the protocol scalable, LBAG make improvements in two directions. One direction is to reduce communication cost for each data aggregation operation. The other direction is to balance the communication traffic load in the whole network. For the first direction, LBAG sets aggregation areas, so that data from individual nodes do not need long forwarding. In addition, LBAG suppresses duplicated aggregated data from transmitting. The underlying geocast routing protocol can further reduce the cost by improve its forwarding and broadcast. For the other direction, location base proxy strategy allows the redistribution of traffic loads in the location-based hierarchy. LBAG allows the flexible settings of aggregation areas, so that each query sender is able to distribute the traffic load according to the network states.

To the problem of unreliable data transmission, LBAG adopts some methods to enhance the correctness of performance. Limited number of duplications of aggregated data is introduced to achieve that goal.

During the development of LBAG, we find that our implementation is able emulate the behavior of different aggregation strategies, such as the centralized and fully distributed aggregation. Hence,

LBAG can work as a framework, which allows different kind of aggregation strategies. This flexibility is actually very important to the data aggregation in vehicular networks. Because our mathematical study reveals that each aggregation strategy has its cons and pros. They have advantages in certain scenarios. Now with such a data aggregation framework, LBAG can adapt its performance to the real scenario flexibly.

In Chapter 6, the protocol for LBAG is implemented for ns-2 simulator. Related modules of LBAG are also implemented, including GEOCAST, Spatial Model and Application. Chapter 7 shows the simulation scenario and the simulation results. In the simulation, the data aggregation of LBAG is validated. Aggregated results with semantic meaning can be obtained. In addition, we noticed the great impact from underlying MAC and routing layers on the performance of LBAG. With some optimizations on the GEOCAST routing layer protocol, the overall performance is enhanced greatly. We then do sets of simulations to test the behavior of three data aggregation strategies: centralized, fully distributed and Location Based Aggregation. Instead of implementing protocols for centralized and fully distributed aggregation separately, we use LBAG to emulate their behaviors. Thus we got the result of different strategies with three sets of parameter settings. The simulation results show that location based proxy strategy can use fewer overall communication cost to achieve satisfying aggregation results.

The testing of the scalability is not validated by simulation yet. It can be a good topic for future work. Apart from it, lots of work can be done. Routing protocol needs improvement to enhance the overall performance. The performance of the protocol in different density of nodes should be compared.

9. Bibliography:

- [ZGE03] Y. Zhao and R. Govindan and D. Estrin, **Computing Aggregates for Monitoring Wireless Sensor Networks** *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications (SNPA'03)*, May 2003.
- [MSFC02] S. Madden and R. Szewczyk and M. J. Franklin and D. Culler, **Supporting Aggregate Queries Over Ad-Hoc Wireless Sensor Networks**, *Proceedings of the 4th IEEE Workshop on Mobile Computing and Systems Applications (WMCSA)*, June, 2002.
- [MFHH02] S. Madden and M. J. Franklin and J. M. Hellerstein and W. Hong, **TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks** *Proceedings of the Fifth Annual Symposium on Operating Systems Design and Implementation (OSDI)*, December, 2002.
- [IEGH02] Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann, **Impact of network density on data aggregation in wireless sensor networks**, *C. Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02)*, July, 2002.
- [GRB01] I. Gupta and R.V. Renesse and K.P. Birman, **Scalable Fault-tolerant Aggregation in Large Process Groups**, *Proceedings of The International Conference on Dependable Systems and Networks (DSN01)*, July, 2001.
- [LRS02] S. Lindsey and C. Raghavendra and M. Sivalingam, **Data Gathering Algorithms in Sensor Networks Using Energy Metrics**, *IEEE Transactions on Parallel and Distributed Systems, Vol. 13, Nr. 9*, 2002.
- [KDN02] K. Kalpakis and K. Dasgupta and P. Namjoshi, **Maximum Lifetime Data Gathering and Aggregation in Wireless Sensor Networks**, *Proceedings of IEEE Networks'02 Conference*, 2002.
- [IGE00] Intanagonwiwat and R. Govindan and Deborah Estrin, **Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks**, *C. Proc. ACM Mobicom*, August, 2000.
- [ML02] Markus Legner, **Map-based geographic forwarding in vehicular networks**, *Diplomarbeit-Nr.1994, IPVS, Universität Stuttgart*
- [LTLS00] WH. Liao and YC Tseng and KL. Lo and JP. Sheu **GeoGRID: A Geocasting Protocol for Mobile Ad Hoc Networks Based on**

GRID, *J.Internet Technology*, vol.1, no.2 2000, pp.23-32

- [JC02] X. Jiang, T. Camp, **A Review of Geocasting Protocols for a Mobile Ad Hoc Network**, *Grace Hopper Celebration (GHC)*, 2002,
- [KK00] B.Karp and H.T.Kung, **Greedy perimeter stateless routing for wireless networks**, *In Proc. of the 6th Annual ACM/IEEE Int. Conf. on Mobile Computing and Networking (MobiCom 2000) pages 243-154, Boston, MA,USA; August 2000.*
- [THRC03] J. Tian, L. Han, K. Rothermel, and C. Cseh. **Spatially Aware Packet Routing for Mobile Ad Hoc Inter-Vehicle Radio Networks**, *to appear in the IEEE 6th Inter. Conference on Intelligent Transportation Systems (ITSC), Shanghai, China, October 12-15, 2003*
- [BH00] L.Briesemeister and G.Hommel, **Overcoming fragmentation in mobile ad hoc networks**, *Journal of Communications and networks*, 2(3): 182-187,September 2000
- [THRC] J. Tian, L. Han, K. Rothermel, and C. Cseh. **Spatially Aware Packet Routing for Mobile Ad Hoc Inter-Vehicle Radio Networks**, *to appear in the IEEE 6th Inter. Conference on Intelligent Transportation Systems (ITSC), Shanghai, China, October 12-15, 2003*
- [LXZ02] D.Lun Lee, J.Xu, B.Zheng. **Data Management in Location-Dependent Information Services**, *IEEE Pervasive Computing*, 1(3):65--72, 2002.
- [FM] W.Franz, C. Maihöfer, **Geographical Addressing and Forwarding in FleeNet**
- [KV98] Y.-B. Ko and N. H. Vaidya. **Geocasting in mobile ad hoc networks: Location-based multicast algorithms**. *Technical Report TR-98-018, Texas A&M University, September 1998.*
- [SENSORNETS] <http://www.research.rutgers.edu/~mini/sensornetworks.html>
- [BGL] <http://www.boost.org/libs/graph/doc/>
- [NS2] <http://www.isi.edu/nsnam/ns/>

Erklärung

Ich versichere, dass ich diese Arbeit selbständig verfasst und nur die angegebenen Hilfsmittel verwendet habe.

Chi Chen