Report Number: WUCSE-2003-55

2003-07-23

Spatiotemporal Multicast and Partitionable Group Membership Service

Qingfeng Huang

The recent advent of wireless mobile ad hoc networks and sensor networks creates many opportunities and challenges. This thesis explores some of them. In light of new application requirements in such environments, it proposes a new multicast paradigm called spatiotemporal multicast for supporting ad hoc network applications which require both spatial and temporal coordination. With a focus on a special case of spatiotemporal multicast, called mobicast, this work proposes several novel protocols and analyzes their performances. This dissertation also investigates implications of mobility on the classical group membership problem in distributed computing, proposes a new specification for a partitionable... **Read complete abstract on page 2.**

Spatiotemporal Multicast and Partitionable Group Membership Service

Qingfeng Huang

Complete Abstract:

The recent advent of wireless mobile ad hoc networks and sensor networks creates many opportunities and challenges. This thesis explores some of them. In light of new application requirements in such environments, it proposes a new multicast paradigm called spatiotemporal multicast for supporting ad hoc network applications which require both spatial and temporal coordination. With a focus on a special case of spatiotemporal multicast, called mobicast, this work proposes several novel protocols and analyzes their performances. This dissertation also investigates implications of mobility on the classical group membership problem in distributed computing, proposes a new specification for a partitionable group membership service catering to applications on wireless mobile ad hoc networks, and provides a mobility-aware algorithm and middleware for this service. The results of this work bring new insights into the design and analysis of spatiotemporal communication protocols and fault-tolerant computing in wireless mobile ad hoc networks.

Department of Computer Science & Engineering

Washington University in St. Louis
School of Engineering & Applied Science

2003-55

Spatiotemporal Multicast and Partitionable Group Membership Service - Doctoral Dissertation, August 2003

Authors: Huang, Qingfeng

July 23, 2003

Abstract: The recent advent of wireless mobile ad hoc networks and sensor networks creates many opportunities and challenges. This thesis explores some of them. In light of new application requirements in such environments, it proposes a new multicast paradigm called spatiotemporal multicast for supporting ad hoc network applications which require both spatial and temporal coordination. With a focus on a special case of spatiotemporal multicast, called mobicast, this work proposes several novel protocols and analyzes their performances. This dissertation also investigates implications of mobility on the classical group membership problem in distributed computing, proposes a new specification for a partitionable group membership service catering to applications on wireless mobile ad hoc networks, and provides a mobility-aware algorithm and middleware for this service. The results of this work bring new insights into the design and analysis of spatiotemporal communication protocols and

WASHINGTON UNIVERSITY

SEVER INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SPATIOTEMPORAL MULTICAST AND PARTITIONABLE GROUP MEMBERSHIP SERVICE

by

Qingfeng Huang

Prepared under the direction of Professors Gruia-Catalin Roman and Chenyang Lu

---

A dissertation presented to the Sever Institute of
Washington University in partial fulfillment
of the requirements for the degree of

Doctor of Science

August, 2003

Saint Louis, Missouri

WASHINGTON UNIVERSITY
SEVER INSTITUTE OF TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ABSTRACT

SPATIOTEMPORAL MULTICAST AND PARTITIONABLE GROUP MEMBERSHIP SERVICE

by Qingfeng Huang

ADVISOR: Professors Gruia-Catalin Roman and Chenyang Lu

August, 2003
Saint Louis, Missouri

The recent advent of wireless mobile ad hoc networks and sensor networks creates many opportunities and challenges. This thesis explores some of them. In light of new application requirements in such environments, it proposes a new multicast paradigm called *spatiotemporal multicast* for supporting ad hoc network applications which require both spatial and temporal coordination. With a focus on a special case of spatiotemporal multicast, called *mobicast*, this work proposes several novel protocols and analyzes their performances. This dissertation also investigates implications of mobility on the classical group membership problem in distributed computing, proposes a new specification for a *partitionable group membership service* catering to applications on wireless mobile ad hoc networks, and provides a mobility-aware algorithm and middleware for this service. The results of this work bring new insights into the design and analysis of spatiotemporal communication protocols and fault-tolerant computing in wireless mobile ad hoc networks.

to my parents

Contents

List of Tables

List of Figures

Acknowledgments

I am grateful to many who have helped me in completing this work. First and foremost, I want to thank my advisor Dr. Gruia-Catalin Roman for his help and encouragement, for the fruitful discussions and collaborations on many research topics in and out of this dissertation, and for never stop correcting my English.

I would also like to thank Chenyang Lu who has been involved in this project since last Fall and co-advised the research on spatiotemporal multicast. I would also like to thank other members of my thesis committee, Weixiong Zhang, Shirley Dyke and Chris Gill for their constructive input. In particular, I thank Chris for providing many excellent editorial suggestions.

I also wish to thank Subhash Suri for introducing me to the exciting field of computer science, and Ron Miller for inviting me to the Ford Research Lab for two summer internships. Some of the seed ideas of my research grew out of my experiences at FRL. I thank Myrna Harbison and Jean Grothe for taking care of miscellaneous matters during the years, for making the computer science office a very warm and family-like environment to walk into.

I owe a lot to Christine Julien, Chien-Liang Fok, Radu Handorean, Jamie Payton, Rohen Sen and Ali Hazemi for tirelessly proofreading my papers over the years, for sitting through and critiquing the "dry runs". In particular, I thank Chien-Liang for proofreading the whole dissertation. I thank Christine and Ali for their collaboration on the group membership problem at different stages.

I also thank Yan Zhou, Sherlia Shi, Anshul Kantawala, Delbert Hart, Min Wang and Qi Zhang for making my life as a graduate student in computer science more exciting in various ways, including the comradeship and being great opponents in multi-player games. Radu is also among the gang.

Last, but not the least, I thank my parents and my brothers for always being there for me. Their love, patience, understanding and support make my pursuit in scientific research possible.

Qingfeng Huang

Washington University in Saint Louis
August 2003

Chapter 1

Introduction

Computers are rapidly becoming indispensable parts of our social fabric. For instance, computers are being used more and more for solving problems, exchanging information, preparing papers, giving presentations and carrying out business transactions. Recent advances in the cost reduction and minimization of computing devices, wireless communication technologies and sensors have clearly facilitated the spread of computers into our everyday lives and beyond. For instance, calendars, calculators, email applications, infrared and bluetooth communication modules, and even Java virtual machines are available in cell phones that are being used by people on the move. IBM is developing watches with embedded Linux operating systems. Fossil is planning to release its Palm OS watches. Wireless sensor networks are being tested and deployed for environmental and habitat monitoring tasks. Mobile ad hoc networks[1] are being developed and tested for tasks such as emergence response, disaster relief, roadway safety, batter field communication because of its advantage of low cost, easy of deployment and infrastructure independence. The mobility of computers with ubiquitous sensing systems will assist our interaction with the world in a more untethered and natural way. They already have and will further empower us with (seemly unbounded) extensions of our natural capabilities.

The availability of positioning systems together with the pervasiveness of computing devices including sensors and actuators creates an even more interesting picture. Imagine you have landed in an unfamiliar future Caribbean town. Your personal digital assistant (e.g., in your watch or badge or cell-phone or various futuristic form you might

[1] Ad hoc networks refers to the types of networks formed spontaneously by a group of autonomous computers via wireless links for communication tasks. In ad hoc networks everyone serves as a router for others, i.e., everyone is an endpoint and a router at the same time. Examples of such networks include those formed by vehicles communicating with each other on the road, an area of wireless sensors communicating with each other for monitoring tasks. Ad hoc networks are often "unplanned" and often have more dynamic topologies and yet less power and fewer computation resources than infrastructure-based networks.

imagine) senses your current location and quickly pools information from area databases and various tourist kiosks and makes local information of your interest, such as best restaurants and your preferred seafood bars nearby, available to you. It also talks to the sensors close by and tells you that the radiation level in the sunlight was higher than normal and that you'd better put on some protective suntan lotion. As you drive along the scenic roadway, your car adjusts its speed automatically as it continuously talks to vehicles nearby and exchanges precise location and velocity information with them, and coordinates the traffic. As a result, you have more time and attention to spare for enjoying the wonderful views along the way. One can see that computers with location awareness bring many new opportunities.

As always, the advance of science and technology creates new opportunities and brings new challenges at the same time. This dissertation envisions and focuses on a few opportunities brought by the pervasiveness of computers, wireless communications and positioning systems, proposes novel solutions, and addresses the key challenges. It identifies key challenges under the technical terms (scopes) of "spatiotemporal multicast" in ad hoc networks and "partitionable group membership" in mobile ad hoc networks, and explores respective location-aware and mobility-aware solutions.

1.1 Paradigms of Multicast

In the real world, oftentimes a piece of information realizes its highest intrinsic value when reaches the recipients at the right time and place. For instance, a tornado warning is most valuable to you only if you are in the potentially affected area and you received the information some time before the tornado hits. The warning information is of no or little use to you if you are not in the area or you received too late, or too early (e.g, you might have travel plan to move out of the area before the tornado hits anyway.)

This is also is true for many information dissemination scenarios in mobile ad hoc network or sensor network applications. Consider an intruder tracking application in an ad hoc sensor network. When a group of sensors detects some moving object and is able to estimate its velocity, one sensor can be elected to send an "intruder detected" message to inform the sensor nodes "down the road", i.e., in a geographic envelope approximating the object's expected movement path, to get ready, e.g., to wake up if they are in power-saving mode, and prearm for the tracking task. Note that this "intruder coming" information has two unique spatial and temporal properties with respect to its intended application. First, it is most useful for the sensor nodes that are close to the object's expected movement path and within certain distance to the current location of the object. Second, it is

desirable to be delivered within a specific period of time: early enough for the sensors to wake up before the object gets to where they are, and close enough to the object's arrival time so as to avoid unnecessary dissipation of energy. In other words, it has a preferred time window of delivery for every respective spatial destination.

Consider another scenario where an ambulance on the road tries to inform vehicles down the road that it is coming and ask the vehicles to be alert and to yield the way. Currently, this is achieved by the ambulance using a siren which can be heard within a few blocks. We envision a futuristic scenario where an electronic siren information is propagated in a multi-hop vehicle network, especially for auto-piloting vehicles. The siren information (e.g. "ambulance coming from direction x at speed y with current location z, etc") needs and only needs to be delivered to the vehicles at some distance down the road relative to the ambulance that might interfere with the ambulance's motion, as is shown in Figure 1.1. Furthermore, the delivery needs to be done a few seconds before

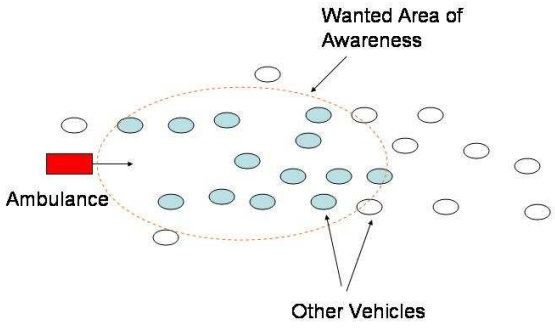


Figure 1.1: Information Dissemination Needs of An Ambulance

potential interference can take place so the vehicles have enough time to react to the information. As the ambulance moves, the relative geographic area of delivery changes accordingly. Note also that the delivery of siren information should not be delivery too early before the potential time of interference. Because the vehicles might change their route well before the potential interference time, the likelihood of the siren information to become invalid is high if it is delivered too early.

Similar spatial and temporal properties can be found in other information dissemination applications such as a forest sensor network sending information about the size of a fire and its spreading velocity to fire-fighters in some dangerous area, or a biological sensor network sending bio-agent alarm information to personnel within a certain area.

In all such cases, the type of information being disseminated is useful only to recipients in a specific geographic area at a specific time, and the information is preferred NOT to be disseminated as soon as possible in space, but rather to be disseminated in a controlled manner, e.g., in constant velocity. The fundamental reason is that the information being disseminated has a correlated space-time value. For this type of information, its arrival at a destination "on time" usually exhibits higher value than "earlier" or "later." This distinguishes itself from most other information that exhibits "sooner is better" delivery semantics, and demands new strategies for dissemination.

Delivery of information to an area often involves multiple recipients. Multicast is a well-known communication abstraction for information delivery from one source to multiple recipients, and is widely regarded and used as a fundamental building block for many distributed computation applications and systems. We first examine existing multicast mechanisms and discuss their applicability to the previous scenarios.

1.1.1 IP Multicast and Geocast

There were two main multicast paradigms: IP multicast [52] and geographical multicast (geocast) [128]. They differ in their recipient identification and addressing schemes. IP multicast identifies the recipients by their subscription to a common multicast IP address. A message sent to a multicast IP address such as 238.5.1.8 is expected to be delivered to all subscribers of this address. Figure 1.2(a) shows an IP multicast example, in which the sender is S and other solid circles represents expected recipients. The IPv4 networking standard defines and reserves the "Class D" addresses (ranging from 224.0.0.0 to 239.255.255.255) for multicast.

Geocast identifies the set of recipients by the geographical locations of the relevant parties. In other words, the set of recipients is addressed in geocast by a geographical area, often described by a polygon. A message geocast to a area is expected to be delivered to all parties in the area. Figure 1.2(b) shows an example where the source node S wants to geocast a message to all nodes in one of the southeast rectangles. A geocast protocol may use one path or multiple paths from the source node to the rectangle for forwarding the message. Once the message arrives in the rectangle, it is flooded to all nodes in the rectangle [99]. Geocast is a more natural and economic alternative for building geographic service applications than conventional IP address based multicast addressing and routing.

While different in style and approach, both conventional IP multicast and geocast assume the same information delivery semantics in the temporal domain, i.e., the information is expected to be delivered as soon as possible. This delivery semantics turns out not to be the best choice for many applications targeted to resource constrained ad hoc

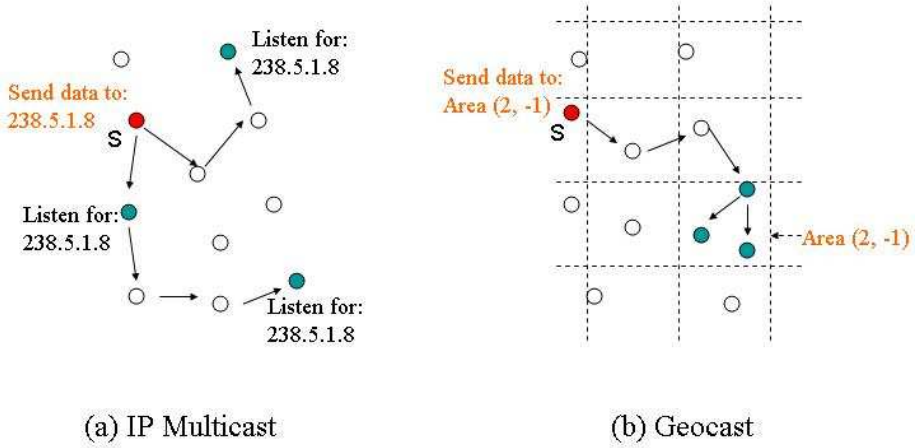


Figure 1.2: IP Multicast and Geocast

networks, and for disseminating information with temporal value. For instance, in the object tracking example above, an early delivery of the wake-up call is likely to waste the precious power resources and reduce system's affectiveness over time.

The Contention Between Slack and Overhead

Using existing multicast mechanisms to achieve the goal of information delivery to the right place at the right time is indirect and exhibits significant overhead. The fundamental reason is that the existing protocols all have "as-soon-as-possible" delivery semantics.

In this section, we will use geocast to construct a solution for the previous ambulance siren information delivery problem, and illustrate a few inherent issues relating to the solution quality. Geocast is a reasonable match in the existing communication mechanism arsenal for supporting the ambulance siren application example.

Assume that the ambulance uses geocast, as shown in Figure 1.3. For simplicity, we use a rectangular delivery area in the example. Let the rectangle between points A and C be the initial geocast area for the siren information when the ambulance at point A. As it moves forward, the ambulance's desired geocast area moves forward as well. So the ambulance should periodically re-geocast the siren information. Clearly, the period

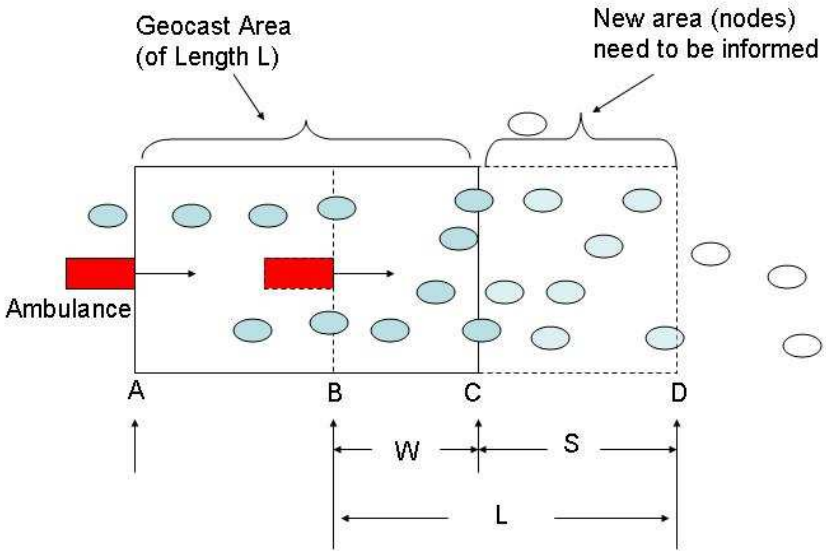


Figure 1.3: Ambulance Geocast Scenario

should be small enough such that there is at least one re-geocast between points A and C. Otherwise, some of the vehicles on the road will miss the siren information. Furthermore, for the siren system to work, one may require the vehicles to receive the siren information by a certain time, t_a, before the ambulance reaches them. This advanced warning is necessary for the vehicles to have enough time to take appropriate evasive actions and serves the purpose of a siren. This leads to a necessary re-geocast, say at point B. Re-geocasting the siren information at point B means that some of the nodes between point B and point C receive the information at least twice and they also need to act as routers for nodes beyond point C. The area between B and C increases proportionally with the required advanced delivery time and the speed of the ambulance, and so does the number of nodes that receive the redundant information.

Now we formulate the issues more precisely. Let the width of each geocast destination (area) be L. Regardless of the actual forwarding scheme used, the number of radio transmissions needed for all nodes in the geocast area to receive the message is roughly[2] proportional to L.

[2] "Roughly" in the sense that we omit the discreteness effect which occurs when the size of the area is close to the radio range. Note also that we assume a constant height of the geocast area in the analysis.

Let the ambulance speed be v_a. Let the desired advance warning time for the vehicles be t_a. The distance W between points B and C must be no less than $v_a t_a$. To inform the nodes beyond point C (the lightly shaded ones), the ambulance geocasts a message to them using some of the nodes between B and C as intermediate forwarding routers. Regardless of the actual forwarding scheme used, the number of forwarding nodes between B and C is proportional to the length W. In turn, the number of radio transmissions generated for this forwarding between B and C, M_W, is proportional to W:

$$M_W \propto W \sim v_a t_a \tag{1.1}$$

Note that the number of these radio transmissions between B and C represents the extra message overhead of using the geocast protocol since the nodes between B and C have received and forwarded the message before (i.e., in the previous geocast initiated at A.).

In the above scheme, the ambulance re-geocasts at B for the goal of having the vehicles close to C on the right to receive the message t_a time in advance. This leads to the following consequence: most of the nodes between C and D receive the message at more than t_a time in advance since geocast delivers message in an as soon as possible fashion. Let v_m be maximum message propagation speed in the network. Then the following quantity, called "average slack time," measures the average earliness of the nodes between C and D on receiving the message:

$$\overline{t_s} = \frac{1}{2}(\frac{S}{v_a} - \frac{S}{v_p}) = \frac{1}{2}(L - W)(\frac{1}{v_a} - \frac{1}{v_p}) \tag{1.2}$$

Usually smaller average slack time is more desirable, like in the cases of most real-time systems. Smaller average slack time in our example entails more nodes receiving the message just in time, resulting in a system that is more flexible and robust against uncertainties and changes over time.

The average slack time decreases when W increases, while the message overhead increases with W. As such, we observe a fundamental contention in the geocast based approach: one can not reduce the message overhead and the average slack time simultaneously. One reason behind this contention is that the geocast protocol does not explicitly concern the temporal domain of message delivery. That is, geocast only addresses application's need for the "right place" perspective of information delivery, but does not address the "right time" perspective. Inevitably, exact steps and methods are necessary when using geocast protocols for constructing "just-in-time" type solutions, which in turn leads to extra overhead as we see in the previous discussion.

We also observe that, if the ambulance did not change its speed and direction during its course between A and B, a more economic alternative approach to this problem is letting some nodes in the front of the geocast area forward the message at an appropriate time rather than requiring that the ambulance re-issue the geocast periodically. This approach entails a self-sustained mobile message wave on the network and leads to our discovery of a new multicast paradigm which we describe next.

1.1.2 Spatiotemporal Multicast

In this dissertation we propose a new multicast paradigm, called "spatiotemporal multicast," catering to the class of applications that need to disseminate their multicast messages to the "right-place" at the "right-time." Spatiotemporal multicast identifies the set of recipients by a space and time constraint characterized by an area function $Z[t]$ and a time period $[T_s, T_s + T]$. In other words, a spatiotemporal multicast session can be specified by a four-tuple

$$\langle m, Z[t], T_s, T \rangle \tag{1.3}$$

where m is the multicast message, $Z[t]$ describes the expected area of message delivery at time t, T_s and T are the sending time and duration of the multicast session, respectively.

At each instant of time $t \in [T_s, T_s+T]$, $Z[t]$ describes a geographical area, which we call "delivery zone" henceforth. A message sent to the delivery zone $Z[t]$ is expected to be delivered to a node at time t if the node is in the area of $Z[t]$ and has not yet received the message. As the delivery zone $Z[t]$ evolves over time, the set of recipients for m changes as well. Note that in geocast [128, 88, 99] the delivery area Z is fixed for each multicast session, and there is no explicit specification of when the session terminates. The key characteristic of the spatiotemporal multicast service is its ability to give applications explicit control over both the spatial and temporal perspectives of multicast information delivery. Clearly, geographical multicast can be viewed as a special case of spatiotemporal multicast.

Figure 1.4 shows two examples of spatiotemporal multicast with different kinds of delivery zones. Figure 1.4(a) depicts a rectangle-shaped zone (shaded) that moves from the source located at the bottom of the figure to the top. As the delivery zone moves, some nodes enter the zone while others leave the zone. The delivery specification of spatiotemporal multicast may require that a node be delivered the message m at the time the delivery zone reaches the node. Note that the shape and evolving behavior of a delivery zone are defined by users (for their spatiotemporal delivery requirement of information

m). A spatiotemporal multicast protocol then needs to achieve this spatiotemporal delivery requirements efficiently in various network topologies. Figure 1.4(b) shows a more general example where the delivery zone assumes an arbitrary shape with both its shape and location evolving over time. This may be the case when the delivery requirements change in response to dynamic context observed in the mobile delivery zone.

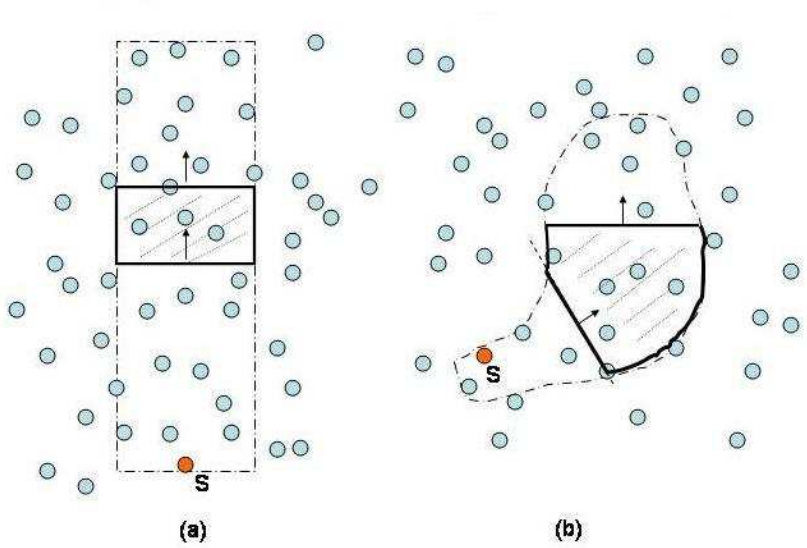


Figure 1.4: Spatiotemporal Multicast Examples

The unique characteristic of spatiotemporal multicast makes it a natural candidate for many space-time-sensitive information dissemination tasks in ad hoc networks. Application developers can easily encode their desired spatiotemporal constraints for information dissemination into the changing behavior of the delivery zone. For instance in the object tracking example, if the sensor group initiates a spatiotemporal multicast for the intruder detected message using a rectangular delivery zone that moves at the estimated velocity of the object (or a little faster), the wake-up just-in-time requirement can be readily satisfied. This is also true for the ambulance example. Using a mobile delivery zone quickly leads to a solution in which only nodes in the forefront of the delivery zone are involved in forwarding the siren information; there is no need for the ambulance to re-issue the siren information unless there is a change in its motion plan, or the network experiences vicious partitions.

Just-in-time information propagation has an additional benefit. It is more flexible in responding to changes than an "as soon as possible" information propagation mechanism. It enables the possibility of undoing an outdated information dissemination process in a simple way. For instance, again in the object tracking scenario, when the object has a relatively big change in its moving speed and direction, one can stop the sensors on the

old expected path from being waken up by sending a faster message to stop the previous information propagation. While in an "as soon as possible" delivery mechanism, this is relatively hard to do without a specialized and prioritized control message channel. This natural ability to allow flexible responses to dynamic changes is very desirable in our envisioned applications.

1.1.3 Some more comparisons

In a sense, IP multicast by its logical addressing scheme is concerned with whom you are. Geocast's spatial addressing scheme emphasizes that the information it is disseminating is more relevant to where you are rather than whom you are. And spatiotemporal multicast emphasizes when you are and where you are. Table 1.1 summarizes the differences between the three multicast paradigms. Note that even though we have only talked

Table 1.1: Three Multicast Paradigms

Multicast paradigm	IP	Geographical	Spatiotemporal
Addressing	Logical	Physical	Physical
Location concern	No	Yes	Yes
Time concern	No	No	Yes
Additional comment	whom	where	where and when

about 2-dimensional cases for geocast and spatiotemporal multicast, their extension to 3-dimension cases is straightforward.

Chapters 2, 3, and 4 continue on the topic, propose and investigate various protocols for a special class of spatiotemporal multicast, called mobicast. Next we switch our attention to the second problem this thesis addresses: the group membership problem in mobile ad hoc networks.

1.2 Partitionable Group Membership

Group membership has been an important problem in the area of fault-tolerant distributed computing and has been the subject of extensive investigation [44, 5, 15, 148, 6, 62, 123, 160, 141, 13, 37]. Solving the problem requires the provision of a service that establishes and maintains some kind of agreement over time among participating components about who is currently in the group, despite the presence of failures in the corresponding distributed system. Such a group membership service simplifies the development of many fault-tolerant distributed applications [44] and is widely used for supporting reliable group communications.

We encountered a group membership problem in our attempt to support group computation in ad hoc mobile environments. Peer-to-peer or group cooperation are common scenarios for ad hoc mobile applications. When two or more mobile hosts come together to form a group working on the same problem, it is sometimes essential for all of them to have the same view of the joint computation state when they start working or when some of the members leave the group. One important piece of group state information is membership in the group, i.e., who is and who is not in the working group. Any situation that demands (for legal or technical reasons) the presence of two or more specific entities to carry out a task may impose the need for a consistent membership view. One can envision the futuristic notion of an electronic witness to a contractual transaction or the circumstance in which routine maintenance of commercial aircraft requires secure logging in the presence of an FAA inspector carrying an authorized electronic badge. The need for consistent view among group members also rises in the fledging area of intelligent transportation systems [43, 91]. Driving assistance systems are being developed on top of the mobile wireless ad hoc network for collision warning and avoidance [154, 86, 166]. In such systems, a consistent view about participating vehicles and consistent decisions across the members about acceleration, deceleration, lane change [78], etc., are critical.

To further see why the same view about membership and the order of membership changes are important, consider a computation scenario that involves four hosts, H_a, H_b, H_c, and H_d. Initially H_a and H_b form a group; they agree that if H_c joins the group, they will execute protocol C; if H_d joins the group, they will execute protocol D; if H_c and H_d join at the same time, they will execute protocol E. If H_a and H_b do not observe the same order of joins by H_c and H_d, (e.g., H_a observes H_c joining first, and H_b observes H_d joining first) then the group will result in a protocol execution confusion: H_a will execute protocol C while H_b will execute protocol D. The importance of agreeing on the same view about membership state introduces the need for group membership services, which maintain a consistent view about the membership among group members.

The group membership problem we encountered in the mobile ad hoc environment is new in the sense that it has special requirements that are different from all previously studied ones. It not only requires availability and progress in the presence of network partitions, as most partitionable group membership services do, but also requires consistency when partitions occur, which none of the previous partitionable group membership services support [63, 13, 7, 6, 56, 55, 160]. The reason why the previous partitionable group membership services do not support consistency in the presence of partitions is fundamental; because the assumed system model is asynchronous and distributed, consensus is impossible [65, 33]. Furthermore, network partition in a fixed network is usually

infrequent and short-lived. This makes manual checking a viable option to solve any inconsistencies that might occur during a network partition. Yet in our case, we emphasize the consistency requirement because network partition is a frequent event in ad hoc mobile environments and the cost of "short-term" inconsistency can be very high in mobile computation scenarios. Mobile hosts interact over wide spaces, and inconsistency can propagate indefinitely and cause irreparable damage in mission critical applications. Similar strict consistency has been considered by Cristian and Fetzer [49] for the primary group membership problem in timed-asynchronous systems, but has not been investigated for the partitionable group membership problem.

The goal of our group membership service is not only to create a consistent view of group membership among participants, but also to help users and application programmers avoid the complexities introduced by potential data inconsistencies caused by mobility-induced link failures. Mobility-induced link failures refer to the communication failures caused by mobile units moving out of each other's communication range. The key characteristic of a mobility-induced link failure is that it is unrecoverable and more damaging than link failures in fixed networks. For instance, a message sent while a physical network partition is taking place can be in a dubious state of delivery. That is, one side (sender/receiver) thinks it is "delivered," while the other side (receiver/sender) thinks it is not. To make matters worse, no mechanism exists for either party to disambiguate this situation. The link failures occurring in fixed networks are usually recoverable, in the sense that the communicating peers can always use an acknowledgement-retransmission-based protocol like TCP to ensure a link failure is transient because in a fixed network its is assumed that a failed link will eventually come back. The unrecoverability of mobility-induced link failure can result in permanent data inconsistency and poses a great challenge to mobile application programmers. Our group membership service tries to help programmers in this matter by guaranteeing that communication between group members will not suffer from mobility-induced failure.

The strong requirements of the new group membership service make it impossible to implement in asynchronous systems. To make this problem solvable, we introduce a certain level of synchrony into our system model. We assume the communication service is reliable in each physical network partition and has a bounded message delivery time t_d within the partition. Furthermore, as we mentioned earlier, a message sent at time $t < t_d$ before a physical network partition takes place can be in a dubious state of delivery. We introduce a key concept called *safe distance* to solve this problem. It works by preventing a group message from falling into a pocket of network instability caused by partitioning. We rely on location information to decide when a host within communication range is

admitted to or eliminated from a group. The group membership policy is conservative in nature to ensure that the changes in group membership appear to be atomic, i.e., are serializable transactions. The algorithm accommodates both the merging of groups and the partitioning of one group into multiple disjoint groups.

We have implemented a version of the strong partitionable group membership protocol in Java. It supports LIME [124, 125], a middleware for rapid development of mobile applications. The implementation works properly if the system assumptions of the protocol are met. The validity of the system assumptions and how they might be implemented are discussed later.

Chapter 5 continues with detail on this topic.

1.3 Contributions

This dissertation has contributions in two areas. It is the first to propose and investigate the spatiotemporal multicast paradigm. This contributes to the arsenal of multicast communication technologies. It is also the first to identify the implications of mobility for the group membership problem and to investigate new solutions. This contributes to the areas of software engineering and fault-tolerant distributed computing. More detailed technical contributions include:

- **Specifications** of *spatiotemporal multicast* and *mobicast*, and their performance metrics.
- **Two new graph metrics** for spatiotemporal communication analysis, namely Δ-*compactness* and Γ-*compactness*, capturing the distortions in a network relevant to communication.
- **Novel protocols** for mobicast with performance analysis.
- **Two new geometry notions**, namely *k-cover* of a polygon and *spatial neighborhood* in planar graphs. In spatial communication, K-cover is a useful tool for locating shortest paths with minimal information. Knowledge of the spatial neighborhood in planar graph helps construct efficient geometric routing algorithms.
- **A distributed algorithm** for spatial neighborhood discovery and maintenance.
- **A new specification** of the group membership problem catering to the requirements of applications in wireless mobile ad hoc environments.

- **A mobility-aware protocol** for partitionable group membership service in mobile ad hoc environments.

1.4 Thesis Organization

The dissertation is organized as follows. Chapter 2 explores the formal specification of delivery guarantees of spatiotemporal multicast and proposes optimization metrics for spatiotemporal multicast protocols. It focuses on mobicast and presents a reliable mobicast protocol. It introduces new network compactness metrics and the geometric notion of k-cover for analyzing the protocol. Chapter 3 investigates the compactness properties of random sensor networks. Using the insights gained from this investigation, it explores alternative mobicast strategies and evaluates them via network simulations. Chapter 4 introduces and analyzes the Face-Aware Routing (FAR) protocol for mobicast with the notion of spatial neighborhood in planar graphs. It also features a novel distributed algorithm for spatial neighborhood discovery. Chapter 5 presents the new specification for partitionable group membership service and proposes a corresponding mobility-aware protocol for the service. Chapter 6 summarizes this dissertation and discusses future work.

Chapter 2

Mobicast: Mobile Multicasting Cases and Protocols

This chapter explores the formal specification of delivery guarantees for spatiotemporal multicast and proposes optimization metrics for spatiotemporal multicast protocols. It focuses on a special class of spatiotemporal multicast called "mobicast" and presents a reliable mobicast protocol. It introduces two new notions of network compactness and proves several related theorems useful in analyzing information propagation in wireless ad hoc networks. Another by-product is a notion of "k-cover" for polygons, which is a generalization for ellipse and is useful for locating shortest paths using network compactness values.

2.1 Mobicast

Mobicast is a special class of spatiotemporal multicast that has the following special behavior: its delivery zone is some fixed convex polygon P that translates through a 2-D space at some constant velocity $\vec{v}$, i.e.,

$$Z[t] = P[\vec{r}_0 + \vec{v}(t - T_s)] \tag{2.1}$$

with $P[\vec{r}_0]$ being the polygon centered at $\vec{r}_0$. We call this specific class of spatiotemporal multicast "constant velocity mobile multicast", abbreviated as "mobicast". A mobicast session example is shown in Fig.(2.1) in which the solid rectangular area $Z[3]$ represents the current delivery zone (at time $t = 3$), and the two dashed rectangles $Z[1]$ and $Z[2]$ represent two instances of the same delivery zone at times $t = 1$ and $t = 2$, respectively.

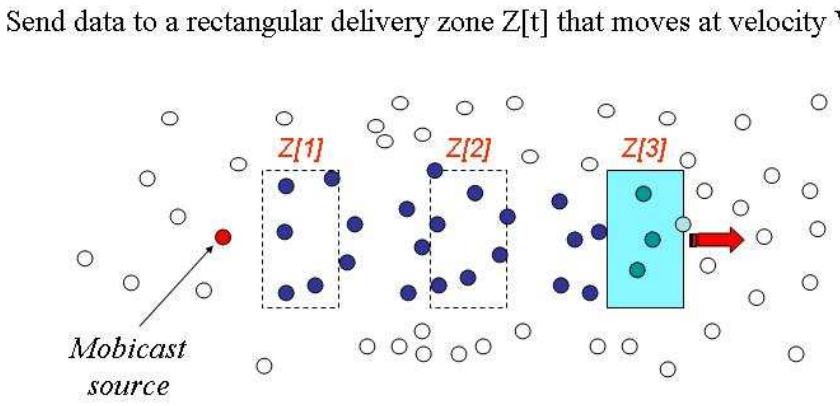


Figure 2.1: Mobicast example: A Moving Rectangular Delivery Zone

Note that there are many other spatiotemporal multicast and mobicast scenarios besides the constant velocity mobicast. For instance, some applications may like the delivery zone to move on a path under a specific speed schedule, or on a path with maximum information gain [113].

As an abstract communication paradigm, mobicast is conceptually independent of underlying communication media. However, its actual implementation has to take the characteristics of the media into account. In this chapter, we consider mobicast on wireless sensor networks with random but static topology.

2.2 Application Examples

Mobicast can be used for sensor network applications such as intruder tracking [28, 113] or information scouting, as shown in Figure 2.2. On the left we have an intruder tracking example. In this example, a set of sensors discovers an enemy tank, and sends an alert message to sensors and actuators (e.g., camera control units) on the intruder's expected path to alert and pre-arm them for better tracking and other operations. This alert message can be sent by a mobicast service using a delivery zone of desired size that moves at a certain distance ahead of the intruder, with a speed approximating that of the intruder's, thus creating an evolving alert "cloud" just in front of it. The right side of

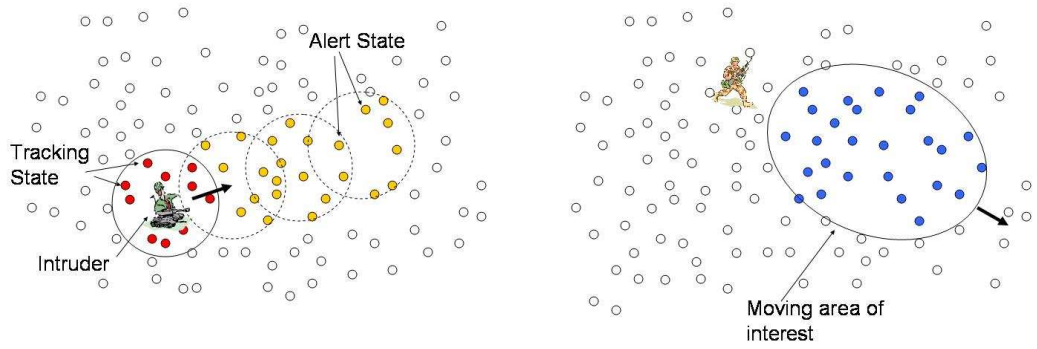


Figure 2.2: Tracking and Scouting Applications

Figure 2.2 depicts an information scouting example. A solider is running to the southeast area. For safety and efficiency, he would like to know the field information ahead of him so as to adjust his actions accordingly. His area of interest changes as he runs. One can see that this is another natural application scenario for mobicast. The solider can send a scouting request to a delivery zone that moves on his path in front of him. Only the sensors that enter the delivery zone (receive the scouting message) will pool their currently sensed information and send aggregated data back to him. The use of mobicast naturally delivers the spatial and temporal locality requirements of information dissemination and gathering exhibited by these applications.

A key observation here is that if the mobile event does not change its motion very often, the system alert or scouting message does not need to be continuously issued. But rather, one can let the message roll on its own according to a motion plan. If the mobile event changes or the old mobicast expires, one can always issue a new mobicast reflecting the new information.

2.3 Spatiotemporal Delivery Guarantees

As we have pointed out earlier, application developers can encode their spatiotemporal information delivery requirements via the delivery zone behavior. The complexity of a mobicast protocol in general depends on the delivery guarantees it is required to achieve. A straight-forward delivery specification may demand that once a node α is in a delivery zone $Z[t]$, it receives the information m immediately. Here we will first define and refine this specification formally, and discuss its feasibility and implications. Let Ω be the set of all nodes in space, let $\vec{r}(j)$ be the location of node j, and let $D_m[j,t]$ denote the fact that j has been delivered the information m at time t. Let the time when the mobicast

is initiated be T_s. This mobicast delivery property can be formally stated as

$$\langle \forall j, t : j \in \Omega \wedge T_s \leq t \leq T_s + T :: \vec{r}(j) \in Z[t] \Longrightarrow D_m[j,t] \rangle^1 \tag{2.2}$$

This statement can be interpreted as "During the mobicast session, all nodes inside zone Z at time t should have information m."

Unfortunately, the delivery property (2.2) is practically impossible to realize in most wireless ad hoc networks. The reasons include:

- First, communication latency is often not negligible in wireless ad hoc networks. This is especially true in wireless sensor networks where sensor nodes might have a sleeping schedule in order to save energy. Note that (2.2) implies instantaneous delivery to all nodes at the initial delivery zone $Z[0]$. If $Z[0]$ contains a node other than the sender node, it is impossible for the node to receive information D instantly when considering the communication latency.

- Second, a wireless ad hoc network may be partitioned. A delivery zone, specified by some geometric property alone, might cover nodes in multiple disjoint network partitions, which renders delivery impossible.

- Third, we did not put any restrictions on the speed of the delivery zone. One can imagine cases where a user-specified delivery zone moves so fast that it exceeds the maximum delivery speed a network can support.

As such, we are forced to weaken the ideal mobicast delivery property in the following practically-minded manner: mobicast satisfies property (2.2) only after some initialization time t_{init}. That is

$$\langle \forall j, t : j \in \Omega \wedge t_{init} < t \leq T :: \vec{r}(j) \in Z[t] \Longrightarrow D[j,t] \rangle \tag{2.3}$$

Thus, each mobicast session has two phases. The first, from time 0 to t_{init}, is an initialization phase in which no delivery guarantee is specified. The second phase, from time t_{init} to T, is a stable phase in which the strong spatiotemporal guarantee is required. We also implicitly assume the speed of delivery zone is smaller than the maximum speed the network can support and the network is not partitioned.

[1] The three-part notation $\langle \mathbf{op}\, quantified\_variable : range :: expression \rangle$ used throughout the text is defined as follows: The variables from *quantified_variables* take on all possible values permitted by *range*. If *range* is missing, the first colon is omitted and the domain of the variables is restricted by context. Each such instantiation of the variables is substituted in *expression* producing a multiset of values to which **op** is applied, yielding the value of the three-part expression.

2.4 Optimization Concerns

Note that specification (2.3) addresses only the functional requirement for mobicast, and does not address any performance optimization perspectives. Yet, performance is an indispensable dimension of protocol design. Here we discuss three optimization dimensions for mobicast protocols.

Note that, since communication latency is a random variable, it is impossible to deliver a message to a node at an exact time. In order to achieve the delivery property (2.3), one has to consider the worst case scenario and schedule the delivery of mobicast messages ahead of time.

Let $t_r(j)$ denote the time at which node j first receives the mobicast message, and let $t_{in}(j)$ be the first instant of time j enters the delivery zone. We call the time difference $t_{in}(j) - t_r(j)$ the "slack time" associated with the message delivery. It measures how early the message is delivered to a node with respect to its requisite deadline (to be at the specific node). Note that specification (2.3) implies that t_{in} is the deadline of message delivery. In general, one would like to maximize the number of expected recipients that meet the delivery deadline. Let Θ be the set of the "delivery zone nodes" that is defined as the set of all nodes that are expected to receive the mobicast message in a mobicast session. Let Ξ be the set of delivery zone nodes that received the mobicast message on or before the deadline, i.e.,

$$\Xi \equiv \{j | (j \in \Theta) \wedge (t_{in}(j) - t_r(j) \geq 0)\} \tag{2.4}$$

One obvious optimization dimension is to make the initialization phase as short as possible. A smaller t_{init} means more nodes will meet the delivery deadline. In general, the length of the initialization time depends on the size of the delivery zone, the network connectivity pattern within the region, and the protocol execution behavior. While a mobicast protocol has no control over the first two factors, it can try to make t_{init} as short as possible by optimizing its execution strategy.

Another optimization concern for any mobicast protocol is to reduce the overall time interval between the reception of a message and its required delivery to the application, i.e., the slack time. Minimizing the average slack time t_{slack} for all nodes that were ever in the delivery zone improves the timeliness of mobicast message delivery, and means less time is spent in "holding" the message before it is needed. A small t_{slack} is also desirable as it potentially leads to less energy consumption and better locality in spatial data aggregation. As such, a mobicast protocol should seek to minimize the average slack

of the delivery zone nodes:

$$\overline{t_{slack}} = \frac{\sum_{j \in \Xi}(t_r(j) - t_{in}(j))}{|\Xi|} \tag{2.5}$$

where $|\Xi|$ denotes the cardinality of the set Ξ. The ideal case for a mobicast protocol involves reducing $\overline{t_{slack}}$ to zero, i.e., a node receives the mobicast message right when it enters the delivery zone. Unfortunately, this may not always be possible due to the randomness of the communication latency.

The third optimization dimension for mobicast is to reduce the total number of retransmissions needed for each mobicast session while delivering the spatial and temporal guarantees. This is common to all broadcast and multicast protocols for ad hoc networks.

2.5 Simple Mobicast Solutions

To help clarify the inherent complexity of mobicast, we present first two simple mobicast protocols that succeed and fail in different ways. In both protocols, mobicast packets are always marked by the sender with a description of the packet delivery zone and a lifetime.

The first simple mobicast protocol is based on flooding. Once a node receives a mobicast call from the application, it floods the mobicast message to the whole network. The rest of the nodes in the network, in addition to participating in the flooding, behave as follows: once a mobicast message is received, they schedule the delivery of the mobicast message to the interested applications at the time when the delivery zone reaches them. If a node finds itself to be never in the delivery zone, it drops the message after having fulfilled its forwarding responsibility and after the message has expired, without delivering the message to application layer. Even though this protocol can achieve the spatiotemporal delivery specification (2.3), it is not desirable in at least two respects. The first is that the global flooding has a large overhead, especially when the cumulative (union) delivery zone area is much smaller than the span of the whole network. The second is that the average slack time of the mobicast reception is too high, especially when the mobicast delivery zone speed is much slower than the maximum information propagation speed on the network.

The second protocol examined here employs a hold-and-forward strategy where only nodes on the path of the delivery zone will participate. For convenience, we call it the "Delivery-Zone Constrained" (DZC) protocol. The DZC protocol exhibits minimal delivery overhead and has good slack time characteristics on "good networks," but is not entirely reliable. A good network is one that is compact and with no large holes.

The compactness of and holes in a network will be formally defined later. For simplicity, Figure 2.3 shows a mobicast example on a one-dimensional network with a rectangular delivery zone moving at a constant velocity. The DZC protocol works as follows. Once

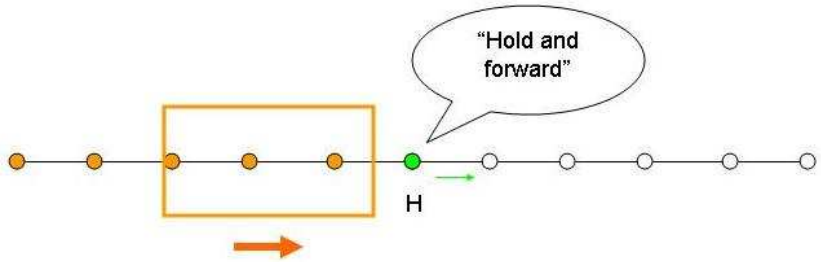


Figure 2.3: Greedy Hold-and-Forward Mobicast Protocol

a node receives a new mobicast packet, it first checks if the packet has expired. If not, the node checks if it is in the current delivery zone for the packet. If it is, the packet is delivered to the application immediately and is forwarded as soon as possible; otherwise, if the node is not in the delivery zone but expects to be in it soon, like node H in Figure 2.3, the packet is held and scheduled for delivery and forwarding at the time the delivery zone reaches the node. In all other cases, nodes will ignore mobicast packets. The hold and forward behavior of nodes in front of the running delivery zone makes the packet delivery and forwarding "just-in-time," and creates a self-sustained mobile message wave. Note that only nodes that find themselves in the delivery zone path will join the forwarding. This keeps the forwarding overhead at a minimum. However, the protocol fails to deliver the mobicast message to delivery zone nodes that are not directly connected to the source through a path fully contained in delivery zone's path. This is illustrated in Figure 2.4. The DZC protocol fails to deliver the mobicast message to node X because the nodes outside of the delivery zone do not participate in the forwarding process.

The drawbacks of the DZC protocol indicate that in order to guarantee mobicast delivery for all delivery zone nodes, some nodes that are never in the delivery zone must still participate in message forwarding. An important question is: how to determine who should participate without knowing the detail of the global network topology? Furthermore, potential holes in the network (as in Figure 2.4) show that two nodes close in physical space can be relatively far away in terms of network hops. This presents a major challenge for the timely delivery of mobicast messages, i.e., a mobicast protocol needs to consider potential propagation latency in physical space due to long underlying network paths in order to achieve timely delivery across the physical space. In the next section,

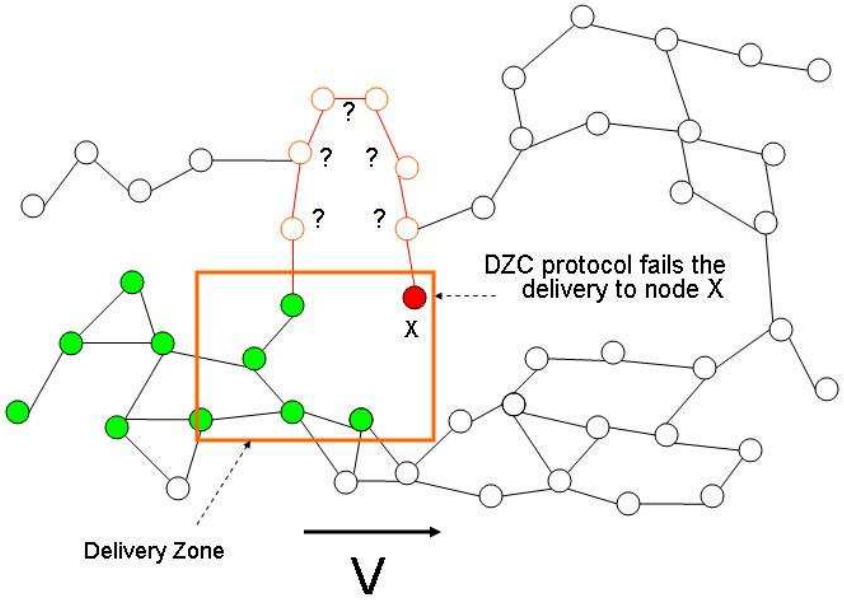


Figure 2.4: DZC protocol cannot guarantee delivery

we further investigate these challenges under the backdrop of random sensor networks and propose a reliable mobicast protocol.

2.6 A Reliable Mobicast Protocol

As we alluded earlier, a key challenge is the reliable mobicast delivery specified by (2.3) on networks of arbitrary topology while using only limited information about the network topology. In this section, we explain the key assumptions regarding the network, describe the framework for a reliable mobicast protocol, and offer an analysis and proof of reliability for this protocol. Our effort in deriving the protocol yields new insights and concepts useful in the study of spatiotemporal information dissemination strategies across sensor networks.

2.6.1 Sensor Network Model

The sensor network model for our protocol is as follows. The network is not partitioned, and all nodes are location-aware, i.e., they know their location $\vec{r}$ with reasonable accuracy. The maximum clock-drift among the sensors in the system is assumed to be small enough to be negligible. All nodes support wireless communication and are able to act as

routers for other nodes. Local wireless broadcast is reliable, i.e., once a local broadcast is executed, it will be heard by all its neighbors within latency τ_1.

2.6.2 Forward-Zone Constrained Mobicast Protocol

In this section we propose a mobicast protocol featuring a "forwarding zone" that cruises in front of the moving delivery zone at a certain prescribed distance. Only nodes in the path of the forwarding zone will participate in the mobicast forwarding. We call this protocol the "Forward-Zone Constrained" (FZC) mobicast protocol. In order to describe the FZC mobicast protocol more concisely, we need to introduce some terminology. The reader is reminded that the delivery zone, specified by the application , is an area where the delivery of messages to the application takes place. Our protocol creates and uses a "forwarding zone" $F[t]$ that is moving at some distance ahead of the delivery zone, as shown in Fig. 2.5. We call the distance between the forwarding zone and its associated delivery zone the "headway distance". The shape of the forwarding zone depends on the shape of the delivery zone, and the topology of the underlying network. More specifically, in our protocol, the shape of the forwarding zone is generated from a "seed shape" which we call the "core" of the forwarding zone with a metric of the network topology. The choice of the headway distance and the size of the forwarding zone guarantees that all nodes entering the delivery zone will have received the mobicast message in advance, even if some of them are not directly connected to any nodes already in the delivery zone. The forwarding zone limits retransmission to a bounded space while ensuring that all nodes that need to get the message will. We will discuss how the forwarding zone is determined in the next section.

While nodes in a forwarding zone retransmit the mobicast message as soon as they receive it for the first time, the nodes in front of the forwarding zone enter a "hold-and-forward" state whenever they hear the mobicast message. They will forward the message only after becoming members of the forwarding zone. As we pointed out earlier, the action of the nodes in the hold-and-forward zone implements the "just-in-time" feature of the mobicast delivery policy while keeping the average slack time t_{slack} small. This behavior results in a virtual "hold-and-forward zone" in front of the forwarding zone, as also indicated in Fig.2.5.

When a request $\langle m, Z[t], T_s, T\rangle$ is presented to the mobicast service, it constructs and broadcasts a mobicast message to all the neighbors at time T_s. A mobicast packet $\tilde{m}$ contains the following information: a unique message identifier, a delivery zone descriptor, a forwarding zone descriptor, the session start time T_s, the session lifetime T, and the message data m. The unique message identifier is created from the combination of the

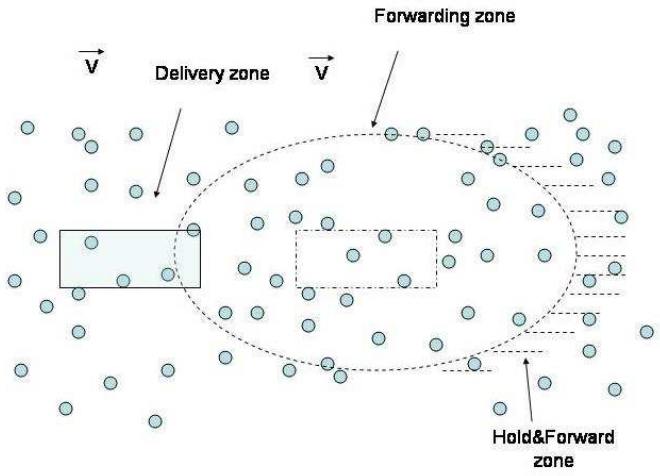


Figure 2.5: Mobicast example

location of the source and the time T_s of the request. The delivery zone descriptor encodes the original location, the shape of the zone, and its velocity. The forwarding zone descriptor encodes the shape and the original location of the forwarding zone, which is computed using some knowledge about the network and the shape of the delivery zone. We will discuss in detail the computation of the forwarding zone in later sections.

The FZC mobicast protocol is described in Fig.2.6. While not explicitly shown in the code, this mobicast protocol exhibits two phases in its spatiotemporal behavior. The first is an initialization phase, in which the nodes are trying to "catch-up" with the spatial and temporal demands of the *mobicast*. When a node in the path of the forwarding zone receives a message for the first time, it rebroadcasts the message as soon as possible. This phase continues until a stable forwarding zone travelling at a certain distance d_s ahead of the delivery zone is created.

The second is a cruising phase where the forwarding zone moves at the same velocity as the delivery zone. The protocol enters this phase after the delivery zone and the forwarding zone reach the stable headway distance d_s. This cruising effect is achieved by having the nodes at the moving front of the forwarding zone retransmit the mobicast message in a controlled "hold-and-forward" fashion to make the forwarding zone move at the velocity $\vec{v}$. The initialization and the cruising phases together establish mobicast property (2.3) with t_{init} being the time required by the initialization phase.

In the next section we turn our attention to explaining how the forwarding zone and its stable headway distance are computed, what is the value of t_{init} given a specific

Upon hearing a mobicast message $\tilde{m}$ at time t.

1.**if** ($\tilde{m}$) is new and $t < t_0 + T$
2. **if** (I am in F[t]) **then**
3. broadcast $\tilde{m}$ immediately ; // fast forward
4. **if** (I am in Z[t]) **then**
5. deliver the message data D to the application layer;
6. **else**
7. compute the earliest time t_{in} for me to enter the delivery zone;
8. **if** t_{in} exists and $t_{in} < t_0 + T$
9. schedule delivery of data D to the application layer at t_{in};
10. **end**
11. **end**
12. **else**
13. compute the earliest time t' for me to enter the forwarding zone;
14. **if** t' exists
15. **if** $t_0 \leq t' \leq t$
16. broadcast $\tilde{m}$ immediately ; // catch-up!
17. **else if** $t < t' < t_0 + T$
18. schedule a broadcast of $\tilde{m}$ at t'; //hold and forward
19. **end**
20. **end**
21. **end**
22. **end**

Figure 2.6: The FZC mobicast protocol

mobicast request and the spatial properties of the underlying network, and how the protocol delivers on its guarantees.

2.7 Reliability Analysis

The key elements in the FZC mobicast protocol (Fig.2.6) are the forwarding zone and its headway distance d_s. As we mentioned earlier, the purpose of the forwarding zone and its headway distance is to ensure that all the nodes entering a delivery zone will receive the mobicast message in advance, while minimizing the total number of nodes participating in each mobicast session.

The shape of the forwarding zone depends on the following three factors: the shape of the delivery zone, the spatial distribution of the network nodes, and the topology of the

network. Fig.2.7 illustrates this for a rectangular mobicast delivery zone (solid rectangle). The source node S initiates a mobicast. For node A to be able to deliver the message (to the application layer) when it becomes a member of the delivery zone, it must have received the message by that time. In scenario Fig.2.7(a), the message is required to have gone through G in order for it to reach A. This requires A and G to be in the forwarding zone together at some point in time before A can receive the message. Otherwise, the mobicast message will not be forwarded to A, since A is in a "past" location comparing to G (with respect to the delivery zone velocity direction). On the other hand, if the network connectivity is "denser," as in Fig.2.7(b), the width of forwarding zone represented by the dashed rectangle, can be smaller. Furthermore, in Fig.2.7(a) the height of the forwarding

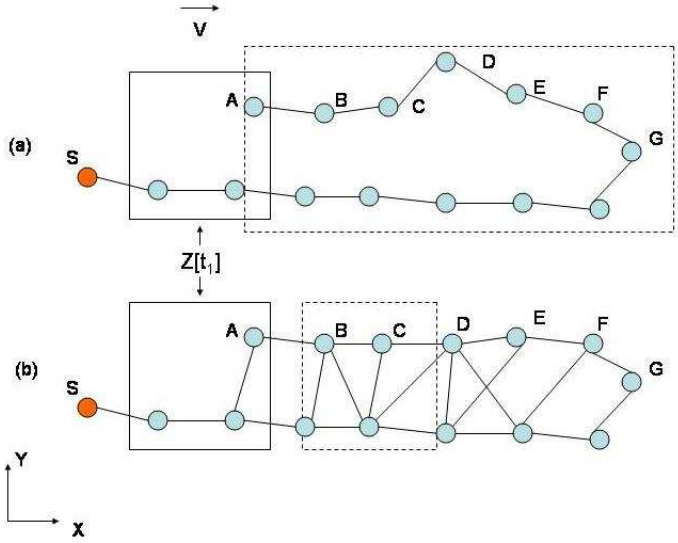


Figure 2.7: Spatial and connectivity configuration of the network influences the size of the forwarding zone

zone has to be bigger than the height of the delivery zone so as to include D. Otherwise, nodes A, B and C will be virtually disconnected from the rest of the network nodes, because node D will not participate in the routing process. This is just one special example. The question we would like to address is, in an arbitrary sensor network, how does one determine the forwarding zone and its headway distance for a specific delivery zone?

We found that the minimum size, shape, and headway distance for a mobicast protocol that provides a strong spatial and temporal delivery guarantee (in the presence of an arbitrary network topology) depend on two network metrics we call "Δ-compactness" and "Γ-compactness." These two metrics capture the spatial and temporal information

propagation properties of geometric networks, and are related to three of the following four distance metrics between two nodes i and j in a network:

- Euclidean distance, denoted as $d(i, j)$;
- Shortest network distance, in terms of smallest network hops, denoted as $h(i, j)$;
- Shortest Euclidean path distance, in terms of the shortest Euclidean path length, denoted as $D(i, j)$.
- S2 distance, defined as *Smallest* Euclidean path length among the set of *Shortest* network paths between nodes i and j, denoted as $\tilde{d}(i, j)$. We will call the corresponding path S2 path.

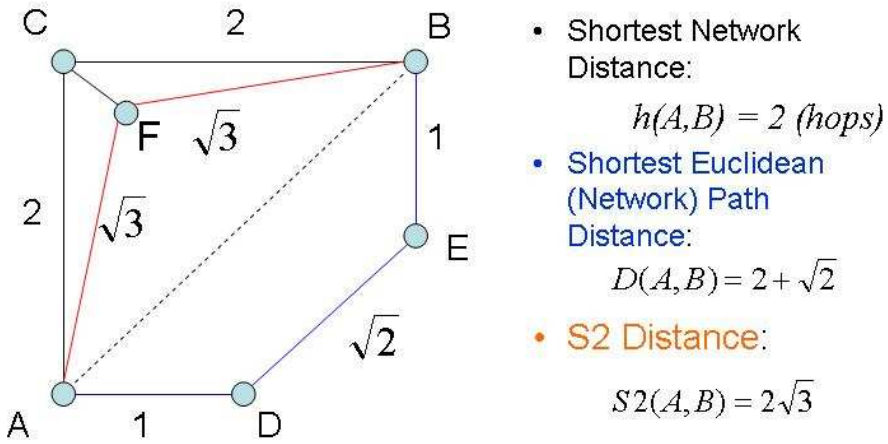


Figure 2.8: Three network distances

Note that a shortest hop network path is not necessarily a shortest Euclidean network path and vice versa. A shortest Euclidean network path is not necessarily a shortest S2 path and vice versa. Figure 2.8 illustrates such an example. The solid edges in Figure 2.8 represent the network connections. The numbers along the edges represent the Euclidean lengths of the corresponding distances. The shortest network paths between nodes A and B include $\overline{ACB}$ and $\overline{AFB}$ since both of them are 2 hops. The shortest Euclidean network path between nodes A and B is $\overline{ADEB}$, whose length is $2+\sqrt{2}$ (Euclidean unit). However,

the S2 path between nodes A and B is $\overline{AFB}$, even though its length is $\sqrt{3}$ (Euclidean unit), which is greater than the length of $\overline{ADEB}$. The unique definition of S2 effectively connects the world of Euclidean geometry and the world or communication topology. Its uniqueness makes S2 distance play a key role in our spatiotemporal multicast protocols.

Next we first formalize the network compactness metrics and discuss how they relate to the computation of the forwarding zone and its headway distance. Then we show that our protocol provides the desired spatiotemporal guarantees given the proper choice of the forwarding zone and its headway distance.

2.7.1 Computing the Forwarding Zone

In order to describe how the minimum forwarding zone can be determined for a specific delivery zone in an arbitrary network, we first introduce the definition of the "Δ-compactness" measure for the network.

Δ-compactness

Given a geometric graph/network $G(V, E)$, Δ-compactness seeks to quantify the relation between the Euclidean distance and the S2 distance among network nodes. We denote the Euclidean distance to S2 distance ratio between two nodes i and j as $\delta(i, j)$, i.e.,

$$\delta(i,j) = \frac{d(i,j)}{\tilde{d}(i,j)} \tag{2.6}$$

We call $\delta(i, j)$ the pairwise "Δ-compactness" between nodes i and j. The Δ-compactness of a geometric graph $G(V, E)$ is defined as the smallest Δ-compactness of all node pairs of the network:

$$\delta = \min_{i,j \in V} \{\delta(i,j)\} \tag{2.7}$$

In a sense, Δ-compactness describes the worst case "distortion" of the shortest paths in a network. Figure 2.9 shows Δ-compactness value of four different networks.

Note that Δ-compactness has a close relation with the terms "dilation"[60], "spanning ratio"[19], and "stretch-factor"[127] used in the graph and computational geometry community. "Dilation" is defined as the maximum ratio between Euclidean path distance and geometric distance,

$$Dilation = \max_{i,j \in V} \{\frac{D(i,j)}{d(i,j)}\}$$

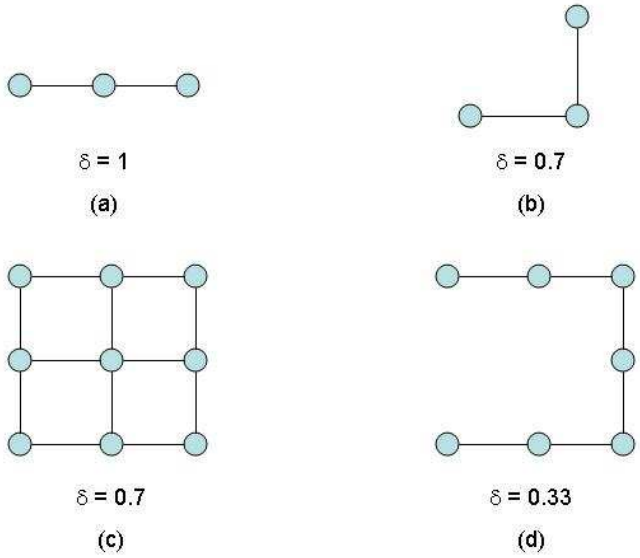


Figure 2.9: Δ-compactness of various networks

while Δ-compactness is defined as minimum ratio between the geometric distance and the corresponding S2 distance. They have more than an inverse relationship. For instance, for nodes A and B in Figure 3.2,

path $\overline{ACB}$ contributes to the computation of Δ-compactness while path $\overline{ADEB}$ (which is shorter than $\overline{ACB}$ in Euclidean measure) contributes to the computation of dilation. Note that Δ-compactness is computed on the set of shortest network paths (path of minimum hops) only, while dilation is computed on the set of all paths. Path $\overline{ADEB}$ has 3 hops and is not a shortest network path between A and B, even though it is a shortest Euclidean network path[2] between them. For convenience, we call the inverse of Δ-compactness Δ-*dilation* . One can see that the Δ-dilation of this graph is $4/2\sqrt{2} = 1.414$, while the dilation is $3.56/2\sqrt{2} = 1.26$. (Reader can verify that the pairwise dilations of other pairs of nodes are smaller than 1.26.)

[2]The path lengths are

$$EuclideanLength(\overline{ACB}) = 4$$
$$EuclideanLength(\overline{ADEB}) = \sqrt{3} + 2\sqrt{2} - 1 = 3.56$$

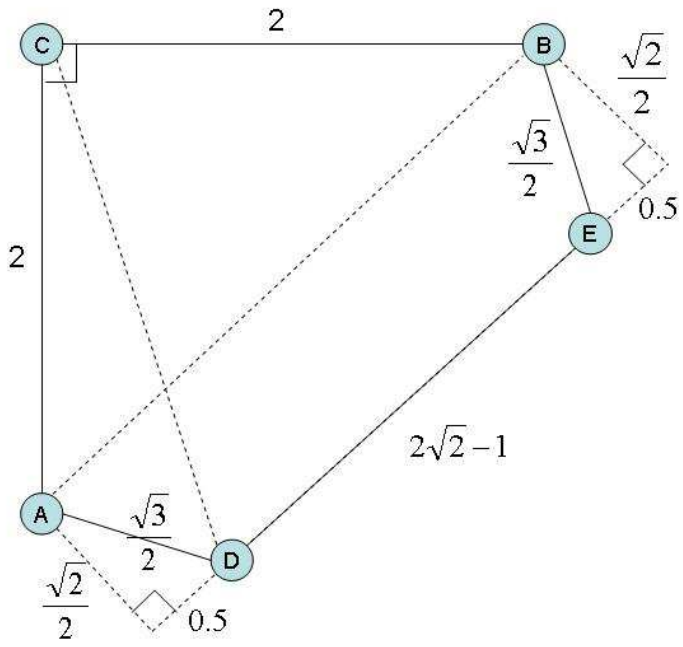


Figure 2.10: Dilation and Δ-compactness

Theorem 1 *Let i, j be any two nodes in a network with Δ-compactness δ. Let $E(i, j, \delta)$ be an ellipse using i, j as two foci and with eccentricity δ. There is at least one shortest path between i and j inside the ellipse $E(i, j, \delta)$.*

Proof: We prove this theorem by contradiction. Assume the theorem is not true. There must be at least one pair of nodes i and j, whose shortest paths all have at least one vertex outside the ellipse $E(i, j, \delta)$. Using the fact that for all points k on the ellipse, $d(i, k) + d(j, k) = d(i, j)/\delta$, it is easy to prove in this case

$$\tilde{d}(i, j) > \frac{d(i, j)}{\delta}$$

that is

$$\delta > \frac{d(i, j)}{\tilde{d}(i, j)}$$

this directly contradicts the definition of Δ-compactness (2.7). ■

This theorem is very useful for limiting the forwarding region while guaranteeing point to point message delivery in a geometric network. In our case, this metric helps decide the shape and size of the forwarding zone, which turns out to relate to a notion called "k-cover."

K-cover

We introduce the notion "k-cover" of a polygon to simplify the mathematical description of the forwarding zone. The k-cover of a convex polygon P is defined as *the locus of all points p in the plane for which two points q and r in the polygon P exist such that*

$$d(p,q) + d(p,r) \leq kd(q,r) \tag{2.8}$$

where the $d(x,y)$ is the distance between points x and y.

Theorem 2 *Let i, j be two nodes in a network with Δ-compactness value δ, and assume that i and j are inside a convex polygon P. The $\frac{1}{\delta}$-cover of P contains at least one shortest path between i and j.*

Proof: The proof is similar to that of theorem (1). Proof by contradiction. Assume that none of the shortest network paths between nodes i and j is inside the $\frac{1}{\delta}$-cover of P. That means for any shortest network path between i and j, there is at least one node z on the path that is outside of the $\frac{1}{\delta}$-cover. Then, from the definition of k-cover we know that

$$d(i,z) + d(z,j) > \frac{1}{\delta} d(i,j) \tag{2.9}$$

We also have

$$\tilde{d}(i,j) \geq d(i,z) + d(z,j) \tag{2.10}$$

This leads to

$$\delta > \frac{d(i,j)}{\tilde{d}(i,j)} \tag{2.11}$$

This contradicts the definition of Δ-compactness . ■

One may view an ellipse to be a special case of k-cover. An ellipse of eccentricity e is a $\frac{1}{e}$-cover of the line segment between the two foci of the ellipse. In other words, the k-cover is a generalization for the ellipse.

Henceforth, for convenience, we will also call a k-cover for a polygon P "a cover for polygon P with *cover factor k*".

The Forwarding Zone

Given a mobicast delivery zone of convex shape P, if the mobicast is executed on a network with Δ-compactness value δ, then we choose the shape of the forwarding zone's core to be P and the shape of the forwarding zone to be the $\frac{1}{\delta}$-cover of its core. In other

words, the shape of the forwarding zone is a cover for the delivery zone with a cover factor $\frac{1}{\delta}$. In this case, we will also say the forwarding zone factor is $\frac{1}{\delta}$. For a given delivery zone, the bigger the forwarding zone factor is, the bigger the forwarding zone is.

Corollary 1 *Let i, j be two nodes in the core of a forwarding zone on a network whose Δ-compactness is δ. The forwarding zone contains at least one shortest path between i and j.*

Proof: This results from theorem (2) and the construction of the forwarding zone. ■

2.7.2 Computing the Stable Headway Distance

The headway distance tells the protocol how far ahead to prepare for the message delivery in order not to miss the delivery deadline as a result of some unexpected distortions on related network paths. It should be intuitively clear that a network with more "indirect" network paths requires a longer headway distance than one whose paths are more "direct." In order to capture this notion more precisely, we introduce the "Γ-compactness" metric.

Γ-compactness

Γ-compactness quantifies the relation between the shortest network distance and the Euclidean distance among the nodes in a geometric network. We define the Γ-compactness of a geometric graph $G(V, E)$ to be the minimum ratio of the Euclidean distance to the shortest network distance between any two nodes in the network, i.e.,

$$\Gamma = \min_{i,j \in V} \frac{d(i,j)}{h(i,j)} \tag{2.12}$$

Intuitively, if a network's Γ-compactness value is γ, then any two nodes in the network at a distance d have a shortest path no greater than d/γ hops.

Theorem 3 *Let N be a network with a Γ-compactness value γ and let τ_1 be its maximum 1-hop communication latency. The lower bound of the maximum message delivery speed over the space on N is $\frac{\gamma}{\tau_1}$.*

Proof: Let $d(i, j)$ be the distance between two arbitrary nodes i and j in the network. We know that the shortest network path h between the two nodes is bounded by

$$h(i,j) \leq \frac{d(i,j)}{\gamma} \tag{2.13}$$

We also know that a message sent from one node to another node h-hops away takes no longer than $h\tau_1$ if each intermediate node forwards the message immediately after receiving it. Let t be the time it takes for the message to go from i to j. In this case we have

$$t \leq h(i,j)\tau_1$$

From this we know that the average speed v of this information propagation over distance $d(i,j)$ is

$$v = \frac{d(i,j)}{t} \geq \frac{d(i,j)}{h\tau_1} \geq \frac{\gamma}{\tau_1} \tag{2.14}$$

Note that the bound $\frac{\gamma}{\tau_1}$ is not dependent on $d(i,j)$. This inequality (2.14) is true for any two nodes in any network with Γ-compactness value γ, when all nodes in the network relay the message as soon as possible. That means $\frac{\gamma}{\tau_1}$ is a lower bound on the maximum spatial message delivery speed on networks with Γ-compactness value γ. ■

Theorem (3) states that, given a geometric network, there is a clear limit to how fast spatiotemporal information dissemination can be achieved. For instance, given a geometric network with Γ-compactness value γ, one can not guarantee the delivery zone to move at a speed higher than $\frac{\gamma}{\tau_1}$ in all areas.

2.7.3 The Headway Distance

The stable headway distance d_s must be large enough to ensure that the message has been received when the delivery zone reaches a node, i.e., $t_{in} > t_r$ is achieved for all nodes.

Theorem 4 *Let S_d be the maximum distance between the boundary points of the delivery zone, v be the speed of the delivery zone, τ_1 be the 1-hop maximum network latency of the network and γ be its Γ-compactness . If we let $d_s = v\tau_1\lfloor\frac{S_d}{\gamma}\rfloor$, then all nodes in the core of the forwarding zone will have received the the mobicast message by the time delivery zone reaches them assuming there is at least one node in the core that has received the message.*

Proof: Let us consider a snapshot of the mobicast at some time t and the core of the current forwarding zone. Let i denote the node in the core that already has the message. Its distance to all other nodes in the core is less than S_d, because S_d is the maximum size of the delivery zone, as well as that of the core. The longest of the shortest network paths from i to all other nodes in the core of is less than $\lfloor\frac{S_d}{\gamma}\rfloor$ hops. In turn, at most $\tau = \lfloor\frac{S_d}{\gamma}\rfloor\tau_1$ time is needed for a message to traverse the core if all nodes forward the message as soon

as possible. We can conclude that after τ, all nodes in the core of the forwarding zone will get the message, because in the protocol all nodes inside the forwarding zone forward mobicast messages as soon as possible, and there is always a shortest path inside the forwarding zone for any two nodes inside its core. Since the speed of the delivery zone is v, a distance $d_s = v\tau_1\lfloor\frac{S_d}{\gamma}\rfloor$ takes exactly τ time to be traversed.

Hence, it is true that all nodes in the core of the forwarding zone will have received the the mobicast message when the delivery zone reaches them, assuming at least one node in the core has received the message and given the headway distance $d_s = v\tau_1\lfloor\frac{S_d}{\gamma}\rfloor$. ■

Given the headway distance d and the shape F of the forwarding zone, a node can easily determine the current forwarding zone using velocity v, current time t, sending time t_0 and the source location r_0. Note that t_0 and r_0 can be obtained from the mobicast protocol message header.

2.7.4 Duration of Initialization Phase

As we pointed out earlier, it is in the cruising phase that the mobicast protocol guarantees on-time delivery. In the initialization phase, the timing constraint of mobicast is realized in a best-effort manner. It is possible that in the initialization phase, some nodes may not get the messages in time. The initialization phase continues until one node inside the core of the forwarding zone that is d_s ahead of the delivery zone receives the mobicast message. From theorem (4), we know that after this the timing constraint of mobicast is always satisfied.

The time it takes for the mobicast protocol to enter the cruising phase, t_{init}, is related to the needed stable distance, the delivery zone speed, and the maximum admissible spatial propagation speed of the network.

Theorem 5 *Let d_s be the required headway stable distance between the forwarding zone and the delivery zone. Let w be the width of the delivery zone. Let v be the speed of the delivery zone and u be lower bound on the maximum message delivery speed achievable in the network. The mobicast protocol initialization time t_{init} is no greater than $\frac{(d_s+w)}{u-v}$*

Proof: In the protocol, the nodes in the forwarding zone and between the forwarding zone and the delivery zone retransmit the message immediately the first time they receive it. As such, the protocol achieves a maximum message propagation speed v_{max} in this phase. This message propagation speed relative to the delivery zone is $v_{max} - v$.

Meanwhile, the end-to-end distance between the delivery zone and the core of the forwarding zone is $d_s + w$, which can be covered by a message propagating at the speed $v_{max} - v$ in $t = \frac{d_s+w}{v_{max}-v}$ time. When a message from the delivery zone reaches the core of the forwarding zone d_s distance ahead of the delivery zone, by definition the initialization phase is over. Hence we have

$$t_{init} \leq t = \frac{d_s + w}{v_{max} - v} \leq \frac{(d_s + w)}{u - v} \tag{2.15}$$

in the above we also used $u < v_{max}$, which by definition is true. ■

The Spatiotemporal Guarantees of the Protocol

The spatiotemporal guarantees of the FZC mobicast protocol (2.6) are addressed by the following theorem:

Theorem 6 *If at any instant of time in a mobicast session, its (user-defined) delivery zone covers at least one node in the network, our mobicast protocol delivers property (2.3) with*

$$t_{init} \leq \frac{v\tau_1 \lfloor \frac{S_d}{\gamma} + w \rfloor}{\frac{\gamma}{\tau_1} - v}$$

.

Proof: If a delivery zone covers at least one node in the network at any instant of time, then whenever the last node in a delivery zone is leaving, there must be another node entering it. The same is true for the core of the forwarding zone, because it is of the same shape as the delivery zone and moves on the same path. If a node in the core of the delivery zone receives the mobicast message, it will always be able to pass the message to all other nodes on the path because our protocol and choice of the forwarding zone guarantees that if two nodes ever appear together in the same core of the forwarding zone, one having the message means the other will get it too.

By using theorems (4) and (5), it is easy to see that property (2.3) is satisfied. ■

2.8 Discussion

For reliable mobicast, we introduced two network compactness metrics to help us choose the right forwarding zone and its headway distance for a given delivery zone so as to

achieve the mobicast delivery guarantee without unnecessary flooding. These compactness values must to be computed for supporting the FZC mobicast protocol. Calculating them involves computing the shortest path and Euclidean distances of each pair of nodes in a given network. The all-pair shortest path of a graph $G(V, E)$ can be computed in $O(VE \log V)$ time by using Johnson's algorithm [42]. All-pair distance can be computed in $O(V^2)$ time. Therefore, we can compute the Γ-compactness of the graph in $O(VE \log V)$ time. Δ-compactness can also be computed in $O(VE \log V)$ time. It is not feasible for individual sensor nodes to compute these values in a large network. A central server may be used to collect all the location and connectivity information, do the computation and use one broadcast to inform all the nodes this value. However, for the local compactness values, it is possible for the sensor nodes to compute the metric values as they involve only a relatively small number of nodes in their respective neighborhood.

While we chose the shape of the forwarding zone to be a $\frac{1}{\delta}$-cover of the shape of the delivery zone, this was done only for the purpose of analysis. Computing an exact k-cover for an arbitrary polygon P can be difficult. Yet one can always choose some approximation techniques such as using the k-cover of P's bounding box or bounding circle, which is computationally much simpler, but still has the required property that a shortest path between two nodes inside a specific instance of the delivery zone exists in the cover. However, the resulting forwarding zone is bigger than necessary, and thus may entail more re-transmissions for the same delivery goal. We should note that in the FZC protocol the forwarding zone only needs to be computed once by the sender. The nodes that receive the mobicast message only need to translate the forwarding zone with respect to their distances from the sender.

An important aspect of mobicast is that applications have control over the velocity of the information dissemination over the space. This brings many new spatial and temporal coordination and interaction possibilities across a network. For instance, an application might use a mobicast to send some information to the east at a speed of 40 miles per hour. One second later, it may detect a change (e.g., there is a change in the intruder's expected path) and may want to send the new information and stop further propagation of the old information in the network. As alluded to earlier, stopping previous information dissemination is impossible in conventional protocols which have explicit or implicit "as-soon-as-possible" delivery semantics. Yet, in mobicast, a "stop that message" message can be sent at a much higher speed, say 120 miles per hour, (or even more than 1000 miles per hour which we found possible in our simulation, see Chapter 3), with a same-size delivery zone along the previous path. Clearly, this new

mobicast recall message can easily catch up with its target message which propagates at a much lower speed.

Furthermore, for simplicity of presentation, our protocol essentially carries out flooding inside the forwarding zone. If the nodes have an accurate picture about the locations of their one-hop or two-hop neighbors, one can reduce the number of re-transmissions by using this knowledge in a manner similar to techniques proposed for improving broadcast efficiency [136, 145]. In a probabilistic guarantee scenario, one may also use probabilistic retransmission-reduction techniques such as the one proposed in [129]. A review of these and other related methods can be found in [162]. Our protocol, by only using the compactness values of the network, tries to use minimum number of bits to capture the relevant topology. If the nodes have local knowledge about the network topology in the neighborhood, (e.g., know the locations of all nodes within certain distance) more communication efficient mobicast protocols can be designed.

2.8.1 Additional Related Work

Mobicast is motivated by the need for coordination activities related to moving entities in the physical environment. In [31], Cerpa et. al. proposed a Frisbee model in which an active sensing zone moves through the network along with the target. [108] and [38] proposed several data service protocols for improving the accuracy of distributed sensing in mobile environments. Both protocols entail communication schemes that push information about the object to the nodes close the projected location of the object in the future. The EnviroTrack group management protocol [17] dynamically creates and maintains a group that tracks mobile entities in the environment. [53] proposed localization and dynamic tracking algorithms. However, neither of the aforementioned projects include communication mechanisms geared toward meeting explicit spatiotemporal constraints related to mobility. Mobicast can be viewed as complimentary to these projects by providing a convenient underlying communication mechanism that allows applications to push information with specified spatiotemporal requirements.

Mobicast has a spatial multicast component similar to geocast, a multicast paradigm proposed by Navas and Imielinski [128]. In a geocast protocol, the multicast group members are determined by their physical locations. The initiator of a geocast specifies a fixed area for a message to be delivered, and the geocast protocol tries to deliver the message only to the nodes in that area. Ko and Vaidya [99] investigated geocast in the context of mobile ad hoc networks. Other mechanisms ([155, 110, 18]) have been proposed to improve geocast efficiency and delivery accuracy in wireless ad hoc networks. Mobicast differentiates itself from geocast by a mobile delivery area rather than a fixed one, and

gives application developer a powerful tool for controlling information dissemination in the spatiotemporal domain rather than just the spatial domain. As a mobicast protocol, FAR uses face routing to achieve high space delivery guarantee and uses timed forwarding for controlling information propagation speed.

2.8.2 New Challenges

We have proved that our FZC mobicast protocol is able to achieve the strong mobicast delivery guarantees specified in Section 2 in theory, given a proper choice of the forwarding shape and its headway distance and under a set of necessary assumptions. Note that for the FZC mobicast protocol, the forwarding zone is the $\frac{1}{\delta}$-cover of the delivery zone. A small value for Δ-compactness implies a relatively big mobicast overhead, defined as the number of nodes that participate in the mobicast message forwarding. Note also that in the previous protocol, the network compactness values are used because of the need for strong delivery guarantees, captured by the concern with the worst case path distortion in a network rather than the average case. Several new questions arise: (1) What is the typical compactness value for common sensor networks? (2) Can we make a network more compact to support better spatiotemporal communication? (3) The protocol uses the worst case compactness among all paths, as it was geared towards 100% delivery guarantee. This choice might be pessimistic if the worst case is rare. How will an optimistic choice of forwarding zone perform in reality? (4) Can we use a local notion of compactness and can the mobicast session and the forwarding zone be adaptively adjusted to the local compactness values? We turn our attention to addressing these questions in the next chapter.

Chapter 3

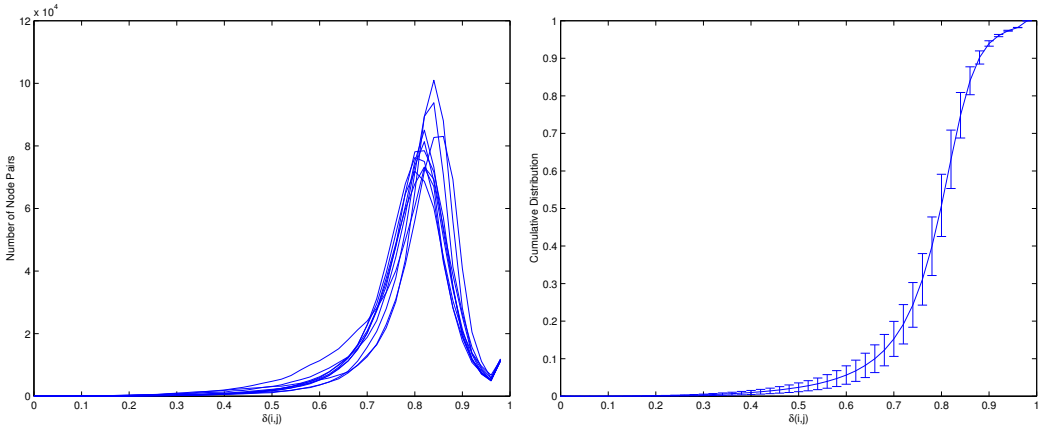
Properties of Random Geometric Network and Alternative Mobicast Protocols

In this chapter, we first investigate the compactness properties of random sensor networks. Using the insights gained from this investigation we propose alternative mobicast strategies and evaluate them via network simulations.

3.1 The Statistical Compactness Properties of Random Unit Disk Graphs

In the previous chapter, we showed that a network with higher compactness admits a more economic FZC mobicast protocol, i.e., fewer nodes need to participate in mobicast forwarding. Notice that Δ-compactness is the minimum ratio of the Euclidean distance and the S2 distance, which accounts for the worst case "indirect" path among all nodes. An immediate question is, how typical is the worst case scenario? The answer to this question is very important to applications which do not need 100% delivery guarantee. If most of the pairwise ratios among nodes are much larger than the minimum, then a choice of a much smaller forwarding zone may be able to practically guarantee mobicast delivery most of the time with only a small number of participating nodes. Energy can be saved by slightly sacrificing the delivery guarantee. This is desirable for sensor networks as they are typically resource limited and for applications that are not very strict on delivery guarantees.

Motivated by these observations, we carried out several experiments to see the potential distribution of the pairwise Δ-compactness value $\delta(i,j)$ in randomly distributed networks. We found that, indeed, in random networks of uniform distribution, most $\delta(i,j)$ are close to 0.8 while the minimum $\delta(i,j)$ is close to zero. Figure 3.1(a) shows



(a) (b)

Figure 3.1: Pairwise Compactness distribution

the distribution of pairwise compactness value $\delta(i,j)$ in 10 different randomly generated uniformly distributed networks. Figure 3.1(b) shows the average case (averaged over the above 10 network instances) in a cumulative distribution view with standard deviation bars. Note that more than 90% of node pairs have a $\delta(i,j)$ greater than 0.6, while the minimum $\delta(i,j)$ (i.e., the value of Δ-compactness of the network) is less than 0.2. Note also that a mobicast protocol using Δ-compactness value $\delta = 0.2$ to construct its forwarding zone results in a forwarding zone size 25 $(= (1/\delta)^2)$ times bigger than the delivery zone, while using $\delta = 0.6$ results in a forwarding zone less than 3 times bigger than the delivery zone. So more than 200% $\sim(\frac{1/0.2-1/0.6}{1/0.6})$ of the forwarding cost may be saved by slightly sacrificing the delivery guarantee if one uses $\delta = 0.6$ rather than the minimum pairwise compactness value in the construction of the forwarding zone. (Note that in the above calculation we use a linear rather than a quadratic relation of $1/\delta$ in estimating overhead because, while the forwarding zone size is quadratic to $1/\delta$, its integral volume over the path of a mobicast is proportional to $1/\delta$).

These results suggest three approaches to improve the efficiency of mobicast. The first is to design a sensor network with high compactness to support spatial temporal communication. The second is to use a smaller forwarding zone than the one needed for an "absolute" delivery guarantee. The third is to use a protocol that adapts to the local compactness conditions rather than the global one. We investigate and examine these approaches in the next few sections.

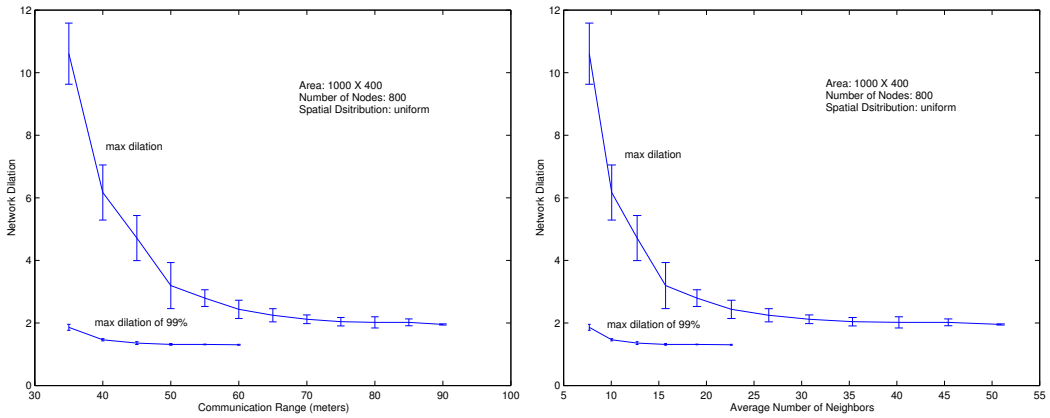
3.2 Impact of Node Density on Network Compactness

As we pointed out earlier, for a specific delivery zone, the more "compact" a network is, the smaller the forwarding zone needs to be. An immediate question is, can we design the sensor network so as to make its Δ-compactness value as close to the maximum value of one as possible? Since we want to continue with the random distribution assumption, there is only one design dimension left: the sensor node density. Note that we define sensor density as the average number of immediate network neighbors for each node, rather than number of nodes in a unit area.

Intuitively, the higher the sensor density, the "better" connected the sensor network is and the larger the corresponding network Δ-compactness . To verify this observation, we designed the following experiment. We scatter 800 sensors uniformly distributed in a 1000x400 rectangular area and select only configurations that are not partitioned for a communication range of 35. (Note that because of random distribution, the network is sometimes partitioned. 35 is close to a critical range for connectivity in our experimental configuration). For the surviving configurations, we compute the values of Δ-compactness assuming communication range values between 35 and 90. Note that in this experiment we chose to vary the communication range rather than to vary node density directly by adding more nodes to the area. The reason we chose to vary the communication range as a mechanism to vary the relative sensor density is because this does not change the actual location configuration of the sensors in the experiment, which makes the corresponding compactness value comparison more meaningful.

The above procedure was repeated for five different configurations and the results are presented in Figure 3.2. Figure 3.2(a) shows the average (across the 5 runs) Δ-Dilation (defined as the inverse of Δ-compactness) versus the change of communication range. Figure 3.2(b) shows a corresponding figure with the average node degree as the x-axis.

The results show that the network compactness indeed increases when the node density increases. But surprisingly, there appears to have a saturation point at a moderate density. The network exhibits a rapid increase in compactness (rapid decrease in Δ-dilation) when the average number of neighbors changes from 8 to 15 and then starts to saturate. This appears to be an area to increase the compactness of the network with highest efficiency for these randomly distributed networks. This may provide a good heuristic for deploying mobicast/communication friendly sensor networks. For instance, if one wants to monitor a area of 1000x1000 square meters using sensors of average range



(a) (b)

Figure 3.2: (a)Δ-Dilation vs Range, (b)Δ-Dilation vs Average Number of Neighbors

50 meters, the total number of sensors should be about $\frac{1000\times1000}{\pi 50^2}\times 15 = 1910$ for a random scattering deployment method. One can also see that after a certain threshold ($15 \sim 20$ neighbors in this case), increasing the node density no longer introduces much benefit in terms of improving compactness.

In addition, we also examined the value of the majorities of the pairwise Δ-compactness and how they change with node density. The lower curve in Figure 3.2(a) and (b) shows how the lower bound of the top 99% of the $\delta(i, j)$ of the network changes with node density. One can see that the occurrence of the lower extreme compactness value is a rare event. This further suggests that an optimistic choice of k-cover for the forwarding zone is a good mobicast strategy in practice.

3.3 An Optimistic Mobicast Protocol

To verify our observations about the potential benefit of optimistic mobicast on random networks with uniform distribution, we implemented an extended mobicast protocol on the ns-2 network simulator. Our implementation was extended with a mode to let the user specify the parameter (delta) for determining the forwarding zone. This allows us to test the trade-off between the message[1] forwarding cost and the delivery guarantee.

The header of our mobicast protocol packet contains the following information:

- message type
- sender packet sequence number
- delivery zone size (radius)
- delivery zone velocity

[1]Note that we use "packet" and "message" interchangeably here. In the simulation, we only deal with cases where a mobicast message can be fit in one mobicast packet.

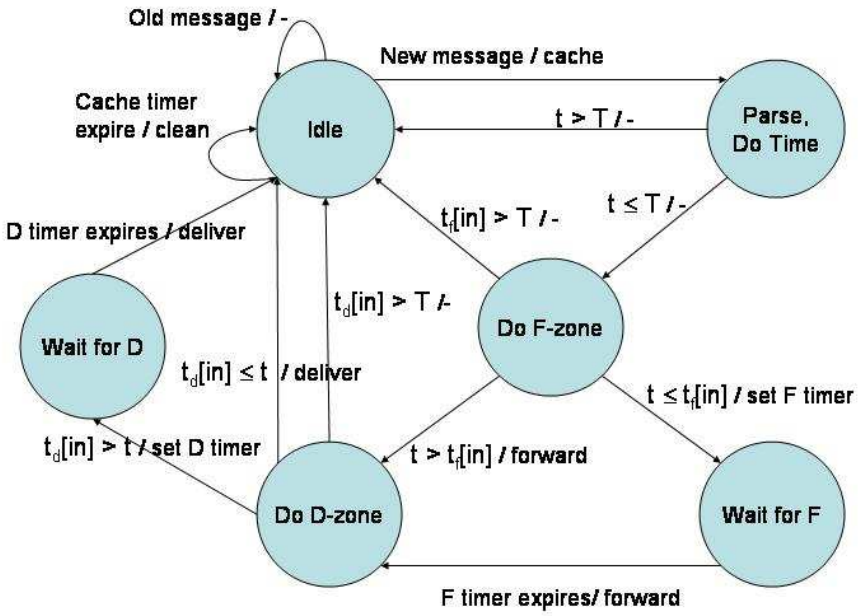
- sender location (x and y coordinates)
- sending time
- message lifetime
- delta factor
- gamma factor

Our protocol only provides support for a circular delivery zone. We also assume that the initial delivery zone is centered at the sender. One may augment the header with the information about the initial delivery zone center to allow applications to explicitly set the initial delivery zone location. Because this is not essential for our validation and verification test purposes, we simply default the sender location as the center of the initial delivery zone.

```
Upon hearing a optimistic mobicast message m̃ at time t.
1.if (m̃ ) is new and t < T
2.     cache this message
3.     if the value of the delta field is zero
4.          use local knowledge of delta for later computation
5.     else
6.          use the value in the packet for later computation
7.     end
8.     if (I am in current forwarding zone F[t]) then
9.          broadcast m̃ immediately ;                    // fast forward
10.         if (I am in current delivery zone Z[t]) then
11.              deliver the message data D to the application layer;
12.         else
13.              compute the earliest time t_d[in] for me to enter the delivery zone;
14.              if t_d[in] exists and t_d[in] < T
15.                   schedule delivery of data D to the application layer at t_d[in];
16.              end
17.         end
18.    else
19.         compute the earliest time t_f[in] for me to enter the forwarding zone;
20.         if t_f[in] exists
21.              if t_0 ≤ t_f[in] ≤ t
22.                   broadcast m̃ immediately ;          // catch-up!
23.              else if t < t_f[in] < T
24.                   schedule a broadcast of m̃ at t_f[in];      //hold and forward
25.              end
26.         end
27.    end
28. end
```

Figure 3.3: Optimistic Mobicast Protocol

The mobicast protocol is depicted in Figure 3.3 (modified from Figure 2.6, with additional compactness information processing). In this presentation we omit details about the geometric computation involved in determining if and when a node is in a forwarding zone and delivery zone, because it is not conceptually essential. The mobicast protocol also maintains a transient message cache (it is periodically cleaned by throwing out expired messages). Figure 3.4 is a state transition diagram for the complete protocol.



T: Message expiration time; t: Current Time;
$t_f[in]$: Earliest time to be in the forwarding zone;
$t_d[in]$: Earliest time to be in the delivery zone

Figure 3.4: State Transition Diagram of Implemented Mobicast Protocol

To minimize the dependence of simulation results on the network configuration used, our experiments were run on five different connected network configurations generated via uniformly distributing 800 sensor nodes on a 1000x400m area. Figure 3.5(b) shows one such configuration example used. The network connectivity pattern is shown Figure 3.5(a). One node close to the left is chosen as the mobicast sender. Our results are averaged over multiple runs on five network configurations. For all runs, the delivery zone velocity is 40m/s, from left to right, and each mobicast session has a lifetime of 20s. For all the configurations used, the critical communication range for all the nodes to form a connected graph is between 30 to 35 meters. We chose the delivery zone radius to be 45 meters.

We designed two sets of experiments. The first investigated how the mobicast delivery ratio and forwarding overhead changes with the size of the forwarding zone on these uniformly distributed networks. Delivery ratio is defined as the percentage of nodes in the delivery-zone path that received the mobicast message. Forwarding overhead is defined as the number of extra message transmissions per node delivery, i.e., the total number of retransmissions divided by the number of delivery zone nodes that actually received the message. Figure 3.6(a) shows the simulation results of delivery ratio versus the

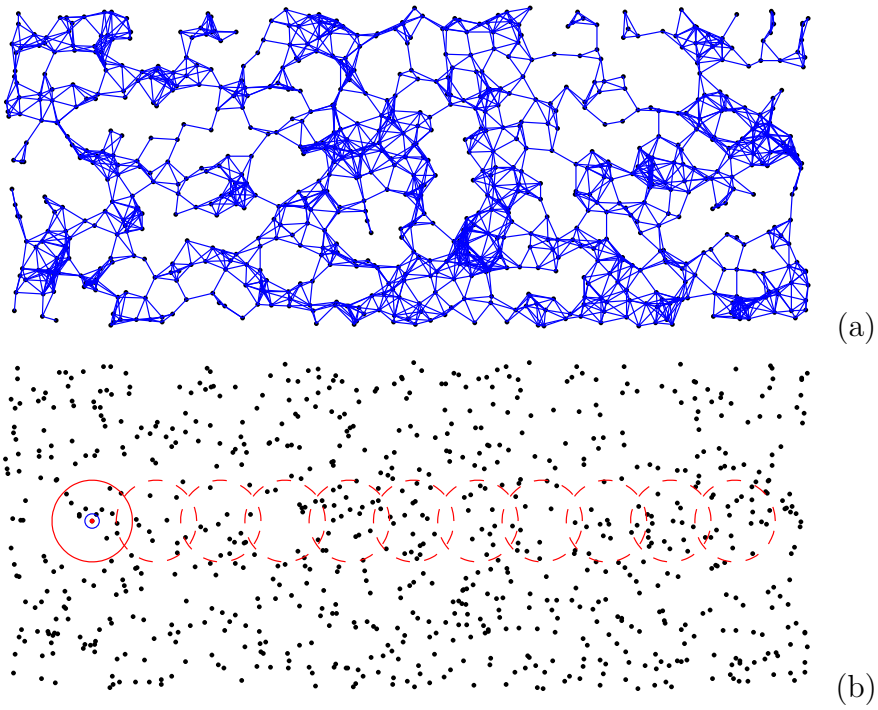
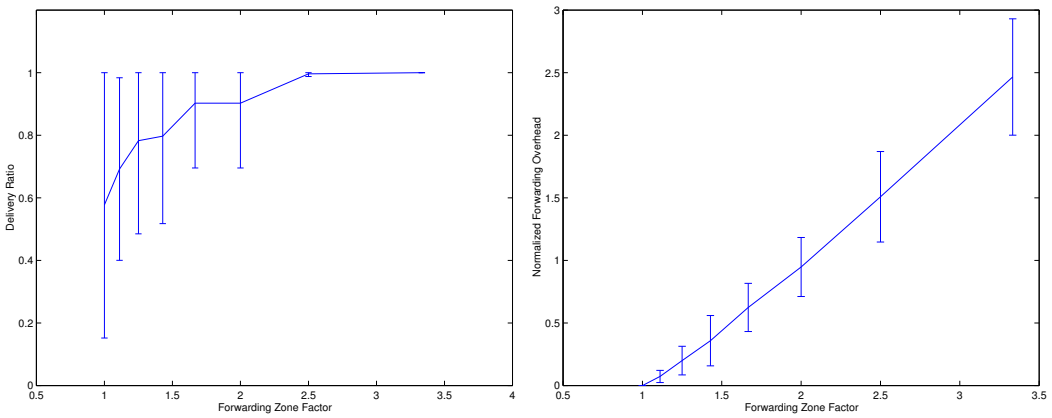


Figure 3.5: Optimistic Mobicast Simulation Example

normalized forwarding zone size (the actual k used in forwarding zone computation.) One can see the delivery ratio improves when the forwarding zone becomes bigger. The high variance in delivery ratio value is due to the random distribution of holes across different configurations, which causes each mobicast session to stop prematurely at different locations across different configurations. The limited number of network configurations used also contribute to this. Figure 3.6(b) shows how the forwarding overhead changes with the forwarding zone factor. Clearly, the message forwarding overhead increases almost linearly with the increase of forwarding zone factor.

The second set of simulation experiments was designed to investigate how the delivery ratio is affected when the network becomes more compact. Due to the limited scalability of ns-2, we again use the change of the communication range to change compactness, rather than by adding more nodes. In the experiment, the delivery zone radius was 45 meters. The communication radius varied from 35 to 45 meters. We collected results from multiple runs of mobicast using different forwarding zone factors over the five configurations and those results are summarized in Table 3.3.

From these results we can see that indeed the delivery ratio increases when the node density increases, and when the size of the forwarding zone increases. Again the high variance in the value is due to the random distribution of holes across different configurations and each mobicast session stops prematurely at different locations across



(a) (b)

Figure 3.6: (a) Delivery ratio vs Forwarding Zone Size; (b) Normalized Forwarding Overhead vs Forwarding Zone Factor

Table 3.1: Delivery ratio v.s. node density and forwarding size

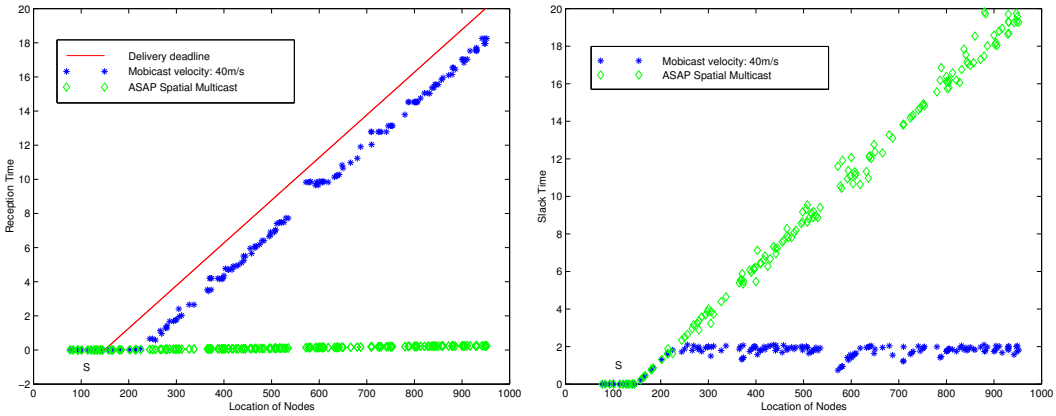
R \ δ	1.0	0.9	0.8	0.7	0.6	0.5	0.4
35	0.69± 0.29	0.78± 0.29	0.80± 0.28	0.90± 0.21	0.90± 0.21	0.99± 0.10	1
40	0.90± 0.21	0.90± 0.21	0.90± 0.21	1	1	1	1
45	0.998± 0.003	1	1	1	1	1	1

different configurations. These results also demonstrate that the forwarding zone in the FZC protocol (based on the value of worst case network compactness) is indeed sufficiently large to guarantee the reliable delivery on a connected network of random topology.

Considering communication speed is much faster than the delivery zone speed in our simulation, we let the headway distance the zero in these experiments. Note that in this case, when $\delta = 1$, the FZC protocol reduces to DZC protocol. In the above data, we can see in high network density, the greedy DZC protocol performs very well, achieving near 100% delivery.

In our simulation, we also examined the timeliness of mobicast delivery on these networks. More specifically, we wanted to see how far ahead a node received the mobicast message before entering the delivery zone (or how late after entering the delivery zone). Figure 3.7 shows one typical result of a mobicast session, when the communication range is 35m, the delivery zone radius is 45m, δ is 0.7, d_s is 0, and the mobicast speed is 40m/s. Figure 3.7(a) shows the mobicast packet reception time relative to the sending time for all the nodes that were ever in the delivery zone. The solid line is the expected reception deadline for nodes in each location, i.e, the first time they are expected to enter the delivery zone. The star dotted line is the actual reception time of the mobicast packet for each node. For comparison, we also included a simulation result (the diamond dotted line)

of a spatial multicast on the same path with "as soon as possible" delivery. (Note that in this case the spatial propagation speed exceeds 1600m/s, i.e., 800m is traversed in less than half a second). We can clearly see the temporal locality property of mobicast. The packet reception time is very close to the deadline specified by the delivery zone semantics. These results also show the benefit of mobicast over a more conventional spatial multicast



(a) (b)

Figure 3.7: Slack Time of Mobicast Delivery

like geocast, which assume implicit as-soon-as-possible temporal delivery semantics, i.e, using mobicast one can control information propagation speed to better satisfy application needs without overwhelming spatiotemporally unrelated nodes. We believe this "just-in-time" delivery nature of mobicast is a powerful mechanism for optimizing resource utilization for related applications in sensor network.

3.4 Adaptation to Local Compactness

So far we have discussed two ways to improve mobicast in uniformly distributed sensor networks: (1) Making the network more compact by choosing an optimal node density; (2) Using an optimistic choice for the forwarding zone size by slightly relaxing the delivery guarantee. A third mechanism likely to improve mobicast is to let the protocol adapt to a local notion of compactness. For instance, when the mobicast delivery zone moves from a more compact area into a less compact one, its forwarding zone will expand. If the delivery zone moves from a less compact area into a more compact one, its forwarding zone should shrink. The local notion of compactness can be area-based or per-node based. For instance, one may partition the space into a grid and compute

the compactness value for each grid area. Or one may let each node probe up to certain depth into its neighborhood and compute the compactness around it. The latter is what we use in our adaptive mobicast simulation which is presented next. This type of local compactness adaptation is beneficial when the network is more compact in one area and less so in others and the difference in compactness is relatively large. Yet, our preliminary investigation shows that for random networks with uniform distribution, all areas have similar compactness. Figure 3.8 shows one example of the spatial distribution of local compactness for a uniformly distributed network of size 1000x400 (as the one in Figure 3.5(a)). The local compactness is computed for each node to a depth of five hops and within a 100 meter radius. One can see that for uniformly distributed networks, local compactness also appears to be uniformly distributed spatially. This suggests that in such networks, local adaptation may not provide significant improvement for mobicast with respect to its efficiency. For networks that have distinct and relatively long range

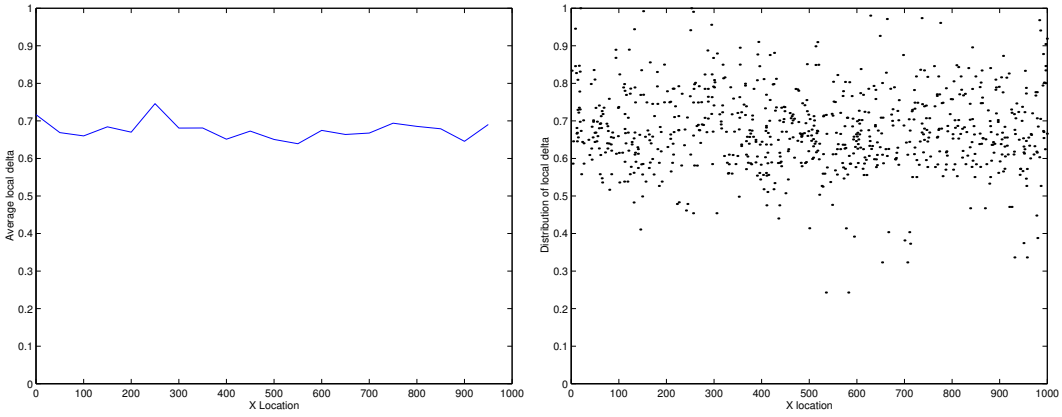


Figure 3.8: Local Delta Distribution Over Space (Uniform Distribution)

spatial variations in compactness, adaptive protocols may prevail.

In order to verify the above observations, we modified the previous mobicast (Figure 2.6) and made it adaptive in the following manner. First, the header is augmented with an extra field: location (x,y), which is used for recording the location of the most recent delivery-zone node that handled the packet. In the adaptive mobicast protocol, when receiving a packet, a delivery zone node will replace the delta value in the packet by its local delta value before forwarding it. This is an attempt to inform other downstream nodes about its view of the network compactness. A node finding itself not to be a delivery-zone node upon receiving a mobicast packet will use the delta in the packet to determine its own forwarding status (forward immediately, schedule for future forward, or drop the packet). If a node not in delivery zone path finds that it is a forwarding-zone node, it forwards the packet but does not make any change to the delta value in the

packet. The reason for this forwarding behavior difference between delivery-zone nodes and non-delivery-zone nodes is that the purpose of forwarding is to guarantee the mobicast delivery for the delivery zone nodes. The potential path distortion between delivery zone nodes is expected to be captured by the local compactness values of the delivery zone nodes. Depending on the network topology, there are cases where a forwarding node is relatively far away from the delivery zone and having a very different neighbor topology. By replacing the delta with its own, downstream nodes might be mis-informed.

Our preliminary simulation results of this adaptive mobicast protocol in ns-2 are as follows. For the uniformly distributed network topology such as the one in Figure 3.5, the adaptive protocol appears to guarantee 100% delivery[2], but the delivery overhead is about two radio transmissions per delivery, which is relatively high. The reason is, as we pointed out earlier, that holes are ubiquitous in the random network and most of the nodes have some worst form of path distortion in their neighborhood. Furthermore, the collective behavior of the protocol forms a forwarding zone whose size in determined by the smallest compactness in each subarea. This in turn causes more nodes to participate in the forwarding.

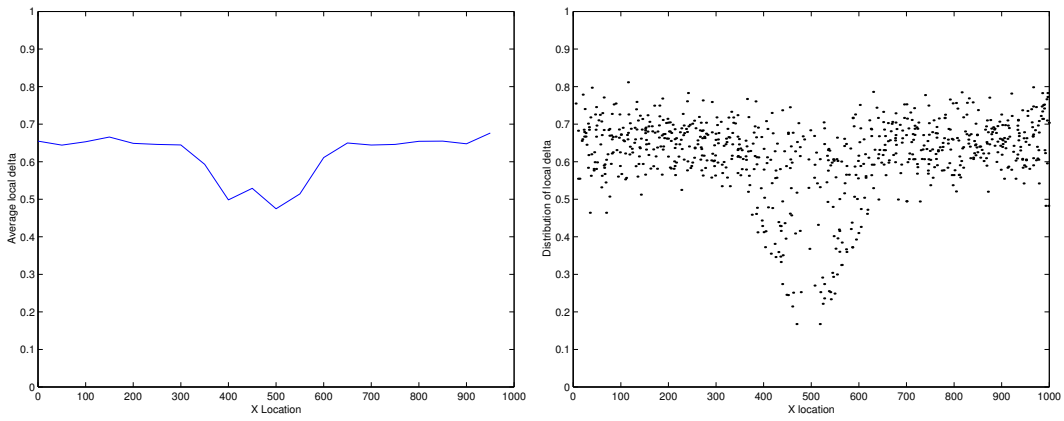


Figure 3.9: Local Delta Distribution Over Space

To examine the behavior of our adaptive mobicast protocol in more detail, we handcrafted a well-connected network in a 1000x400 area with 800 nodes, with a big hole at the center, as shown in the top side of Figure 3.10. For this topology, the compactness value for the nodes close to the hole is distinctively smaller than the nodes close to both ends, as shown in Figure 3.9. The bottom picture of Figure 3.10 shows a mobicast session run from a node on the left to the right. The delivery zone has a radius of 45, and moves

[2]Note that we only experimented on a limited number of configurations, this 100% is not equivalent to an absolute guarantee. Actually, in theory, sometimes this adaptive protocol will not have 100% delivery, because the local compactness value only captures the topology distortion in a limited scope.

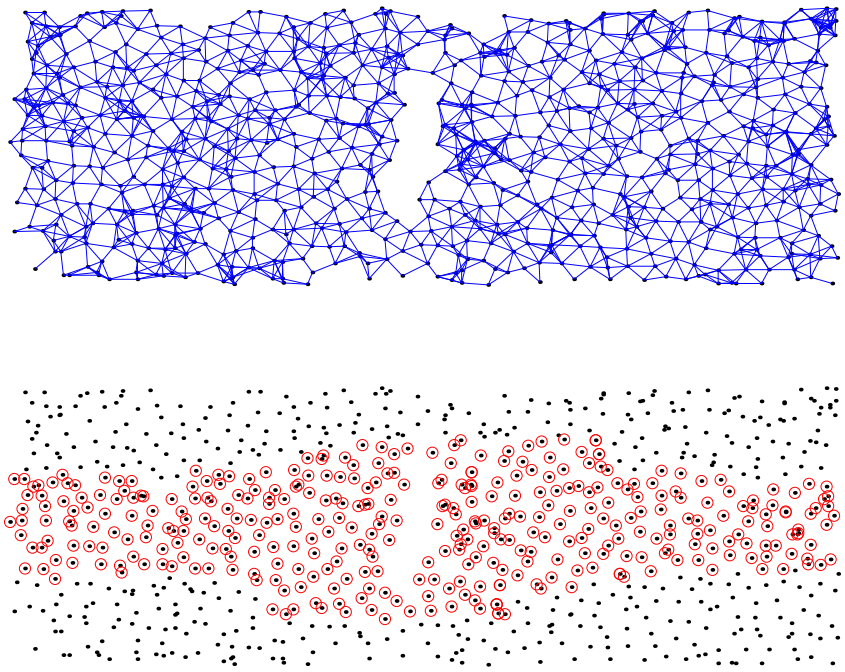


Figure 3.10: Example of Adaptive Mobicast

at a speed of 40 m/s with a session length of 20 seconds. The circled nodes are those participated in the mobicast session. One can see that the protocol adapts to the local topology, and achieves 100% delivery, even in the presence of a big hole on its path. The radio transmission overhead is less than 1.2 transmissions per delivery in the above case (195 extra radio transmissions for 164 delivery zone node deliveries) comparing to the non-adaptive FCF mobicast protocol which has an overhead of more than 3 radio transmissions per delivery. This appears to validate our observation that the adaptive protocol excels in a non-uniformly distributed sensor network.

3.5 New Questions

The adaptive mobicast protocols glossed over the following question: if all the nodes collected multi-hop neighbor location information and used them for computing local compactness values, why not use the location knowledge for better routing in addition to the less informative compactness value for routing? Using more topology information can potentially lead to more efficient routing methods. It is hard to imagine cases where the a network node has only enough temporary storage for the purpose of computing local compactness values yet not enough permanent storage for the collected neighborhood information.

These observations led us in turn to investigate more efficient adaptive algorithms for mobicast by using more information about network neighborhood. This investigation is presented in the next chapter.

Chapter 4

Face-Aware Routing and Spatial Neighborhood

This chapter presents a new Face-Aware Routing protocol (FAR) for mobicast and a related spatial neighborhood discovery algorithm. FAR distinguishes itself from previous mobicast protocols by providing both reliability and scalability at the same time. Its scalability comes from the fact that it does not rely on any global topological information, and each node makes local forwarding decisions based on its *spatial neighborhood* configuration, which is found to be small in average case via both theoretical analysis and simulation for random wireless ad hoc networks. We also prove in theory that FAR can reliably deliver a mobicast message to all nodes that ever enter the delivery zone.

The idea of face routing is inspired by previous geometric routing algorithms such as GPSR [98] and GOAFR$^+$ [106]. They all have a face routing component to help their greedy forwarding component get out of local minima in their unicast message forwarding path. However, these unicast protocols can not be applied directly to mobicast, due to two key problems. In unicast, the destination node is known, and so is its location in geometric routing scheme. The location of the destination node is key in determining the forwarding path and in detecting whether the greedy algorithm has gone into a local minimum. In mobicast, however, there is no single destination location, only the delivery zone is known, the exact locations of nodes in future delivery zones are not known. Simple approaches such as selecting some arbitrary location in the delivery zone path as a destination and use unicast protocols to reach the destination and dispatch the message to nodes close by does not work, since without a global node-location look up service, one can't even tell if a particular location has a node at or near its position. Moreover, the mobicast delivery zone is not fixed. A mobicast protocol must consider the temporal domain of information dissemination, which none of the previous geometric

unicast protocols address. The FAR protocol addresses the first issue via some knowledge about its *spatial neighborhood* (to be defined later), and addresses the second problem by a novel timed face routing strategy.

For clarity, we first assume that the network is a planar graph. In general, a random wireless network may not be planar. Later we also discuss graph planarization methods and how the FAR algorithm can be modified to deal with a non-planar graph. We also assume that each node knows all of its spatial neighbors and their locations. We provide an algorithm for obtaining this information and discuss the cost for storing such information in later sections. We first define the concept of a spatial neighborhood.

4.1 The Planar Spatial Neighborhood

On a planar graph, each node has one or more adjacent faces. A face is a maximal connected subset of the plane that does not contain a point on an edge or a vertex [51]. For instance, in the planar graph as shown in Figure 4.1, node A has six adjacent faces, and node B has four adjacent faces. Note that the "boundary node" M has two adjacent faces. One of them is the "inner" face formed by nodes M, L, G and H, the other is the "outer face" formed by nodes M, H, I, J, K, F, E, D, C, N, O and L. Note also that even though the "boundary", "inner" and "outer" faces of a planar graph seem visually easy to identify, topologically it is hard to define and distinguish them. This has important consequences for face-based geometric routing mechanisms. We will discuss this issue later.

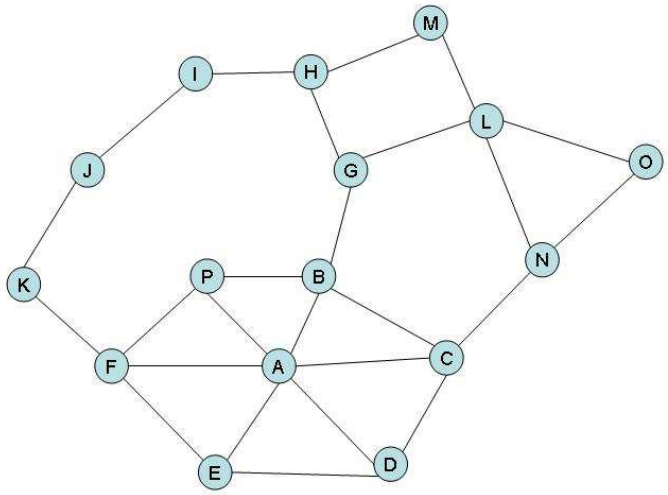


Figure 4.1: Planar Graph and Planar (Spatial) Neighborhood

We define the "spatial neighborhood" of a node in a planar graph to be the set of nodes in all faces adjacent to that node except the node itself. So in Figure 4.1, node A has six spatial neighbors (B,C,D,E,F and P) which are the same as its immediate graph neighbors. Yet node G has 10 spatial neighbors (L, H, B I, J, K, F, P, C and N) while it only has three immediate graph neighbors(L, H, B). Note that the spatial neighborhood of a node X as we define it represents the set of nodes that can be reached from X without crossing an edge or other nodes, and in general is equal to or greater than the immediate graph neighborhood.

The spatial neighborhood information plays an important role in our face-based geometric forwarding strategies, just like immediate network neighborhood information is very useful for many routing algorithms. In our face-ware routing protocol, each node maintains the spatial neighborhood information as a routing table. When a node receives a new mobicast message is received, it builds a *care list* which contains itself and those of its spatial neighbors that are delivery zone nodes.

4.2 Face-Aware Routing

We now describe the face-aware routing algorithm. The essence of the algorithm is very simple: every node that has at least one spatial neighbor that is a delivery-zone node will forward (locally broadcast) the mobicast packet once[1]. We will prove that this simple rule can guarantee all delivery zone nodes will receive the corresponding packet. Yet using this simple rule alone leads to an "as-soon-as-possible" style mobicast protocol that exhibits a high average slack-time which is not desirable. We need certain temporal controls to achieve a just-in-time style mobicast protocol. As a result, the face-aware algorithm consists of two methods for forwarding packets: *greedy forwarding* and *timed forwarding*. Before discussing these two methods in detail, we first presents the format of a FAR mobicast packet.

4.2.1 Packet Format

Each FAR mobicast packet contains the following information in its header:

- sender location
- packet sending time

[1]An optimization can change this to "forward the mobicast packet once, if necessary". We try to keep it simple here and leave the optimization issue aside for the moment.

- initial delivery zone coordinates
- delivery zone velocity
- message lifetime
- message type
- sender packet sequence number
- last forwarder location

Similar to previous mobicast protocols, we do not assume each node has a unique ID. The sender location, the packet sending time stamp and the sender packet sequence number are jointly used to identify each packet on the network. The initial delivery zone field contains an ordered sequence of locations corresponding to the initial vertices of the delivery zone. For a circular delivery zone, the radius and the initial center are recorded instead. The message type field is used for indicating the type of delivery zone, e.g., rectangle, pentagon, circle, ellipse, etc. The initial delivery zone coordinates combined with the delivery zone velocity and packet sending time, can be used to determine the location of the delivery zone at any point time in the future. The message lifetime is used for terminating each mobicast session. The last forwarder information is used for determining if further forwarding of a packet is needed. We will discuss these in more detail.

For simplicity, henceforth we assume each mobicast message fits in one packet, and we use the words packet and message interchangeably.

4.2.2 Greedy Forwarding

Greedy forwarding applies to all nodes that are currently (or previously)[2] covered by the mobicast delivery zone, or have at least one spatial neighbor that is currently (or previously) covered by the mobicast delivery zone. In such cases, a node forwards a new packet in an "as soon as possible" fashion.

Figure 4.2 depicts an FAR mobicast example featuring a rectangular delivery zone moving to the right at speed v at a certain time instance. In this example, the greedy forwarding rule applies to nodes A,K, D, B, C, J, as they are either in the delivery zone

[2]The "previously" condition is unlikely to happen in most cases. Yet it is necessary here as our specification of mobicast requires all delivery zone nodes to receive the packet. If a new specification only admits on-time reception, then this notion of "previously" is unnecessary.

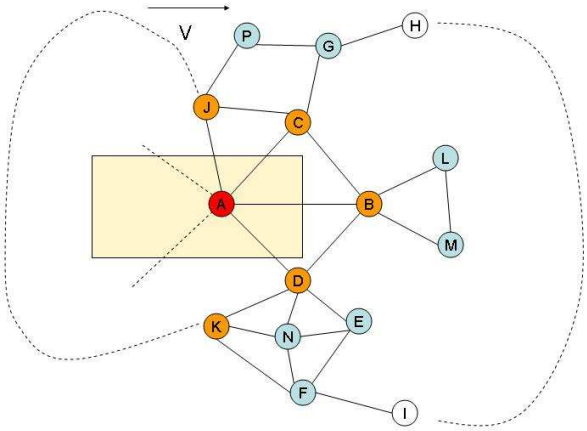


Figure 4.2: Greedy and Timed Face-Aware Forwarding

or have a spatial neighbor that is in the current delivery zone. Note that the condition specifies a spatial neighbor rather than a direct neighbor, which causes K to be included.

Note that after K, D, B, C, J perform local broadcasts, nodes P,G,L,M,N,E and F all hear the mobicast message since they are each connected to at least one of the previous broadcasting nodes. But because P,G,L,M,N,E and F do not have spatial neighbors in the current delivery zone, they do not perform the greedy forwarding, and use timed forwarding instead if they have spatial neighbors to be in the delivery zone.

4.2.3 Timed Forwarding

Timed forwarding applies to a node that has no spatial neighbor in the current delivery zone but either will itself soon be in the delivery zone or will have at least one spatial neighbor that will be in the delivery zone. Nodes H, G,L,M,E, F and I in Figure 4.2 belong to this category. Nodes L and M are to be in the delivery zone themselves as to delivery zone move to the right. Nodes G, E and F find three of their spatial neighbors, B, L and M are to be in the delivery zone. Nodes H and I will discover the same after hearing the mobicast packet from G and F.

The timed forwarding method works as follows. If a node X receives a new mobicast packet at time t and finds itself in the timed forwarding category, it makes a forwarding decision based on the relative times that the delivery zone reaches its delivery zone neighbors and the expected communication latency between itself and those neighbors.

Let $Y_1, Y_2, ..., Y_k$ be the ordered list of all spatial neighbors of X that will be in the delivery zone and $\Delta t_1, \Delta t_2, ..., \Delta t_k$ be the corresponding times for the delivery zone

to reach them. Let $h_1, h_2, ..., h_k$ be distance from X in the number of hops. Let τ_1 be the expected 1-hop network latency. We then have $h_i\tau_1$ as the expected communication latency between X and Y_i. Let T_a be the minimum time difference between the time for the delivery zone to reach Y_i and the expected latency $h_i\tau_1$ for a message sent from X to reach Y_i. i.e.,

$$T_a = min\{h_i\tau_1 - \Delta t_i | i = 1, 2, ..., k\} \tag{4.1}$$

The forwarding decision of X is as follows:

1. If $T_a \leq 0$ forward the packet as soon as possible;

2. If $T_a > 0$ schedule a forwarding at time T_a from now.

In Figure 4.2, nodes H, G, C, B, L, D, M, E, F and I share one face which extends to the east. Among them, nodes C, B and D have already greedily forwarded the packet. Nodes G, L, M,E and F have heard the packet and will schedule the forwarding according to the timed forwarding rule. From this example, one can also see that this face forwarding algorithm can be improved. For instance, nodes L and M do not need to do the forwarding at all since their local broadcast effort does not help the mobicast packet reach any new node, and they have the local topology knowledge to learn that. Node G knows node B has received the message as it heard it from C, and B, C are connected. Note that G does not know if B has re-broadcast the packet but does know B will take care of L and M. So G may take B,L,M off its "care list", i.e, the list of nodes used for computing the forwarding time. A similar argument is true for E. We will discuss optimization methods later.

Note that node F is a different case than G or E. It has heard the packet from node K, and it does not know if E has heard the message, or D, B, M, L, C, G. So its care list has to include B, L, and M. Note also that even though B is the earliest among its spatial neighbors to enter the delivery zone, F can not simply compute its forwarding time based on B, since $h_{FL}\tau_1 - \Delta t_L$ may be smaller than $h_{FB}\tau_1 - \Delta t_B$.

In the previous discussion, we choose τ_1 to be the expected 1-hop latency. If one choose τ_1 to be the maximum 1-hop latency, the protocol will result in higher average slack time but less potentially late receptions.

Since every node makes the forwarding decision locally, it is possible for a node to receive a packet it has forwarded earlier. In this case a node simply ignores the packet. For a node to be able to determine which packets are new and which are old, every node maintains a local cache to log received packets. This cache is periodically checked, and packets that have expired are removed.

Note that in Figure 4.2, although node N has heard the packet, it will never forward the packet since it has no spatial neighbor that is a delivery zone node. This is also true for node P.

4.2.4 Protocol Termination

In addition to greedy forwarding and timed forwarding, the algorithm also has a mobicast termination method based on the packet life time value in the packet header. A packet is not simply ignored if it has expired. An expired packet is dropped only in the timed forwarding mode, i.e., when the recipient node finds that no node in its care list is in any previous delivery zone. If a node is in greedy forwarding mode, it will forward the packet even if the packet has expired. This choice intends to tolerate some level of timing uncertainty by admitting marginal overhead caused by potential "expired face forwarding" in the last few faces in the delivery zone path. This also simplifies our statements and proofs of the delivery properties of the protocol later.

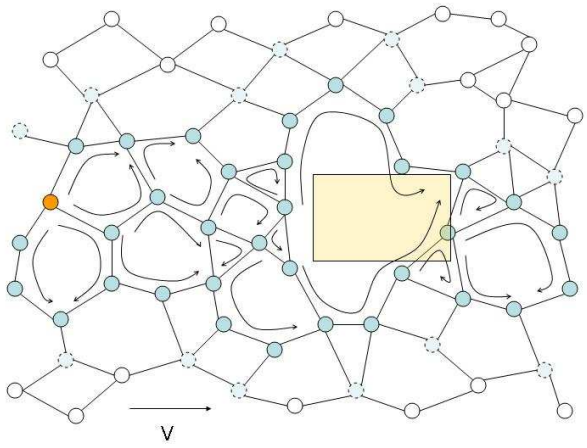


Figure 4.3: Bird's Eye View of the FAR Protocol Behavior and Result

To give a bigger picture of the behavior and results of the FAR algorithm, Figure 4.3 schematically shows a rectangular mobicast history in a larger network context. The faces with arrows are those that have experienced face-forwarding. The solid circles represent the nodes that have forwarded the packet. The lightly shaded dashed circles represent those that have heard (received) the packet but did not forward it. The empty circles never hear the packet. One can see that the face-aware forwarding algorithm creates a localized forwarding cloud (area) surrounding the mobile delivery zone, and the forwarding area adapts to the topology on the delivery zone path and makes the delivery

zone cross holes in the network. Next we prove that our forwarding strategy indeed delivers mobicast packets to all its expected recipients under one reasonable assumption.

4.3 Delivery Guarantee

Our FAR algorithm guarantees the delivery of a mobicast packet to all its delivery zone nodes, under the following assumption on the size of the forwarding zone: the forwarding zone span on the direction perpendicular to the mobicast velocity direction (we call it "perpendicular span" henceforth) must be no smaller than the maximum neighbor distance. (In wireless ad hoc networks, this may be interpreted as the perpendicular span to be no smaller than the maximum communication range). If the perpendicular span is too small, the algorithm may terminate prematurely. Figure 4.4(a) shows such an example in a partial network. Nodes J,C,G and K will not forward the packet because they have no spatial neighbor that is a delivery zone node. This results in E, a delivery zone node, never receiving the packet. Note that the constraint is only on the perpendicular span of the delivery zone. Small delivery zone size on the velocity direction is acceptable, as one can see in Figure 4.4(b), in which the mobicast will not stop prematurely. Next we prove

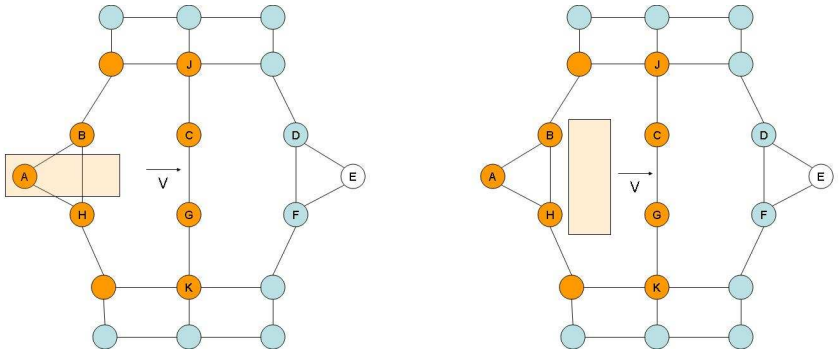


Figure 4.4: FAR Assumption

this delivery guarantee in the general case. We start from the following lemma.

Lemma 1 *If X and Y are in the same face and X is a delivery zone node, the FAR protocol guarantees that if Y has received the mobicast packet, X either has received it or will receive it.*

Proof: Assume that X has not received the packet. X will at some point in time be in the delivery zone. The fact that Y has received the packet means it has the data for computing the delivery zone trajectory over the packet lifetime. Y also has the knowledge

of the locations of all its spatial neighbors which include X. So Y can compute whether X either was previously, is currently or will be in the delivery zone.

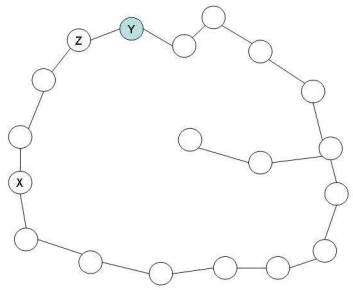


Figure 4.5: FAR on a Face

Without loss of generality, let Y be the closest (among the nodes that have received the packet) in terms of hops to X on the face under consideration. If Y finds X was previously in the delivery zone or is currently in the delivery zone, it will do a local broadcast as soon as possible according to the FAR protocol. Note that one of Y's direct neighbor is closer to X in terms of hops than Y is (e.g., node Z in Figure 4.5). As a result, when the neighbors of Y hear the packet, the packet has at least moved one step closer to X. The same argument applies to the closer neighbor(Z). The mobicast packet moves a node closer to X in each step, until the distance is zero, when X receives the packet.

If Y finds that the delivery zone will reach X some time in the future, it will schedule a forwarding at the appropriate time according to the FAR protocol. The same "one step closer" argument applies. ■

Using Lemma 1 we can prove the following theorem regarding the FAR protocol.

Theorem 7 *In a connected network, FAR guarantees that all delivery zone nodes will receive the mobicast message if the initial delivery zone contains the source node.*

Proof: We prove the theorem by contradiction. Let B be a delivery zone node that missed the packet. Being a delivery zone node, B must be located inside the integral delivery zone (the union of all delivery zone areas over the packet's lifetime), as shown in Figure 4.6 in which the long dashed rectangle represents the integral delivery zone. Let A be the source node. Let $X_1, X_2, ..., X_k$ be the set intersection points between the line segment $\overline{AB}$ and the graph edges, in order from A to B.

If B missed the packet, none of the two end points of edge e_k would have received the packet. Otherwise, by Lemma 1, B should receive the packet because e_k and B are

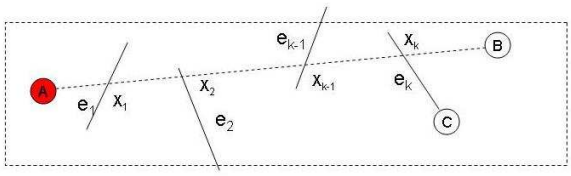


Figure 4.6: Delivery Accuracy of the FAR protocol

around the same face(they are around the same face because there is no edge between X_k and B). Note also that at least one of the endpoints of the edge e_k is in the delivery zone because the height of the integral delivery zone is equal to the perpendicular span of the delivery zone, which is assumed to be larger than the edge length. Let this end point of e_k be C. C being a delivery zone node that missed the message leads to the same argument that none of the endpoints on edge e_{k-1}, in turn $e_{k-2}, ..., e_2, e_1$, has received the message. Yet, e_1 and A are around the same face, and by Lemma 1, this is not possible, because, as the source node, A must have the message, so the message would have traversed e_1. ■

The FAR algorithm assumes all nodes have locally accessible information about their spatial neighbors. An important question is: how big is the spatial neighborhood in general? The answer to this question will shed light on the question of how much memory and storage the algorithm needs, which is very important in protocol and system design. Another important question is: how big is the average face size? The answer to this question relates to the forwarding overhead of the FAR protocol. We address these issues in the next section.

4.4 Cost Analysis

In this section we explore two cost metrics of FAR: (1) the memory space needed for the spatial neighborhood information, and (2) the communication overhead due to the traversing of face nodes that are not in the delivery zone. We start from an investigation of the average face size, average node degree on planar graphs and average spatial neighborhood size via geometric analysis, and conclude with simulation results from random networks.

4.4.1 Spatial Neighborhood Size

Average Face Size

The size of a face is defined by the number of vertices surrounding the face. The following theorem states a bound on the average size of faces on the planar graph.

Theorem 8 *Given a planar graph $G(V, E)$, the average size of a face is*

$$\overline{S_f} \leq \frac{2n_e}{n_f} \tag{4.2}$$

where n_e and n_f are the numbers of edges and faces of G respectively.

Proof: Let $s_1, s_2, \cdots, s_k$ be the sizes of all the faces of graph G. We have $k = n_f$ and the total number of edges on all the faces is

$$s_1 + s_2 + \cdots + s_k \leq 2n_e \tag{4.3}$$

the 2 appears in the equation because each edge is counted at most twice (once on each side). Note that dangling edges are counted only once, resulting in an inequality rather than an equality expression.

The average number of edges on each face is

$$\overline{S_f} = \frac{s_1 + s_2 + \cdots + s_k}{k} \tag{4.4}$$

Combining equality 4.4 with inequality 4.3, we immediately have

$$\overline{S_f} \leq \frac{2n_e}{n_f}$$

■

Next we derive a bound for S_f in terms of the number of nodes and edges rather than edges and faces. This is more desirable because it is straightforward to count the number of nodes and edges in a graph and it is not very obvious how to count the number of faces.

Corollary 2 *Given a planar graph $G(V, E)$, the average size of a face is*

$$\overline{S_f} \leq \frac{2n_e}{n_e - n_v + 2} \tag{4.5}$$

where n_v and n_e are the numbers of nodes and edges of G respectively.

Proof: From Euler's formula[51], we have the following relation between nodes, edges, and faces of any planar graph:

$$n_f + n_v - n_e = 2 \tag{4.6}$$

Use Theorem 8 and the Euler's formula, we get

$$\overline{S_f} \leq \frac{2n_e}{n_e - n_v + 2}$$

■

Average Node (Face) Degree

So far we have derived an upper bound for the average face size. Another question is how many faces each node has. The next lemma helps lead to an answer.

Lemma 2 *On a planar graph $G(V, E)$, the edge degree of a node is always equal to a greater than its face degree. That is, let de_i be the edge degree of node i, and df_i be the face degree of node i. We have the following inequality*

$$de_i \geq df_i \tag{4.7}$$

Proof: For each node i, sort its edges in clockwise or counter-clockwise order. There is at most one face between adjacent edges. Note that it is "at most" because of potential dangling edges which do not create new faces. ■

Using Lemma 2, we can derive the following theorem

Theorem 9 *The average number of faces D_f each node has in a planar graph $G(V, E)$ is upper bounded by the following expression*

$$\overline{D_f} \leq 2\frac{n_e}{n_v} \tag{4.8}$$

where n_v and n_e are the numbers of nodes and edges of G respectively.

Proof: Let de_i and df_i be the edge and face degrees of node i respectively. Then the sum of degrees across all nodes is

$$\sum_{i=1}^{n_v} de_i = 2n_e \tag{4.9}$$

because each edge is counted once on both ends.

From Lemma 2, we also have the sum of face degrees to be no greater than the sum of edge degrees

$$\sum_{i=1}^{n_v} df_i \leq \sum_{i=1}^{n_v} de_i \tag{4.10}$$

This leads to

$$\overline{D_f} \equiv \frac{\sum_{i=1}^{n_v} df_i}{n_v} \leq \frac{2n_e}{n_v} \tag{4.11}$$

■

Average Spatial Neighborhood Size

From Theorem 8 and Theorem 9, we may estimate the average spatial neighborhood size ($\overline{\Upsilon}$) as follows. Let $\overline{D_f}$ be the average number of faces of each node, and $\overline{S_f}$ be the average face size. $\overline{S_f} * \overline{D_f}$ may be used for estimating the average number of nodes in all faces adjacent to each node if the variances in face sizes and node degrees are not high[3]. This leads to

$$\overline{\Upsilon} \sim \frac{4n_e^2}{n_v(n_e - n_v + 2)}$$

Considering the double counting of nodes in adjacent faces, this estimation can be improved. The double counted nodes, say, with respect to node G in Figure 4.7,

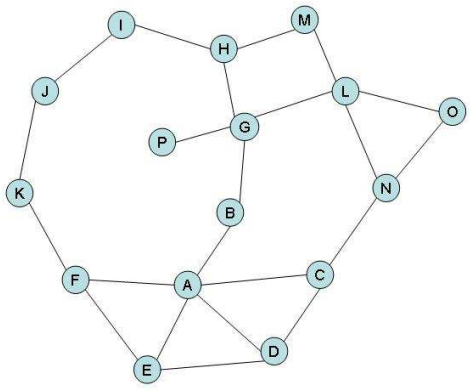


Figure 4.7: Planar (Spatial) Neighborhood

include the following three kinds: (1) the node G itself, being counted twice (once on each adjacent face); (2) immediate double-faced neighbors of G: H, L, B (note that even

[3]Note that $mean(x_i y_i)$ does not equal to $mean(x_i)mean(y_i)$. But these two quantities have close values when all x_i's are close to $mean(x_i)$ and all y_i's are close to $mean(y_i)$.

though P is an immediate neighbor of G, it was not counted twice as it belongs to only one face); (3) non-immediate double-faced neighbors such as node A. We know that on average, the first kind of double-counting occurred $\overline{D_f}$ times, and the second kind also occurred $\overline{D_f}$ times. So there were at least $2\overline{D_f}$ double counting of nodes in $\overline{S_f D_f}$. This leads to

$$\begin{aligned} \overline{\Upsilon} &\sim (\overline{S_f} - 2)\overline{D_f} \\ &\sim \frac{4n_e(n_v - 2)}{n_v(n_e - n_v + 2)} \sim \frac{4n_e}{n_e - n_v + 2} \end{aligned} \tag{4.12}$$

Figure 4.8 plots this estimation of spatial neighbor size against the relative edge to node ratio of a graph. We can see that, given a fixed number of nodes, more edges

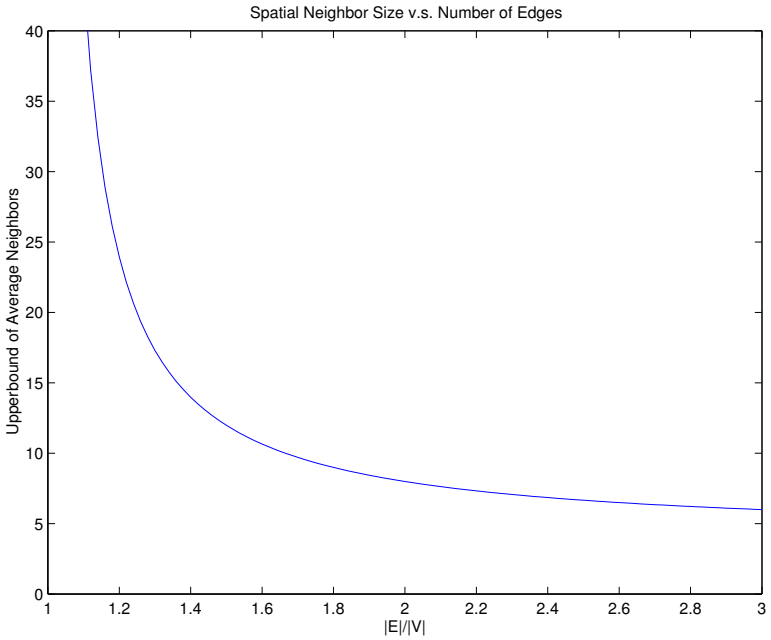


Figure 4.8: Average spatial neighbor Size Estimation

means a smaller spatial neighborhood. In other words, the "denser" the graph is, the smaller its average spatial neighborhood is. Note that planar graphs have a limit on the number edges they can have. A well-known corollary of Euler's formula states that for a planar graph, the number of possible edges it can have is bounded above by

$$n_e \leq 3n_v - 6 \tag{4.13}$$

The curve in Figure 4.8 also suggests that the size is around 6 when n_e/n_v gets close to 3.

An important insight from this analysis is that for random ad hoc networks with uniform distribution, the average spatial neighborhood size is likely to be on the order of 10. As alluded to earlier, the closeness of this estimate depends on the variations on face sizes and node degrees of the planar network. This average case approximation is good only when the variances are small, which is likely to be the case in uniformly distributed networks. Next we test this observation via simulation.

4.4.2 Statistical Face Size and Spatial Neighborhood Size Distribution of Planar Graphs

The goal of this section is to study the statistical distribution of face sizes in a planar graph. This statistical information complements our previous average case results for estimating memory cost for our FAR mobicast protocol.

Note that ad hoc communications networks such as random wireless sensor networks are often not planar graphs. A unit disk graph, in which two nodes have a common edge if and only if their Euclidean distance is less than a constant, is a widely used abstraction for wireless ad hoc networks. Figure 4.9 shows an example of a random unit

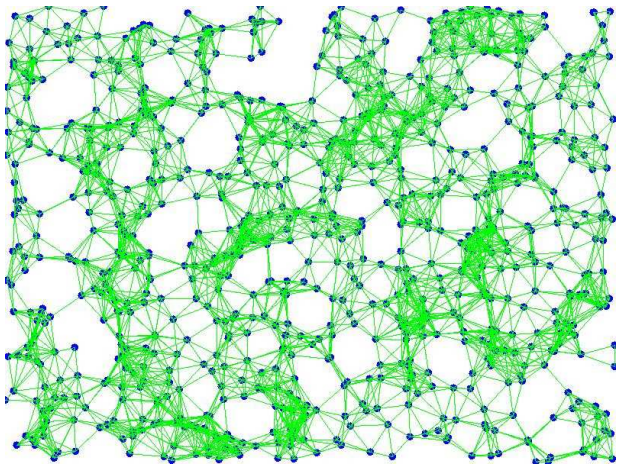


Figure 4.9: Unit Disk Graph

disk graph with nodes distributed uniformly in the 2-d space. One can see that there are many crossing edges, and such graph is thus not planar.

The FAR protocol uses the knowledge of spatial neighborhood defined by a planar graph. To let each node find out locally who its spatial neighbors are, we first need a method to planarize the network.

It is well known that the Gabriel Graph (GG) and the Relative Neighborhood Graph (RNG) [51][93] are planar graphs, and there exist simple distributed algorithms to

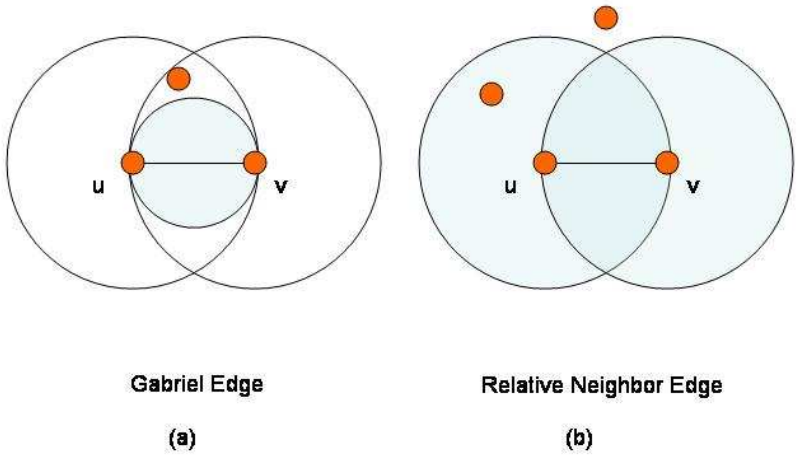


Figure 4.10: Gabriel Edge and Relative Neighborhood Edge

obtain a GG or RNG from a non-planar graph. In a geometric graph, an edge $e = (u, v)$ is called a "Gabriel edge" if there is no other node inside the disk which uses e as a diameter. An example is in Figure 4.10(a).

This definition immediately leads to the following simple distributed algorithm that computes a Gabriel graph [21]:

Algorithm: GABRIEL

1. **for** each $u \in Neighbor(v)$ **do**
2. **if** $disk(u, v) \cap (Neighbor(v) \backslash \{u, v\}) \neq \emptyset$ **then**
3. delete (u, v)
4. **end if**
5. **end for**

Where $disk(u, v)$ is the disk that uses edge (u, v) as a diameter. This algorithm requires $O(d^2)$ time, where d is the degree of node v. More efficient algorithms using a Voronoi diagram and Delaunay triangulation reduce the time complexity to $O(d \log d)$ [21]. It has been proven that this algorithm produces a connected planar graph if the original graph is connected. Figure 4.11 shows a result of running the algorithm on the network shown in Figure 4.9. The definition of RNG is similar to GG. An edge is a "relative neighbor graph edge" if there is no node in its *lune*, where the "lune" of an edge (u, v) is defined as the intersection of two disks D_u, D_v of radius length(u,v) centered at u and v respectively.

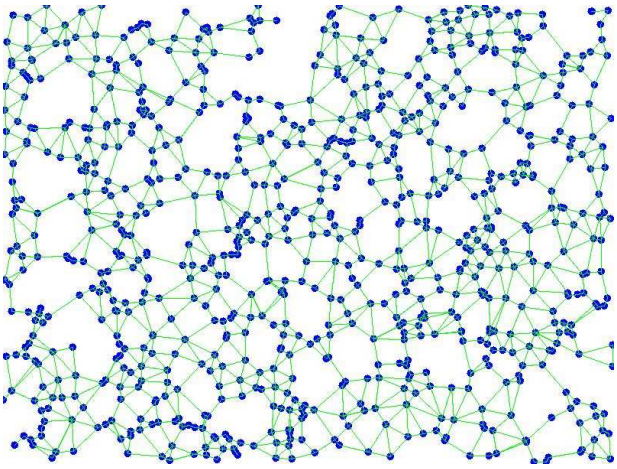


Figure 4.11: The Gabriel Subgraph of the Unit Disk Graph from Figure 4.9

Fig.4.10(b) shows an example of a relative neighbor graph edge. A graph is an RNG if all its edges are relative neighbor graph edges. More details regarding GG and RNG can be found in[93][98][106].

Note that for computing the Gabriel subgraph, immediate neighborhood locations must be known. This can be achieved by simple local beacons; the details are omitted here.

Face and Spatial Neighbor Statistics

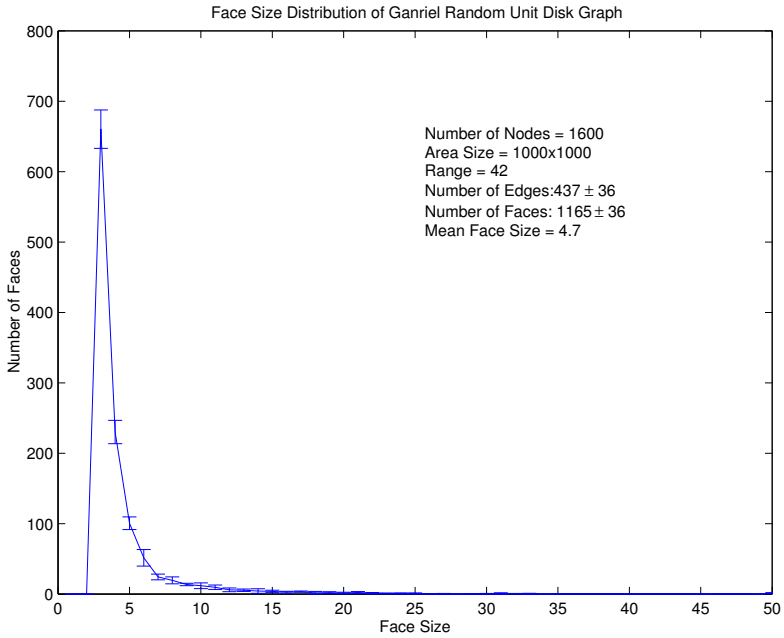


Figure 4.12: Faces Size Distribution of Random Gabriel UDG

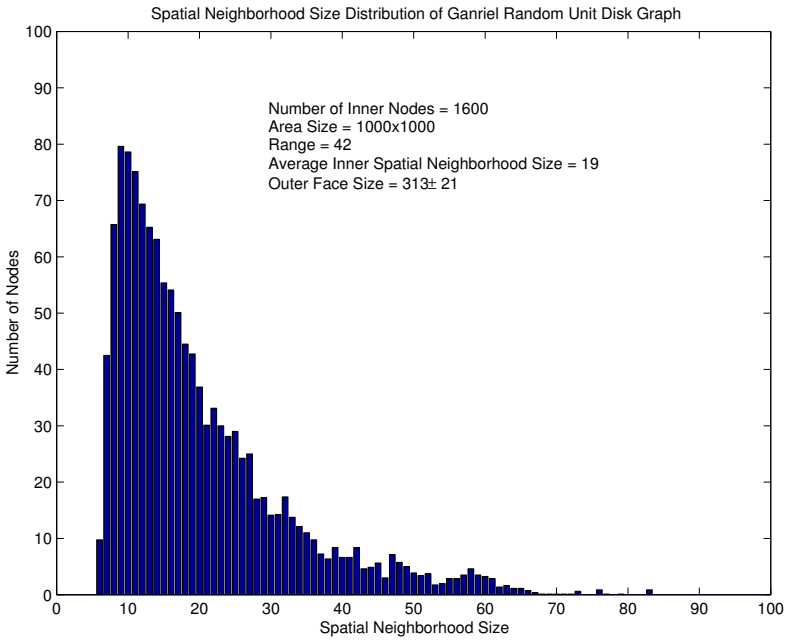


Figure 4.13: Spatial Neighborhood Size Distribution of Random Gabriel UDG

For random unit disk graphs, we found the average face size of their Gabriel subgraph and the average spatial neighborhood size are both on the order of 10. Figure 4.12 shows the face size distribution and Figure 4.13 illustrates the spatial neighborhood size distribution (of non-boundary nodes) obtained in our simulation. The results shown in these figures were averaged over 8 random unit disk graphs. All unit disk graphs were generated in a 1000x1000 area with 1600 nodes and a communication range of 42, which is very close to the critical range (40) for a connected graph. In this case the average face size is about 5 and the average spatial neighborhood size of non-boundary nodes in the Gabriel subgraph stays very close to 19. These results also indicate that, on average, if we use the Gabriel subgraph of a wireless ad hoc network, the memory needed for the FAR algorithm is very low. Furthermore, we also found that the average number of adjacent faces to a node is around 4 and does not vary much across the network. Figure 4.14 shows the distribution of the number of adjacent faces to a node in the graph. These results also suggest our earlier observation about the spatial neighborhood size to be valid.

Furthermore, we observe that when node density increases from the critical (connectivity) density (about 8 network neighbors per node in our experiments), the average face size quickly decreases, as shown in Figure 4.15. When the average number of network neighbors is beyond 14, the average number of spatial neighbors is smaller. This suggests in such cases most spatial neighbors of a node are within one hop[4]. Face-aware

[4]Note that direct neighbors are not necessarily spatial neighbors.

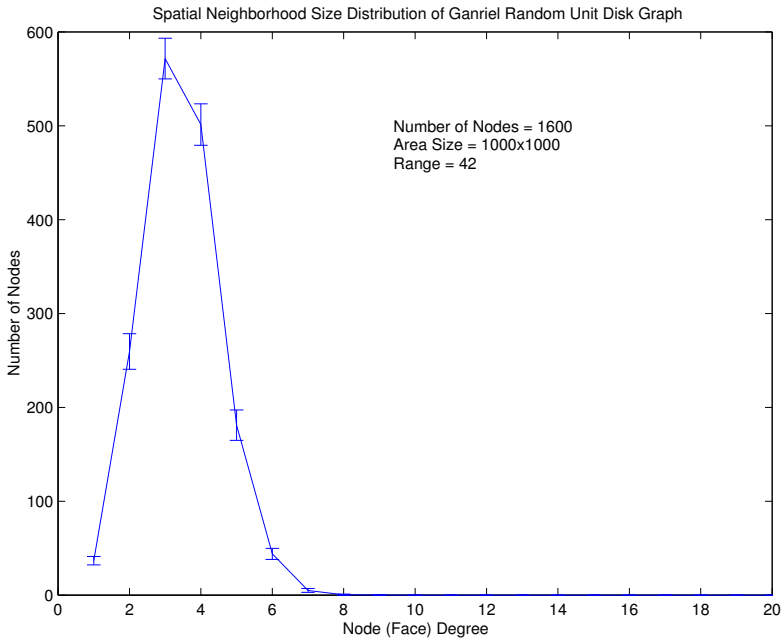


Figure 4.14: Node Degree Distribution of Random Gabriel UDG

forwarding is virtually reduced to local broadcast forwarding. The advantage of face-aware forwarding are expected to disappear from this point on, since there is not many big holes in high density networks.

This result is also consistent with our previous mobicast simulation results showing that in a high density network, delivery zone based greedy forwarding can achieve high delivery guarantee.

4.4.3 Message Overhead

The FAR protocol propagates the message on all faces that are inside or intersecting the path of the delivery zone. Its overhead can be measured by the number of non-delivery-zone nodes traversed per delivery-zone node delivery. Figure 4.16 shows our preliminary simulation results of this delivery cost on uniformly distributed random networks of 1600 nodes in a 1000x1000 area. The mobicast setting is a rectangular delivery zone moving at a velocity of $35m/sec$ for $20sec$. From Figure 4.16 we can see that given a fixed delivery zone width (i.e., the size perpendicular to the velocity direction), FAR overhead decreases with the increase of node density (in terms of average number of network neighbors). This is reasonable since a smaller density means larger holes, and FAR adapts to it and uses more nodes for successfully routing around the holes. Note also that given a network density, the per node delivery cost decreases when the delivery zone path is wider, as a result of amortization effects.

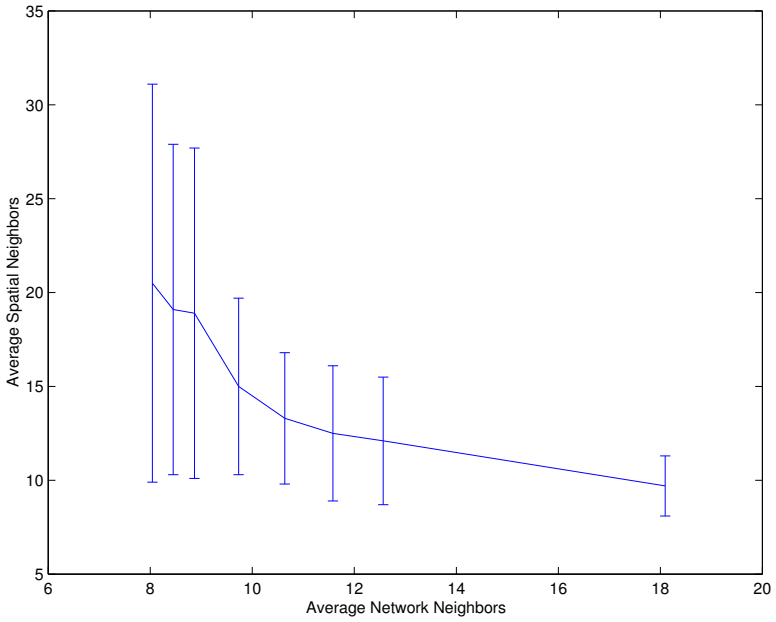


Figure 4.15: Spatial Neighborhood Size and Network Neighbor Size

4.5 Topology Discovery and Maintenance Protocol

In this section we present a protocol for spatial neighborhood discovery. This protocol features a sorted ring-buffer assisted right-hand rule, a randomization strategy and a location-based tie-breaking rule. It used the following result of the Gabriel planarization as a starting point: each node v not only knows who its immediate network neighbors are, but also which among them are its immediate planar neighbors, defined as the set of nodes whose edges to the node v remain in the Gabriel subgraph of the original connectivity graph.

The protocol essentially creates a discovery message flow in each face, as shown in an example in Figure 4.17. As a discovery message traverses a face, the coordinates of the nodes it has traversed are added to the message. After a discovery message finishes traversing a face, all nodes' locations on the face are collected and a message traverses the same face another time to inform everyone on the face of the complete discovery results.

There are four key problems that such a protocol needs to address: (1) Identification: how to make each discovery message traverse the correct face; (2) Termination: how to determine when a message has traversed the whole face; (3) Cost minimization: how to coordinate between nodes such that only one discovery message flows around each face; (4) Outer face limitation: the size of the outface is proportional to $\sqrt{N}$, where N is the total number of nodes in the network. When the network is very large, it is not

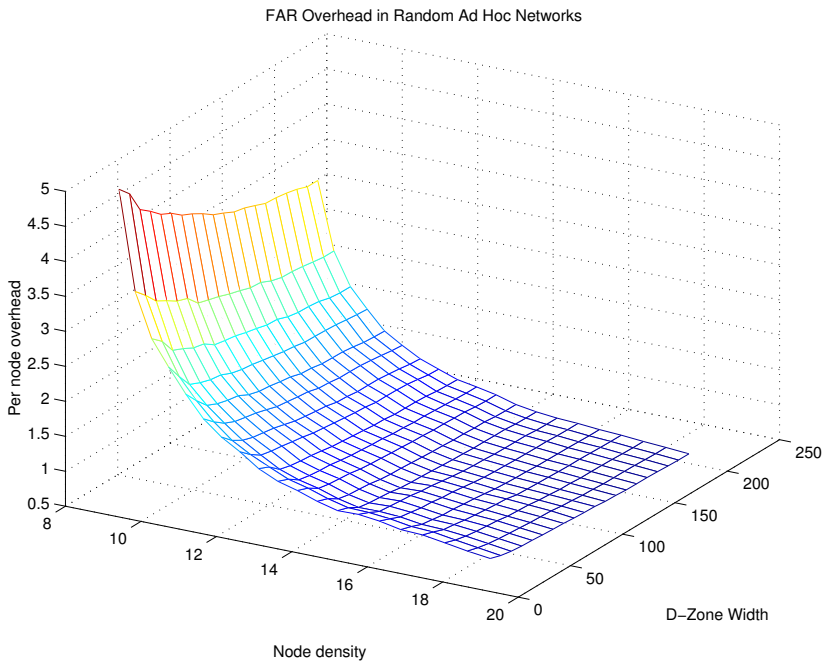


Figure 4.16: FAR Protocol Pernode Delivery Overhead

feasible and not reasonable to traverse this face, since a node shouldn't really concern itself with nodes on the other side of the network boundary.

We solve the first problem by using a ring-buffer on each node for storing the incident planar edges. The edges are directed (all viewed as outgoing edges from the node under consideration) and are sorted counter clock-wise. When a discovery message comes from one edge, it will be sent on the next edge in the ring-buffer. Each discovery message contains the next hop location and an ordered list of visited nodes' locations, so it can be used to identify the incoming edge and designate the outgoing edge. This simple direction sorted ring-buffer enables each node to always choose the right outgoing edge for each discovery message, and in such a way make a message traverse a face correctly.

Upon receiving a discovery message dm, a node determines whether dm has completed a full traversal of a face by the following criterion: the outgoing edge for dm is contained in its ordered traversal list. Note that a node can be traversed many times via a right-hand walk on a face. In turn, a simple termination rule such as "when the message come back to a node already traversed" does not work. For instance, in Figure 4.17, node G is traversed twice on the $\cdots$-H-G-P-G-B-$\cdots$ face, and B is also traversed twice on the $\cdots$-A-B-R-Q-B-G-$\cdots$ face. Note also that the edges should be viewed as directed edges, e.g., edge G-P and edge P-G should be viewed as different edges. If P gets a discovery message that contains a G in the message's ordered traversal list, it should not necessarily think that the edge P-G has been traversed by the message.

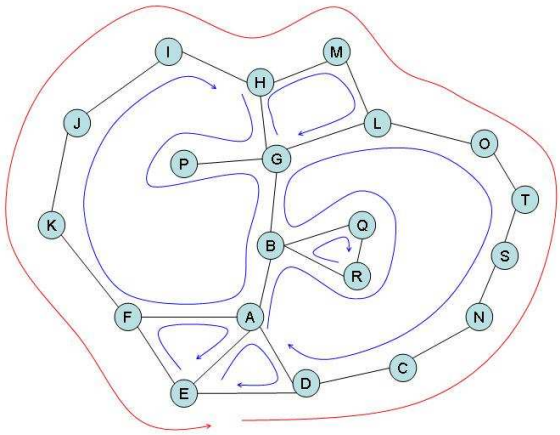


Figure 4.17: Right-hand Neighborhood Discovery Protocol

The cost of the discovery protocol will be unnecessarily high if every node has its own discovery message flowing on each face. On each face, ideally one traversing discovery message will suffice. The problem is, some kind of leader election mechanism is needed for each face to determine who should initiate the discovery message. However, leader election is not possible before the members are known.

We use two strategies for reducing the number of discovery messages. First, we use a random starting time to reduce the number of messages initiated on each face. On each node, an initial discovery message dm_i is scheduled at a random time for each of its faces f_i. The initial discovery message contains the next hop location and a list containing only the sender location. The initial scheduled discovery message dm_i will not be sent if the node receives a discovery message dm from its neighbor regarding the same face before dm_i's scheduled sending time. When this happens, the node simply appends itself to the ordered list in dm, resets the next hop destination in the message, and forwards it. This randomization method can eliminate some but not all unnecessary discovery message initiations. For instance, in Figure 4.18, A,B,and C may have all sent their discovery message for the same face (before receiving any from their neighbors). A tie-breaking strategy is needed to reliably reduce the messages to one. We use a starting location based tie-breaking rule: east is preferred, and if there is still a tie, then north is preferred. That is, if a node receives a discovery message initiated by others on the same face on which it has sent one, it will forward the message only if the initiator of this message is located east of him; if they are on the same east location (i.e., have the same x-coordinate), then only if the initiator is located north from it. When no two nodes have the same coordinates, this rule can uniquely identify one legitimate initiator and make each face have only a single discovery message remaining.

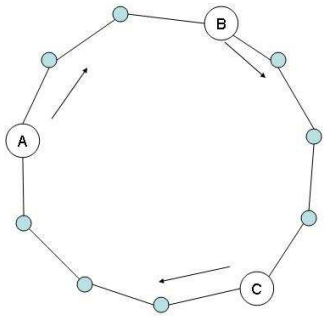


Figure 4.18: Location Based Tie Breaking for Elimination

The outer face problem is hard since there is no way to determine which face is the "outer" one without a global bird's eye view. The outer face and the inner faces are topologically indistinguishable. To see this more clearly, imagine the graph is on the surface of a sphere instead of a plane. The only way to identify the "outer" face is its size. This leads to our solution: a discovery message has a max hop count. If it reaches its hop limit, a flag is set and it will traverse back to the originator. By doing this, every "boundary node" learns a limited amount of spatial neighborhood information on the outer face. Obviously, this strategy also leads to a potentially incomplete traversal in any "inner" face that is large. The existence of a better strategy is an open question.

4.6 Discussion and Related Work

The FAR protocol is similar to geometric unicast routing protocols such as GPSR [98] and GOAFR$^+$ [106] from the perspective of using face routing as a mechanism for routing around potential holes in the network. There are significant differences. In GPSR and GOAFR$^+$, face routing is activated when a void is encountered. The detection of a void relies on the known location of the destination for detecting if a progress toward the destination has been made. For mobicast, however, there is no single destination location, only the mobile delivery zone is known; the exact locations of nodes in the delivery zone's path are not known. Therefore, the face routing activation criterion in both GPSR and GOAFR$^+$ does not apply. This is also why we use spatial neighborhood information as a routing table to help identify when face routing is necessary for mobicast. In FAR, each node is constantly aware of nodes in adjacent faces rather than exploring the nodes on the fly. Furthermore, the mobility of the delivery zone in mobicast invites consideration of

the temporal domain of information dissemination, which none of the previous geometric unicast protocols address.

The FAR protocol relies on the notion of spatial neighborhoods, and a smaller spatial neighborhood means that less memory is needed. This suggests that our protocol desires a planar graph with as many edges as possible. Given a non-planar graph, how to find its maximal planar subgraph is an active research subject. Recently Li, *et al.* [109] proposed a localized Delaunay graph $LDel$ which is denser compared to the Gabriel graph. Some other pointers to related research on maximal planarization can be found in [111].

The following optimization can further reduce the overhead of the FAR protocol. A node does not forward a mobicast packet if it learns that all nodes in its care list are covered by nodes that is closer to them in terms of hops. For instance, assume a node A hears a mobicast packet m from node B and finds node C is the only one in its care list. If A finds B is closer to C (in terms of hops) than it is, it will not forward the packet. We omit the detail of this optimization in this thesis.

4.7 Summary

In this chapter we presented FAR, a new face-aware mobicast routing protocol which, in theory, reliably delivers mobicast message spatially and has good mobicast temporal characteristics. This protocol relies on the notion of spatial neighborhoods and features a novel timed face-aware forwarding method. Since there exists no close localized protocol for interesting and fair quantitative comparison, we focus on analyzing the qualitative perspectives of this protocol such as theoretical delivery accuracy, protocol cost and optimization opportunities. Besides proving that the FAR protocol achieves reliable spatial delivery, we estimated the size of its routing table in random wireless ad hoc networks via geometric analysis, and found that it is on the order of 10 entries. The latter is verified by statistical study of spatial neighborhood sizes on planar graphs. Furthermore, we also presented a novel spatial neighborhood discovery protocol and addressed key issues a spatial neighborhood discovery protocol must face, such as face identification, discovery termination, and duplicate elimination. Besides the novel merits of the FAR protocol and the spatial neighborhood discovery protocol, we believe this study, especially the proven theorems and the insight gained from statistical study about spatial neighborhood properties, helps to build a solid foundation for spatiotemporal protocol analysis in wireless ad hoc networks.

Chapter 5

The Partitionable Group Membership Problem in Mobile Ad Hoc Environments

The design of ad hoc mobile applications often requires the availability of a consistent view of the application state among the participating hosts. Such views are important because they simplify both the programming and verification tasks. We argue that preventing the occurrence of unannounced disconnection is essential to constructing and maintaining a consistent view in the ad hoc mobile environment. In this light, we provide the specification for a partitionable group membership service supporting ad hoc mobile applications and propose a protocol for implementing the service. A unique property of this partitionable group membership is that messages sent between group members are guaranteed to be delivered successfully, given appropriate system assumptions. This property is preserved over time despite movement and frequent disconnections. The protocol splits and merges groups and maintains a logical connectivity graph based on a notion of safe-distance. An implementation of the protocol in Java is available for testing. This work is used in an implementation of LIME, a middleware for mobility that supports transparent sharing of data in both wired and ad hoc wireless environments.

5.1 Problem Definition

Our ultimate goal is to provide application developers the ability to maintain a consistent global data structure in a setting in which mobile hosts come and go as they please and engage in reliable transient collaborative activities. Applications that require this level of consistency are not common today, but with the advent of wireless communication,

the situation is expected to change dramatically. Any situation that demands (for legal or technical reasons) the presence of two or more specific entities to carry out a task may impose the need for a consistent membership view, as seen from the example in the previous section. One can envision the futuristic notion of an electronic witness to a contractual transaction or the circumstance in which routine maintenance of commercial aircraft requires secure logging in the presence of an FAA inspector carrying an authorized electronic badge.

In this context, the group membership service needs to provide an accurate snapshot of the membership view all the time (without detectable inconsistency), and a message entrusted in a view shall be guaranteed to be delivered to members in that view, despite motion and motion-induced disconnections. This property makes the group membership service useful for many mobile applications, such as those we mentioned earlier. At the same time, it is a strong requirement that most other partitionable group membership services do not implement. One reason that they can not implement this requirement is that in traditional distributed system models, there is no way to know when a partition will occur. A message can be sent while a network partition is taking place, and thus may get lost. To make this strong group membership service implementable, we have to make some stronger assumptions on the system model and discuss their implementability. We first give a full specification of this strong group membership service. Second, most of the traditional distributed system models that partitionable group membership services use are asynchronous. A message may eventually be delivered be there is no guarantee on time of delivery. We now formally define the group membership problem.

5.1.1 Membership Specification

The group membership service can be specified by defining its local state variables and the safety and progress properties that it satisfies. We use terminology and notation similar to that of Cristian [46] to specify the properties. Let P be the set of all hosts that exist over time. We assume each host has a unique identifier denoted by i and drawn from the integer set Z^+, and all groups that exist over time have group identifiers drawn from a partially ordered set G. Each host in P maintains the following two state variables: g and π. g is the group identifier, and π is a subset of P. π is also called the membership view of P, or "view" for short. Let $T = [0, \infty)$ be the range of time. Two functions are introduced to simplify the phrasing of the specification:

$$group : P \times T \longrightarrow G$$

$$mem : P \times T \longrightarrow 2^P$$

$group(p,t)$ yields the group identifier for host p at local time t; $mem(p,t)$ yields p's local view of the membership π at time t.We call a group g if its identifier is g. We call g' a successor of group g if there exists a member p of g such that the next group p joins after leaving g is g'. Like in [46], $succ(g,p)$ is used to denote the successor of group g relative to host p; $pred(g,p)$ is used to to denote the predecessor of group g relative to p. Given these terms, the group membership service is specified in the following manner:

- *Self inclusion:* a host is always a part of its membership view, i.e., $p \in mem(p,t)$[1]
- *Local monotonicity:* group identifiers installed on each host are in increasing order, i.e., $pred(g,p) < g < succ(g,p)$.
- *Initial membership view:* a host always installs itself as the only member in its view when it starts, i.e., $mem(p, t_{init}) = \{p\}$.
- *Membership Agreement:* If hosts p and q have the same group id, then they have the same views, i.e., $group(p,t_p) = group(q,t_q) \Rightarrow mem(p,t_p) = mem(q,t_q)$.
- *Membership change justification:* The successor of group g with respect to p is either a proper superset or a proper subset of the group g.
- *Same view message delivery:* If host p sends a message m_{pq} to host q at time t, and q is in $mem(p,t)$, then m_{pq} is guaranteed to be delivered to q at time t', and $mem(q,t') = mem(p,t)$. Note that this property is traditionally included in group communication service ([63][13]) rather than the group membership service([44][13]). We will explain later why we include it here.
- *Conditional bounded integration:* Let p belong to group g_1 and q belongs to group g_2. Assume g_1 and g_2 are the only two groups satisfying the merging criteria (explained later) at time T. There exists a time constant T_c such that, if g_1 and g_2 stay satisfying the merging criteria for at least time T_c, the two groups will merge into one group, i.e., $\exists t_p, t_q \in [T, T+T_c], mem(q,t_q) = mem(p,t_p)$.
- *Conditional group split:* A group splits only if it is necessary, i.e., when a split condition (explained later) is met.

The first two safety requirements are common to most partitionable group membership specifications [56, 62, 63, 13, 37]. The third requirement in our specification,

[1]To simplify notation, we assume unrestricted variables are universally quantified

the initial membership view requirement, differs from most of those in the literature [56, 62, 63, 13, 37] and is relatively unique, catering to the reality of ad hoc mobile environments: a mobile host may start up with no knowledge of other hosts in the world. The membership change justification is introduced to ensure some continuity in view change properties.

The same view message delivery requirement is introduced to add more predictability to the group membership service for some applications (e.g. the LIME[140, 124, 125] middleware for mobility). With this unique property, application developers using the service are assured that message delivery is reliable within the scope of the view. In other words, within the membership view, programmers no longer need to worry about the complexity of potential inconsistencies caused by message loss due to mobility. Note that this property (or a similar one) is traditionally included in group communication service ([63],[13]) rather than the group membership service([44],[13]). The reason that we include it here is to augment the group membership service with a property needed by target applications that do not require a full blown group communication service.

The conditional eventual integration and the conditional group split are introduced to avoid the classical problem of "capricious split" [7]. Without requiring bounded integration, a group membership implementation can simply not perform any merging of groups and still satisfy the specification by keeping all the groups singletons. Without the conditional split requirement, a group membership protocol might continuously merge and split groups without justification, yet still satisfy the specification. Note that in the specification we did not explicitly say what the merging criteria and the split criteria are. The merging and split criteria in general are application dependent. In this paper, we use the weakest merging and split criteria. The merging criterion is weakest in the sense that it only requires the group membership properties to be satisfied for the new group. The split criteria is weakest in the sense that the group splits only if the group membership property cannot be guaranteed without doing so. No other condition outside of the membership specification is forced to hold.

5.1.2 System Model

Our system model assumes that there are no host crash failures and no omission or performance failures caused by network congestion. The only failure in our system model is the communication failure caused by hosts moving out of each other's communication range. This model is a reasonable starting model for ad hoc mobile systems for two reasons. First, mobility-induced disconnection is much more frequent than host crash failure given current hardware and software technology. Second, a mobile network in theory can

be equipped with enough bandwidth for communications needed by the applications on top of it, such that congestion can be made a rare event compared to the occurrence of partitions.

Our system model also assumes that the underlying communication service is reliable and timely [45], in the sense that a message entrusted to it is guaranteed to be delivered within a time bound t_d if the sender and the recipient are physically connected during that time. By physically connected we mean two hosts are either within each other's communication range or transitively connected through other hosts.

For simplicity, we assume that all hosts have the same communication radius and the communication links are bi-directional. We also assume that all hosts know their physical location all the time. We assume no knowledge of the mobility patterns of the mobile hosts except that the movement is continuous in space and has some upper bound V_{max} on speed. We purposely choose this extreme case in order to explore the limits imposed on the membership problem by random ad hoc mobility. The basic ideas behind our solution strategy are explained in the next section.

5.2 Solution Overview

Key to our strategy to implementing the strong group membership is the notion of *safe distance* among hosts and groups, i.e., the idea that if hosts are "close enough", disconnection is not possible for some time and that if they are "just far enough" there is plenty of time to carry out a configuration change protocol before disconnection actually occurs. In other words, we define a logical connectivity graph over the physical connectivity such that an edge appears in the logical graph if and only if two hosts represented by corresponding vertices are within safe distance. Group membership reflects partitions in the logical connectivity graph rather than partitions in the physical connectivity graph. In the remainder of this section we explain the safe distance concept and present the discovery and reconfiguration protocols.

5.2.1 Key Concept: Safe Distance

Given two mobile hosts equipped with compatible wireless transmitters of equal range R, we state that the distance between them is a safe distance if it does not exceed a threshold $r(v, t, t')$, defined as the maximum distance at which one can guarantee that any communication task that takes at most t time units can be completed with certainty, assuming the two hosts move randomly at a speed that does not exceed v, and the upper bound for a single atomic configuration change is t'. Clearly, safe distance cannot be defined in absolute terms but must be considered relative to a context having certain mobility and application characteristics. For example, in Figure 5.1(a) mobile hosts a and b are within communication range (R), i.e., they are able to talk to each other directly. They may want to be in the same group and start coordination or resource

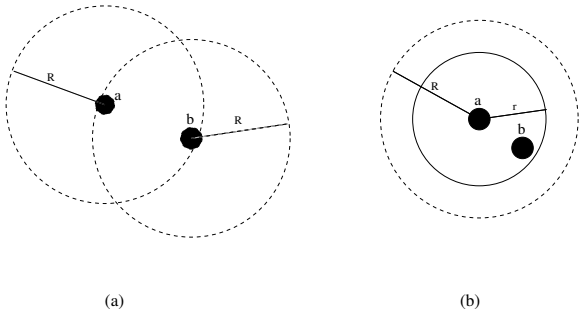


Figure 5.1: An example of safe distance

sharing immediately. Yet, they cannot do so at this point if they wish to guarantee

message delivery within the group. This is because a and b can move out of each other's range immediately after acknowledging membership in the same group, with the result being that messages between them could not be delivered successfully. The problem arises from the mobile nature of the hosts and the asynchronous nature of message passing. Our solution is to require a and b to agree on membership in the same group only when they are "close enough" (as in Figure 5.1(b)), i.e., they are at a distance

$$r \leq R - 2v * (t + t') \tag{5.1}$$

In this context, t is the upper bound for network latency (because the consistency requirement is reliable message delivery) and t' is the time needed for a group level operation (merge or split) already in progress to complete. The factor $2v$ accounts for the worst case movement pattern, i.e., a situation in which a and b are moving in opposite directions at maximum speed. One can readily see that with this restriction, the reliable message delivery between group members is guaranteed because it takes more than $t + t'$ time for the two group members to become physically disconnected, no matter how they move. Within this time any message delivery completes even if a configuration change is in progress.

We call a group safe if any two members of the group are connected via a path along which all consecutive hosts are at a safe distance. We extend the notion of safe distance from pairs of hosts to pairs of (safe) groups by requiring that at least two hosts, one in each of the two groups, are at a safe distance. While this definition seems to assume that the safe distance is independent of group size, this assumption is generally not true because both simple message delivery and reconfiguration actually depend on the number of hops that messages must traverse en route. Because the time bound on message-passing depends on the group size, our approach works only when the group size is limited by the nature of the application or is constrained by the reconfiguration protocol.

The concept of safe distance is used to determine when two groups can be merged and when a group must be split in order to maintain the requirements for group membership. To determine whether two groups are within safe distance, one must know the location of all hosts in the region. Since it is too expensive for everyone to keep track of the location of others all the time, we designate a leader for each group. All hosts in a group constantly report their location to the leader, and the leader keeps the map (Π) of the group, checking constantly to see if the group members are still within safe distance of each other and whether new hosts are present in the region. The map of a group records the spatial location of group members.

5.2.2 Group Discovery Protocol

As defined in the membership specification, each mobile host is given a unique host identifier *id*, and starts as a singleton group containing itself as the only member.

For a group to merge with another group, it must first be able to discover which other groups are present in its vicinity. The discovery protocol carries out this function and serves as a supporting layer for the group membership maintenance protocol, i.e., the reconfiguration protocol. In our discovery protocol, hosts in each group use safe distance as a criteria for finding out who is close enough to be a merge candidate, and they report any positive discoveries to their leader. For simplicity, the host with the smallest identifer in a group is chosen as the group's leader. For convenience, we also choose the host identifier (*id*) of the leader to serve as the name for its group (*gid*). Note that *gid* is not the same as the partially ordered group identifier g used in the membership specification. Rather, *gid* combined with a group change sequence number τ (discussed in more detail later in the paper) yields the partially ordered group identifier g.

Our discovery mechanism requires every host to periodically broadcast a hello message which contains its location information (xy) and its group identifier (*gid*). When two groups move close, several members of one group may receive hello messages from members of the other group. When a host u receives a hello message, it checks the sender's group identifier and location. If u finds the sender, say v, to be a member of another group located within safe distance, u passes on the information to its group leader that, in turn, will use it for merge related operations. As all group members are involved in discovery, it is possible for the group leader to receive multiple copies of the same notification regarding the appearance of one host. Duplicates are discarded.

There are several things one can do to reduce discovery costs. First, each host may attach discovery information to its periodic location update messages to the leader rather than send them separately. This pushes the discovery information towards the leader almost for free, since the location and new neighbor information represent only a few bytes that fit easily in a single packet. The cost associated with this piggy-backing method is the need for each host to keep a short-term memory (ξ) of newly discovered neighbors. Second, by utilizing neighborhood information already available at the MAC layer, a host may send neighbor greetings only when the MAC layer discovers a new neighbor. This reduces the discovery cost significantly in the case when the network topology changes infrequently. The drawback for this method is its dependence on the implementation of the MAC layer on the specific host supporting the application. We chose not to do so in our prototype.

The group discovery protocol allows the group leader to maintain a list of groups which are close enough to be considered for merging. We present the group merging protocol in the next section.

5.2.3 Reconfiguration Protocol

The reconfiguration protocol is the key layer in our group membership service. It seeks to merge groups that come into contact and to split groups that can no longer stay together. From the information collected in group discovery, a leader may find that there are one or more potential candidate groups in its vicinity suitable for merging. If so, it will initiate merging negotiations with the set (Θ) of candidates. Once an agreement is reached regarding who is to participate and who is responsible for coordinating the merge, all affected hosts receive a formal notification about the configuration change from the coordinator. Furthermore, after a host receives a group change notification, in order to prevent messages sent in one configuration from being processed in a different configuration, it must perform a barrier synchronization. One way of accomplishing this is to flush the messages in transit before doing anything in the new configuration. In addition, the participants need to delay the processing of messages arriving from "future" configurations until the synchronization is completed. Message delaying can be accomplished by tagging each message with a configuration sequence number (τ). Flushing requires the participating hosts to send extra messages whose arrivals guarantee that no more messages originating from a prior configuration are in transit. The result is an atomic configuration change. Another way of creating the synchronization boundary is by using a time-out delay. Partitioning works in the same way but without any intergroup negotiation because it involves only one group at a time. Next, we use several simple examples to illustrate the merging and partition processes.

An example of merging

Figure 5.2 depicts the merging process between two groups, G1 and G2. G1 contains hosts a and b, and b is the leader. G2 contains hosts u, v, and w, and has u as its leader. Assume u finds G1 to be in its vicinity through the discovery data sent in by v, and G1 is safe for merging. Next, u initiates the merger by sending a *merge-request* message to b, the leader of G1. If willing to participate in the merger, b sends back an acknowledgment (ACK) along with its group membership list and its configuration sequence number; otherwise, it sends back a disagreement message (NACK), which forces u to abort the merger with G1. If u gets back an ACK, as in Figure 5.2, it generates

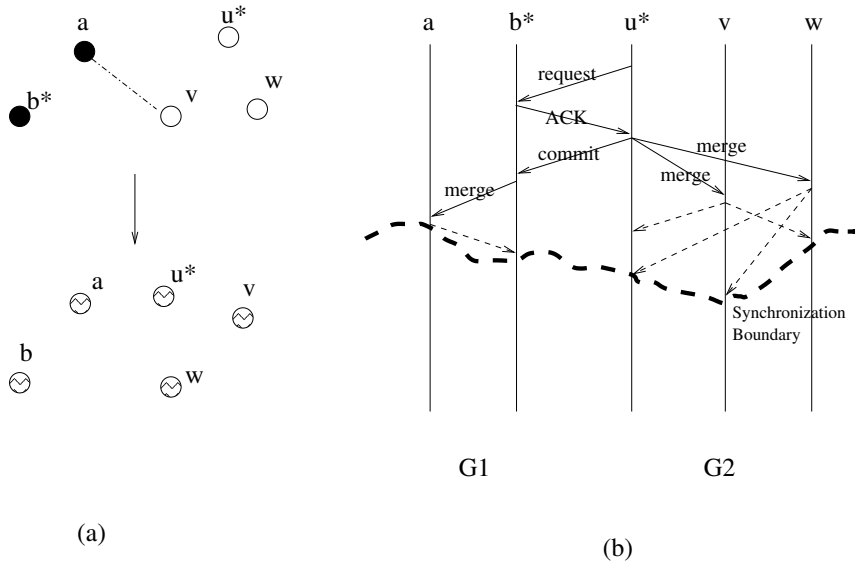


Figure 5.2: Merging Process

a new configuration number by adding one to the larger of the current configuration numbers of the two groups. Next, it sends a *merge-commit* to b and a *merge-order* to its own members. Both the *merge-commit* and *merge-order* messages contain the new group membership list, the new configuration number, and the new leader identity. Upon receiving the *merge-commit* message, b sends a *merge-order* to its own members. A host enters the flushing phase after it gets a *merge-order* message. It sends a *flush-message* to all other members of its original group and stops sending any other messages until it has received all the expected flush-messages from its group members in the old configuration.[2] After receiving all the expected flush-messages, a host enters the new configuration and all new messages it sends will have the new configuration number in their headers.Clearly, hosts may enter the new configuration at different times. It is possible for a host that is still in the old configuration to receive a message from a host that has already reached the new configuration, as shown in Figure 5.3. In such cases, the recipient must postpone the processing of this "future" message until the new configuration is established, thus pretending that the message is "received" in the new configuration. Otherwise, the consistency requirement that messages must be sent and received in the same configuration would be violated. Implementation of this requires a host to check each received message for the configuration in which it was sent before it is processed.

[2]The dashed arrows in Figures 5.2, 5.3 and 5.4 represent flush messages.

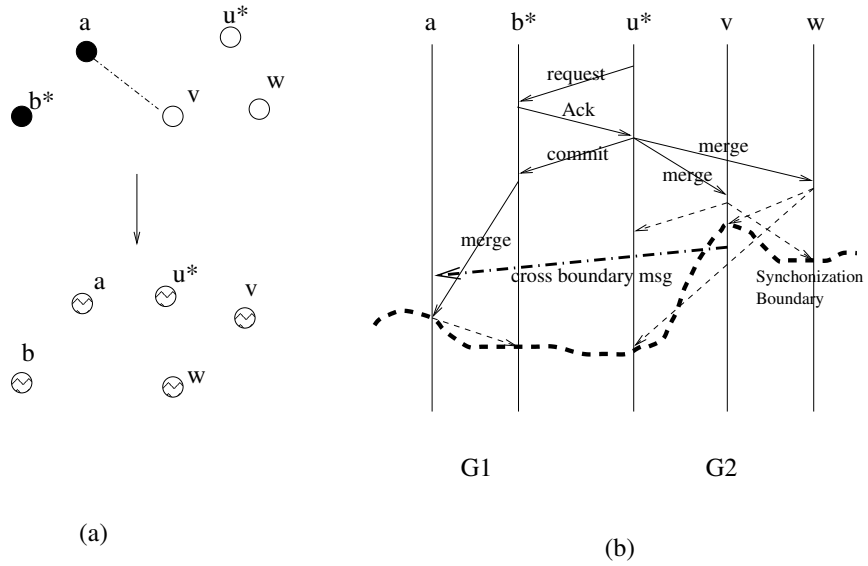


Figure 5.3: Synchronization and the Configuration Number

It is possible for u and b to initiate the merging at the same time. In this case, a tie-breaking mechanism decides who is to coordinate the merger. We use the *id* as the tie-breaker. The host with the lower *id* aborts its merger request when the collision happens. Additional complications may appear when more than two groups are involved. For example, u might have entered a merging process with other groups when it receives b's *merge-request* message, or it may no longer be a leader because of a merge with other groups or due to a partition process. In all such cases, u replies with a NACK.

An example of partitioning

Figure 5.4 illustrates the partition process. Assume that two subgroups of the group G, Z1 and Z2, are moving away from each other. By constantly checking the locations of its group members, the leader u is able to identify if its group is in a safe spatial configuration, given predefined distance-based safety criteria. Once the leader u deems the configuration to be no longer safe, it immediately issues a *split-order* message to all the group members. A *split-order* contains three pieces of information: (1) the new leader (*gid*) for the recipient, (2) the new group membership list for the recipient, and (3) the new configuration number, which is the old configuration number incremented by one. u always chooses the host with the lowest *id* in each subgroup to be the new leader for that subgroup. Upon receiving a *split-order* message, a host enters a message flushing phase, similar to the third phase in the merge process. Each host waits until it is sure that all

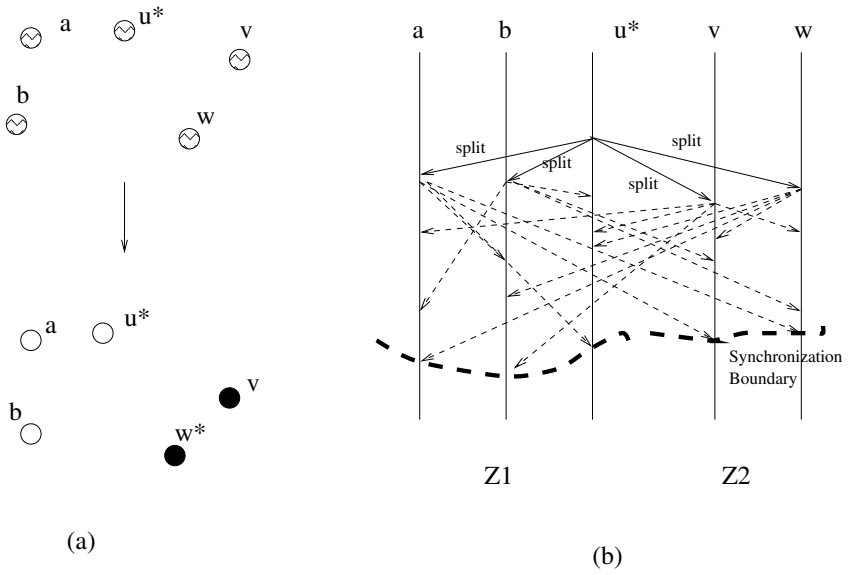


Figure 5.4: The Partition Process

messages sent to it by group members in the previous configuration are received, either by receiving all the expected flush messages or by employing a time-out delay. Each newly assigned leader assumes its leadership role after the synchronization.

The group leader must check its group configuration frequently enough in order to discover any unsafe situation in a timely fashion. As we will see in the analysis section, the threshold for safe distance does depend on the checking frequency, in addition to factors discussed earlier. The example above shows a process in which a group partitions itself into two other groups. In general, a leader might find it necessary to split its group into more than two subgroups in order to preserve the safe distance property. The partition process is the same. Next we explain how the leader determines when the group configuration is not safe and how to split it into safe sub groups.

Determine safe groups

To determine if its group configuration is safe, the leader maintains a logical connectivity graph. In the logical connectivity graph, two nodes have an edge of weight one between them if the physical distance between them is less than a partition safe distance (d_p) and no edge between them otherwise. Whenever a new location is reported, the graph is updated by recomputing all the edges to the reporting node. This takes $O(N)$ steps per update, where N is the number of nodes in the group. Given the logical connectivity graph, the depth first search (DFS) algorithm can be used by the leader to determine

State Variables

id	node identifier
gid	group identifier
xy	node location constantly updated by some external mechanism
τ	group transaction sequence number
π	group member list
ξ	the set of newly discovered leaders
Π	group map containing all members' locations (empty except for the leader)
Θ	the set of merging contacts, all of which are leaders of other groups (empty except for leaders)
$UpdateTimer$	timer for periodic location update
$GreetingTimer$	timer for periodic neighbor discovery

Support Functions

$update(\xi, gid)$;	Add gid to the list of newly discovered leaders.
$update(\Theta, \xi')$;	Update merge contact list Θ with newly discovered leaders from a member's report.
$update(\Pi, v, xy')$;	Update group map Π with group member v's new location, xy'.
$MergeSafe(\Pi, \Pi', P)$;	Verify that the merger of Π with Π' is safe, according to policy P. P includes safe distance information and the merging status of this group member. For example, if a host is in the process of merging, it is not safe to start a merge.
$ClearOldChannels()$;	Clear all group communication channels.
$GeneratePartitions(\Pi, P)$;	Generate partitions for Π, subject to policy P. This function generates a set of triples of the form $\langle \Pi_{new}, \pi_{new}, gid_{new} \rangle$.

Figure 5.5: State Variables and Support Functions

connected clusters in $O(N)$ steps. So the total time complexity for maintaining the logical connectivity graph is linear with respect to the number of nodes in the group.

Figure 5.5 summarizes the state variables a node needs to keep for the execution of the protocol, and the support functions used in the protocol presentation that follows. A brief description of each function is included.

The protocol is presented in Figure 5.6. The table lists each action taken by host u, the action's precondition, and the action's effect, given the satisfaction of the precondition. There are two types of actions in the system. The first column of the figure shows actions that are triggered by a change in the local state at host u. The second column lists actions that are triggered by the arrival of the message at host u. Each of the actions in the latter group have the form Get Message. For each of these, there is a corresponding Send Message. For example, Get NeighborHello at host u is coupled with a Send NeighborHello at another host. The figure shows only the protocol executed at a single host, u, in the system. Each host in the network has its own instance of the actions shown.

Our implementation of the group membership maintenance protocol is discussed in the next section.

Actions triggered by changes in the local state

NeighborGreetings(u)
  Precondition:
    GreetingTimer expires;
  Effect:
    reset *GreetingTimer*;
    LocalBroadcast NeighborHello(u, gid);

LocationUpdate(u)
  Precondition:
    UpdateTimer expires or ξ changes;
  Effect:
    reset *UpdateTimer*;
    Send InformLeader(u,xy,ξ) to gid;

Merge(u)
  Precondition:
    u is the leader;
    Θ is not empty;
  Effect:
    add u's members to a new map ($\Pi_{new} = \Pi$);
    add u's members to a new list ($\pi_{new} = \pi$);
    for each v in Θ
      Send merge-request(u, Π) to v
    endfor
    for each v in Θ
      if Get MergingACK(v, τ_v, Π_v)
        update group map ($\Pi_{new} = \Pi_{new} + \Pi_v$);
        update group member list
          ($\pi_{new} = \pi_{new} \cup \{\Pi_v\}$);
        store τ_v;
      else if Get MergingNACK(v)
        v will not participate in merger;
        remove v from Θ;
      endif
    endfor
    if Θ is not empty
      set τ to the max of all τ_v received;
      for each v in Θ
        Send merge-commit($\pi_{new}, gid, \tau_v, \tau_{new}$)
          to v;
      endfor
      for each w in π
        Send merge-order($\pi_{new}, gid, \tau_{new}$) to w;
      endfor
    endif
    empty Θ;
    update group member list ($\pi = \pi_{new}$);
    update group map ($\Pi = \Pi_{new}$);

Partition(u)
  Precondition:
    u is the leader;
    partition predicted based on location updates;
  Effect:
    $\Psi = GeneratePartitions(\Pi, P)$;
    for each $\langle \Pi_{new}, \pi_{new}, gid_{new} \rangle$ in Ψ
      for each w in π_{new}
        Send split-order($\Pi_{new}, \pi_{new}, gid_{new}, \tau$)
          to w;
      endfor
    endfor
    empty Ψ;

Actions triggered by the arrival of a message

Get NeighborHello(v, gid)
  Precondition:
    true;
  Effect:
    update(ξ, gid);

Get InformLeader(v, xy', ξ')
  Precondition:
    u is the leader;
  Effect:
    update(Θ, ξ');
    update(Π, v, xy');

Get merge-request(v, Π')
  Precondition:
    true;
  Effect:
    if $MergeSafe(\Pi, \Pi', P)$
      Send MergingACK(u, τ, Π) to v
      empty Θ;
      update safety condition P;
    else
      Send MergingNACK(u) to v;
    endif

Get merge-commit($\pi_{new}, gid_{new}, \tau_u, \tau_{new}$)
  Precondition:
    u is the leader;
    transaction numbers match ($\tau == \tau_u$);
  Effect:
    ClearOldChannels();
    for each w in π
      merge-order($\pi_{new}, gid_{new}, \tau_{new}$) to w;
    endfor
    update group id ($gid = gid'$);
    update transaction sequence ($\tau = \tau_{new}$);
    update group member list ($\pi = \pi_{new}$);
    empty Π;

Get merge-order($\pi_{new}, gid_{new}, \tau_{new}$)
  Precondition:
    true
  Effect:
    ClearOldChannels();
    update group id ($gid = gid_{new}$);
    update transaction sequence ($\tau = \tau_{new}$);
    update group member list ($\pi = \pi_{new}$);

Get split-order($\Pi_{new}, \pi_{new}, gid_{new}, \tau_{new}$)
  Precondition:
    true;
  Effect:
    ClearOldChannels();
    update group id ($gid = gid_{new}$);
    update transaction sequence ($\tau = \tau_{new}$);
    update group list ($\pi = \pi_{new}$);
    if $u == gid$
      update group map ($\Pi = \Pi_{new}$);
    endif

Figure 5.6: Protocol specification for host u

5.3 Group Membership Service Implementation

Our neighbor and group discovery protocol is built on top of the MAC layer, while the group membership service is built partly on an ad hoc routing layer. Fig. 5.7 shows the system architecture of our group membership service. The shaded areas indicate our contributed components.

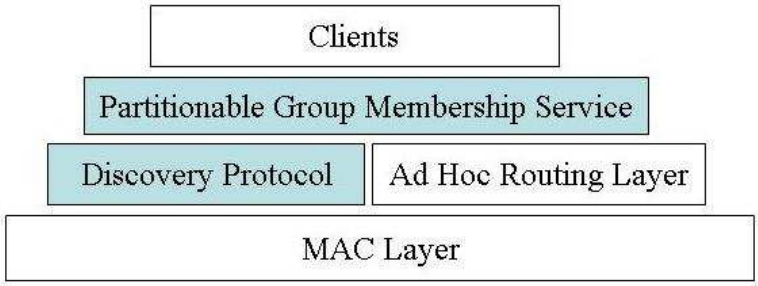


Figure 5.7: System Architecture

The implementation of the service is written entirely in Java. The discovery layer employs a simple beacon based mechanism using a thread that periodically sends a hello message. Another thread listens for incoming hello messages and passes this discovery information to the group membership layer. The main component in the group membership package is the `GroupMember` object, which contains several threads that control communication between the hosts in the network. Each different type of communication is handled by a different Java thread. These threads coordinate with each other through their owner object, the `GroupMember`. The coordination effort includes forwarding discovery information to the group leader, responding to merging and partitioning instructions, and updating the group leader with current location information. Group leaders carry the additional responsibilities of listening to their group members, communicating with other nearby group leaders, and periodically calculating the group's safety.

The group membership package presupposes ad hoc routing to be running on every host participating in the network. Therefore many of the messages discussed above are routed through other hosts in the network. As such, the leader of a group need not be directly connected to every member of the group.

The interface to the group membership protocol builds on the `EventObject` and `EventListener` classes in the Java language. An application running on a host that uses

the group membership package to participate in groups in the network simply creates a `GroupMember` object. It then registers as a listener to `GroupChangedEvent`s generated by its `GroupMember` object. When a new group configuration arises, the group membership package generates a `GroupChangedEvent` that is passed to all registered listeners. The application can take further actions, based on the implementation of this listener.

The `GroupMember` interface allows the user to specify the parameters needed for safe distance calculation. For example, the creator of the `GroupMember` can specify the host's maximum speed and its communication range. In addition to parameters for safe distance, the `GroupMember` creator also specifies the frequency of the hello beacon and the frequency of the group update messages to be sent to the leader.

While the implementation of the algorithm was a straightforward exercise in the use of Java threads and socket programming, some differences worth noting cause the implementation to vary from the examples presented in the previous sections. As presented, the protocol assumes that application level messages and the group membership protocol messages are sent on the same channel. As indicated in the discussion on merging and partitioning, ensuring that messages are received in a FIFO order and that application messages are sent and received in the same group configuration requires some additional work. The example presented in the previous section used flush messages and configuration numbers to accomplish this. The implementation, however, attempts to separate as much as possible the group discovery and maintenance from the application level and therefore leaves the flush messages and configuration numbers presented as part of the example protocol to the particular application. This separation allows each application to choose its own mechanism for ensuring atomicity. Applications with weak consistency requirements may use the group membership package without any atomicity guarantees.

```
public class GroupMember implements GroupBeaconListener {
      public GroupMember(InetAddress leaderAdd, Location loc,
                         int period, int range, int maxSpeed,
                         int updatePeriod, int networkDelay);
      public void start();
      public void stop();
      public void pause();
      public void resume();
      public synchronized void addGroupChangedListener(GroupChangedListener gcl);
      public synchronized void removeGroupChangedListener(GroupChangedListener gcl);
      public void setLocation(Location newLocation);
      public void newGroupBeacon(GroupBeaconEvent gbe);
}
```

Figure 5.8: The Public Interface of the Group Membership Package

Another concern addressed in the design was the clean separation between the group membership package and the application. By building on a model already integral to the Java language, the simple interface requires only that the application programmer understand the Java event model to successfully use the package. The simple interface composed of a single type of listener and a single type of event provides the desired ease of understanding. Figure 5.8 shows the public interface of the `GroupMember` object. The constructor accepts parameters for the safe distance calculation. With a handle to the `GroupMember` object, the programmer can start, stop, pause, and resume the `GroupMember` object. These methods affect the running of the threads that the `GroupMember` object uses for communication. The programmer can also add and remove a `GroupChangedListener`. The two final methods are not used often by the application programmer as they are used by other packages necessary for the group membership protocol to function properly. The first method allows a location generating package (e.g., a GPS monitor) to update the physical location of the host. The second method allows the `GroupMember` to respond to beacon events that are generated by a separate beaconing package. These beacons are the multicast hello messages discussed previously. Figure 5.9 shows an example usage of the group membership package.

As indicated in the previous sections, this protocol was developed because the LIME [140, 124] middleware requires the ability to transparently and consistently share data across mobile hosts in ad hoc network environments. The LIME middleware as originally released required a mobile agent or host to explicitly announce its intention to engage or disengage from a group. The integration of this group membership protocol with the LIME middleware transforms the processes of engagement and disengagement into transparent reconciliations of LIME information when agents or hosts move in the network, thereby changing their status with respect to the protocol's safety requirements. In such a way, we were able to implement a mobility-aware version of LIME which automatically engages and disengages depending on the relative distances and mobility patterns of the participating hosts.

Because the group membership package is completely independent of LIME or any other application that may use it, changes to the package do not affect LIME, as long as the changes to the package do not affect its interface. This allows for future 'pluggable' versions of the group membership package to replace the current version. One can envision an implementation in which the safe distance is based on something more complex than physical location, such as signal strength, stability etc.

```
// The test class monitors the changes to a particular group member's group
// An instance of this class runs on each participating host
public class Test implements GroupChangedListener {
      // keep a handle to the group member object
      private GroupMember g;
      // integer count of the number of changes that have occurred
      private int changes = 0;
      public Test(GroupMember g) {
            this.g = g;
            // make this object a listener for events generated by the package
            g.addGroupChangedListener(this);
      }
      // this method is required by the GroupChangedListener interface
      // it is called when a new GroupChangedEvent occurs
      public void groupChanged(GroupChangedEvent gce) {
            // log the receipt of the change
            changes++;
            System.out.println(``Change:''  S + changes);
      }
      public static void main(String[] args) {
            // create a new GroupMember object for this host
            GroupMember g = new GroupMember(InetAddress.getLocalHost(),
                                            new Location(0,0), 1000, 3, 0, 100, 0);
            // create an instance of the Test class to monitor the GroupMember
            Test t = new Test(g);
            // start the GroupMember
            g.start();
      }
}
```

Figure 5.9: An Example Use of the Group Membembersip Package

5.4 Safety Analysis

The key feature of our algorithm is the use of location information and safe distance in the group membership management. The leader of a group frequently checks the members' locations to make sure that only those that are guaranteed to stay connected with the group for at least $t + t'$ more units of time remain in the group, where t is the time specified by the application layer and t' is the time bound for configuration changes. The combination of t and t' determines the safe distance for a specific operation, which could be the merging operation, the partitioning operation, or any other group operations specified by the application. Let's assume that t_d is the maximum delay between the time a control message is issued and the time it is received and processed, i.e., the sum of the maximum network delay and the maximum process queuing delays both at the sender and the receiver. For convenience, we refer to t_d as the network delay. In the case of

splitting, the maximum time it takes for a group to be partitioned successfully is twice the network delay.

If the leader continuously monitors the group configuration and all member locations are up to date, then mobility-induced unannounced disconnection can be caught in advance and dealt with successfully by requiring $t' > 2 * t_d$. Yet the leader's information about members' locations is always a little bit out of date. If members sample and report their locations every t_u units of time, the location information the leader has about a member could be outdated by time $t_u + t_d$. Taking this into consideration, the reserved time T must be greater than $t_u + t_d + 2 * t_d = t_u + 3t_d$. Whether or not we can use $d_r = R - 2V_{max}(t_u + 3t_d)$ as the safe distance for partitioning depends on the requirement for merging. Because we do not allow a merging process to be aborted once committed, the computation of safe distance for partitioning also needs to account for the time associated with the merging process. Consider the following scenario: immediately before a commit in a merging process, the group configuration is safe using safe distance d_r; right after the commit, a leader might discover that its group is no longer safe, and a partition process needs to be carried out immediately. However, the merging process hasn't finished. This is not acceptable. Taking into account that the two-phase merging process needs at most an execution time of $4t_d$ (4 messages), and the configuration needs to be safe right after merging, the total reserved time for both merging and partitioning needs to be $t_u + 3t_d + 4t_d = t_u + 7t_d$. In other words, the safe distance for both merging and partitioning is

$$d_s = R - 2V_{max}(t_u + 7t_d) \tag{5.2}$$

Using the same distance for merging and partitioning introduces the problem of 'shuttle nodes', i.e., if a node is moving in and out the safe boundary, mergers and partitions occur repeatedly. To avoid this, one can further tighten the safe distance for merging, creating a 'buffer zone', and thus reducing the probability of shuttling.

Our algorithm also requires V_{max} to be no greater than V_{adm}, i.e., the maximum admissible speed for the specific wireless network system the mobile hosts are using. Most wireless network systems (e.g. DECT, GSM, PCS, ETACS) have a maximum admissible speed [141]. When a mobile node is moving too fast, it simply becomes invisible to the network. For GSM and PCS, V_{adm} is about $50m/s$; for DECT microcellular system, V_{adm} is about $11m/s$. Without the condition $V_{max} \leq V_{adm}$, a speed change from $V \leq V_{adm}$ to $V > V_{adm}$ creates an unannounced disconnection. Speed monitoring would be needed to prevent this kind of unannounced disconnection from happening.

Figure 5.10 illustrates the relation between the safe distance r and the maximum admissible network delay t_d with reasonable values of $R = 150m$, $V_{max} = 10m/s$ and location reporting frequency of 1 Hz ($t_u = 1s$). One can see that as the delay bound increases, the safe distance decreases.

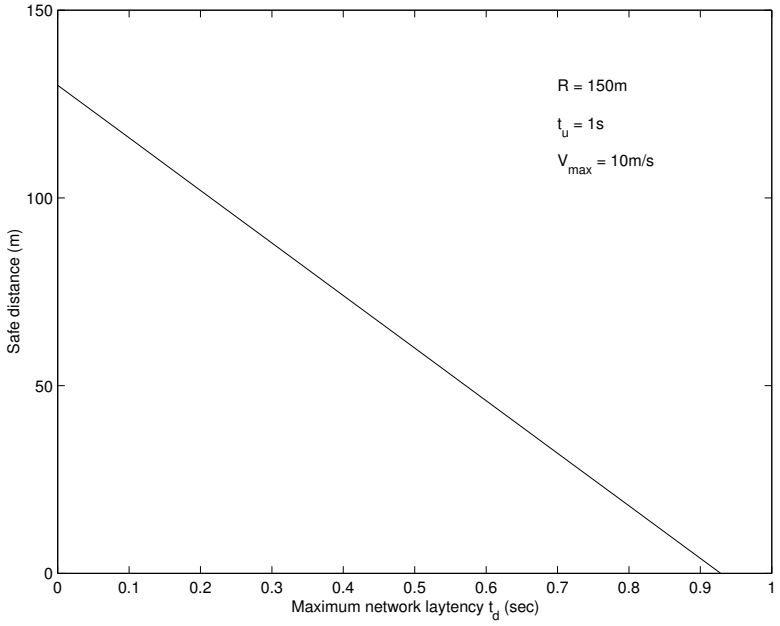


Figure 5.10: Safe distance vs. network delay

Figure 5.11 shows the relation between the safe distance threshold, the upper bound on speed, and the network delay bound. The region above the top curve corresponds to $d_s < 0$. In this parameter space we cannot provide any consistency guarantees for a group containing more than one member. On the other hand, if a mobile system's network delay bound and maximum speed bound fall into the region below the ($d_s = 90m$) curve, we could provide the group view consistency guarantee by using $90m$ as maximum safe distance.

Choosing the proper safe distance is key to the protocol. A choice that is not conservative enough might endanger the correctness of the group membership service. A choice that is too conservative might cause the groups to be too small to be useful for some applications, or smaller than necessary. One must strike a delicate balance between these conflicting choices to make the group membership service as useful as possible.

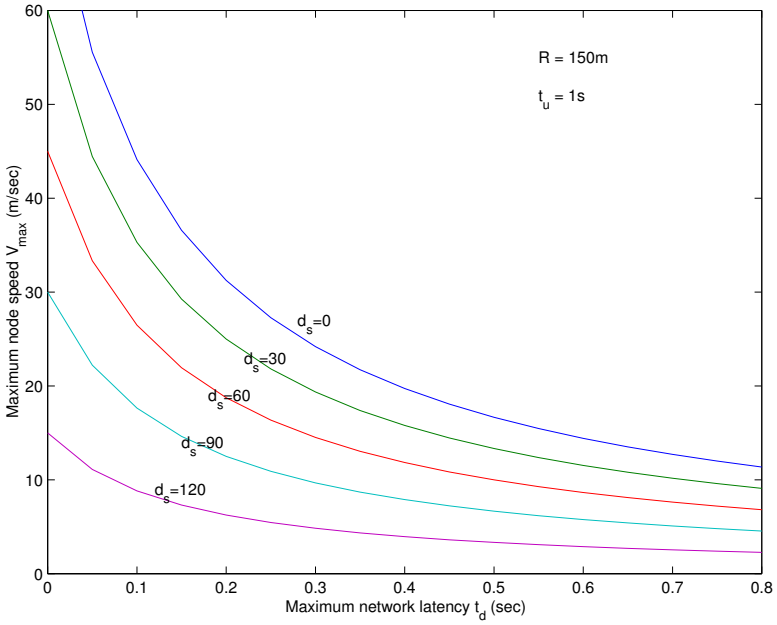


Figure 5.11: Relation between safe distance, speed bound and delay bound

5.5 Discussion

Our partitionable group membership specification is stronger than traditional ones in that it not only requires availability during partition but also emphasizes consistency during partition. Previous work in group membership either admits inconsistency during the partition, or reduces availability during partition.

On the other hand, the strong properties required by our partitionable group membership specification make it impossible to implement in traditional asynchronous system models. Key to our solution to the strong partitionable group membership problem is the notion of safe distance and the corresponding notion of a logical connectivity graph. Given information about physical properties of the mobile system, we are able to predict certain behavior of the system. In turn, we are able to achieve the strong consistency required by the group membership service.

At present, our algorithm assumes that all mobile nodes in the system have a known maximum speed. Unbounded speed is another possible source of unannounced disconnection. Low speed is a requirement for most wireless networks. In systems involving mobile nodes that can control their own velocity, e.g., cars and planes, a safe relative velocity threshold can be used in the decision of merging and splitting. Of course, in such

cases we would have to assume a maximum acceleration for the mobile nodes in order to make disconnection predictions possible.

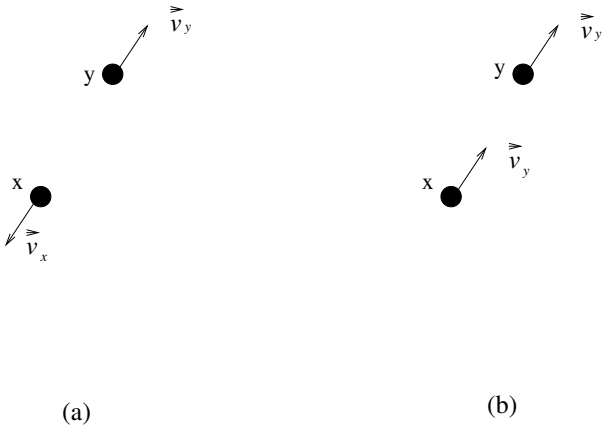


Figure 5.12: Contribution of velocity information

Our membership service can be further refined when velocity information about each mobile host is available. For example, consider cases (a) and (b) of Figure 5.12. In case (a), hosts x and y are moving away from each other, while in (b), they are moving in the same direction. Clearly, x and y are less likely to disconnect in the latter case than in the former. Translating this into the language of safe distance, the maximum safe distance between x and y is greater in case (b) than in case (a). In the current algorithm, we assume the velocity information is not available. Since we cannot differentiate cases (a) and (b), we have to consider the worst case movement scenario for each pair of hosts, i.e., they may be moving away from each other at the maximum relative speed at any point in time. When velocity information is available, the safe distance threshold between hosts x and y (in Figure 5.12) can change dynamically according to the formula:

$$R - |\vec{v}_x - \vec{v}_y| \cdot t - |\vec{a}_{max} \cdot t^2| \tag{5.3}$$

where $\vec{a}_{max}$ is the maximum acceleration for all hosts and t is the time needed for a group operation already in progress to finish. We can change our algorithm to use the velocity information in the following way: (1) each host includes its velocity information in hello messages and location-update messages, and (2) the safe distance is computed using Equation (5.3) with $t = t_u + 7t_d$. The rest of the algorithm remains the same.

Although only safe physical distance is employed in our protocol to avoid unannounced disconnection, other physical attributes can also be used to determine safety.

For instance, if link failure is predictable through monitoring the bandwidth or transmission power change between two nodes, a similar group membership protocol can be built by exploiting similar concepts, e.g., 'safe bandwidth' or 'safe power level' etc.

We assume each mobile host has knowledge of its own location. This is made possible by the availability of positioning systems such as GPS. However, positioning systems are not always accurate. For simplicity we did not consider this in our safe distance analysis. One can factor data precision and sample frequency of a location system into the safe distance and make the service more robust.

We assume that the model of wireless communication has a known communication radius. This choice, often quoted as the "unit disk model" ([69],[106]), is widely used in theoretical studies about the properties of wireless ad hoc networks. While this simplistic radio communication model is an appropriate starting point for reasoning about certain wireless network properties in an open environment, such as highway (automobiles) and desert (battle groups), it is not valid for indoor scenarios. The reason is that radio communication has very unpredictable characteristics in the indoor environment (due to reflection, absorbtion, interference, etc.), and an omnidirectional range is not a good abstraction. In turn, our safe distance based group membership service is applicable only to open field environments in which the omnidirectional wireless communication range is an appropriate approximation.

The correctness of our algorithm relies on the assumption that the network has a delay bound. At this moment, we are not aware of any ad hoc routing protocols which can provide a good delay bound. Yet, it is conceivable that a routing protocol with good delay bound for prioritized group control messages is possible by restricting group size and using location information. An alternative approach is to augment the merging criteria with a maximum group size or even some group spatial configuration condition. By doing this, it may be possible to meet the delay bound assumption with high probability.

5.6 Conclusion

The motivation for this work rests with our desire to provide data consistency in applications that execute over ad hoc networks. Yet, maintaining a consistent view of the global state in a distributed network is difficult in general and essentially impossible in the presence of unannounced disconnections. In ad hoc mobile systems, mobility-induced unannounced disconnection occurs frequently, as part of the normal operation of the network. This makes the development of fault-tolerant systems on top of ad hoc networks very challenging. Our goal of assisting software developers in their efforts to design and

build reliable mobile applications leads us to define a new partitionable group membership service with strong consistency requirements. We have also presented a strategy and an algorithm to implement the service, given appropriate system assumptions. The novel feature of the algorithm is its ability to create the illusion of announced disconnection. By using location and mobility information about the mobile hosts in the region, the membership service is able to guarantee to the application layer a reliable message delivery service to group members in the presence of mobility-induced unannounced disconnection, given appropriate system assumptions. This approach represents a new direction in fault-tolerant distributed computing, one that factors into protocols information about mobility and space. This work also provides a practical solution to masking mobility induced unannounced disconnections in ad hoc mobile systems.

Chapter 6

Summary and Future Work

6.1 Summary

This work proposes spatiotemporal multicast, a new multicast paradigm, for disseminating space-and-time sensitive data for applications in wireless ad hoc networks, and presents an extensive discussion of protocol design issues. This dissertation also explores the partitionable group membership problem and investigates strategies for conserving consistency in the case of mobility.

6.1.1 Spatiotemporal Multicast

Our investigation of spatiotemporal multicast starts with mobicast, a special case of spatiotemporal multicast featuring a mobile delivery zone. The inherently powerful just-in-time spatial delivery semantics of mobicast can optimize resource utilization for multicast tasks in sensor networks and enables application programmers to explicitly address both spatial and temporal perspectives of communication and coordination, in a manner atypical of current multicast models.

To demonstrate the feasibility of mobicast, we first developed a stateless forwarding zone based protocol (FZC) and explored its ability to meet strong spatiotemporal guarantees. We introduced two new notions of network compactness, namely Δ-compactness and Γ-compactness, for capturing key network topological information for spatiotemporal communication. We also introduced "k-cover," a new geometric concept that generalizes ellipses, and proved several related theorems useful in the analysis of information propagation in wireless ad hoc networks. Using these results, we are able to determine the

shape of the forwarding zone and the theoretical headway distance needed for our protocol to ensure strong multicast delivery guarantees in space and time while minimizing retransmission overhead and average slack time.

We also investigated the network compactness properties of randomly distributed wireless ad hoc networks and their implication on performance of mobicast protocols. We found that the distribution of values for pairwise (path) compactness in randomly distributed wireless networks to be highly concentrated around a peak close to one with a very small portion close to zero. This leads to the identification of a fundamental tradeoff between probabilistic delivery guarantees and communication overhead in spatiotemporal multicast. Via analysis and simulation, we found that forwarding zone based mobicast protocols can indeed significantly reduce its communication overhead via a propitious choice of forwarding zone size by only a slight relaxation of its delivery guarantee.

Furthermore, we proposed and studied a scalable face-aware mobicast routing (FAR) protocol which, in theory, also reliably delivers mobicast message. This protocol relies on a notion of spatial neighborhoods and features a novel timed face-aware forwarding method. Besides proving that the FAR protocol achieves reliable spatial delivery, we estimated the size of its routing table in random wireless ad hoc networks via geometric analysis, and found that it is on the order of 10 entries. The latter is verified by a statistical study of spatial neighborhood sizes on planar graphs. To complete the loop, we also proposed a novel spatial neighborhood discovery protocol and addressed key issues such a protocol must face, such as face identification, discovery termination, and duplicate elimination.

We believe this study helps to build a solid foundation for spatiotemporal protocol analysis in wireless ad hoc networks, and hope this work will facilitate a broad research effort in spatiotemporal communication mechanisms and wireless ad hoc network applications.

6.1.2 Partitionable Group Membership

Our goal of assisting software developers in their efforts to design and build reliable mobile applications leads us to define a new partitionable group membership service with strong consistency requirements. Our contribution is two-fold. The first contribution is a new specification of partitionable group membership service catering to the requirements of mobile applications. The second is a proactive partitionable group membership protocol that correctly implements the new specification. A novelty of this protocol is the use of mobility information to avoid the unannounced disconnection, created by mobility.

6.2 Future Work

Spatiotemporal multicast represents a new horizon in communication research and there is much future work waiting to be pursued. We discuss a few key topics here.

Mobicast in mobile environments

The mobicast protocols proposed in this work along with their analysis and simulations have been focused on the context of relatively static networks such as sensor networks. However, our ambulance warning system example (in Chapter 1) clearly points to mobicast's application potential in dynamic mobile ad hoc network environments. The dynamic topology of mobile ad hoc networks demands relaxing of assumptions, and invites new specifications of delivery guarantees and corresponding protocols, and associated performance study and analysis.

Other spatiotemporal multicast scenarios

Mobicast is just one, albeit the first, interesting spatiotemporal multicast scenario that caught our attention. There are many other spatiotemporal multicast possibilities. For instance, one possibility is a "sticky" spatiotemporal multicast that lets a message stay in a geographic region for a certain amount of time. This kind of service mimics an advertisement with an expiration and can be used in ad hoc mobile games. Interesting spatiotemporal multicast extensions include those that dynamically change their spatiotemporal constraints according to local context.

Spatiotemporal services

Higher level spatiotemporal services can be built on top of spatiotemporal multicast. For instance, the information scouting example in Chapter 2 invites a spatiotemporal service. Mobicast can be used to distribute the spatiotemporal query. However, we have not explored strategies for transporting the distributed answers from the network to the mobile scout. Techniques such as directed diffusion[90] can be explored, along with many other strategies.

The application of compactness metrics and the spatial neighborhood concept to larger context

The mathematical byproducts of this investigation have application domains outside of spatiotemporal multicast. We expect them to be helpful in devising more efficient geometric unicast protocols. As general graph concepts, they might also have potential applications in areas such as biological and social science. Many possibilities in these directions have yet to be explored.

Performance under stress

This dissertation has primarily investigated mobicast protocols from a theoretical perspective. The network simulations presented do not have results from stress tests, partly due to the current limitation of the ns-2 network simulator. How background traffic will affect the delivery ratio and timeliness of the spatiotemporal protocols and how the protocols should be adjusted accordingly are questions we hope to answer in the near future.

Identification and classification of wireless ad hoc networks

Different application scenarios and requirements result in different wireless ad hoc network deployment strategies and characteristics. These differences result in wireless ad hoc networks that favor different protocols. Simulation studies of current wireless ad hoc routing protocols are largely done using random network topologies. While performance on random networks provides a baseline, protocols designed with more specific knowledge of target networks such as their topology and expected traffic pattern are often more efficient. In turn, the identification and classification of wireless ad hoc networks is an important step in communication protocol research, and the results points to more optimized spatiotemporal multicast protocols. This topic so far has been largely ignored in the wireless ad hoc networking community and there is much work to be done.

6.3 Final Remarks

This dissertation takes the first steps in studying the spatiotemporal multicast paradigm and protocols in wireless ad hoc networks, and we believe this work provides a glimpse of a compelling new information dissemination paradigm, and helps to lay a solid foundation for future research. We hope this work will lead to more interesting and fruitful explorations in spatiotemporal multicast, and will benefit human society at large.

References

[1] T. Abdelazaher, B. Blum, D. Evans, J. George, S. George, L. Gu, T. He, C. Huang, P. Nagaraddi, S. Son, P. Sorokin, J. Stankovic, and A. Wood. Envirotrack: An environmental programming model for tracking applications in distributed sen-sor networks. Submitted to ICDCS 2003.

[2] S. Akazawa. The scope of the japanese information industry in the 1980s. In K. R. Brown, editor, *The challenge of information technology: Proceedings of the forty-first FID (Federation Internationale de Documentation) congress held in Hong Kong 13-16 September 1982*, pages 19–22, Amsterdam, New York, and Oxford, 1983. North Holland.

[3] Rajeev Alur and Thomas A. Henzinger. A really temporal logic. In *IEEE Symposium on Foundations of Computer Science*, pages 164–169, 1989.

[4] American Library Association, Young Adult Services Division, Services Statement Development Committee. Directions for library service to young adults. Chicago, 1978.

[5] Y. Amir, D. Dolev, S. Kramer, and D. Malki. Transis: A communication subsystem for high availability. In *FTCS-22: 22nd International Symposium on Fault Tolerant Computing*, pages 76–84, Boston, Massachusetts, 1992. IEEE Computer Society Press.

[6] Y. Amir, L. E. Moser, P. M. Melliar-Smith, D. A. Agarwal, and P. Ciarfella. The Totem single-ring ordering and membership protocol. *ACM Transactions on Computer Systems*, 13(4):311–342, 1995.

[7] E. Anceaume, B. Charron-Bost, P. Minet, and S. Toueg. On the formal specification of group membership services. Technical Report 95-1534, INRIA, France, 1995.

[8] Gilberto Artioli. *Structural studies of the water molecules and hydrogen bonding in zeolites*. PhD thesis, University of Chicago, 1985.

[9] Sunil Arya and Michiel Smid. Efficient construction of a bounded degree spanner with low weight. Technical Report MPI-I-94-115, Saarbrücken, 1994.

[10] O. Babaoglu, R. Davoli, and A. Montresor. Group communication in partitionable systems: Specification and algorithms. *IEEE Transactions on Software Engineering*, 27(4):308–336, April 2001.

[11] O. Babaoglu, R. Davoli, and A. Montresor. Group communication in partitionable systems: Specification and algorithms. *IEEE Transactions on Software Engineering*, 27(4):308–336, April 2001.

[12] Ozalp Babaoglu, Alberto Bartoli, and Gianluca Dini. Programming partition-aware network applications. In *Advances in Distributed Systems*, pages 182–212, 1999.

[13] O. Babaoglun, R. Davoli, and A. Montresor. Group communication in partitionable systems: Specification and algorithms. *IEEE Transactions on Software Engineering*, 27(4):308–336, April 2001.

[14] Prithwish Basu, Naved Khan, and Thomas D.C. Little. A mobility based metric for clustering in mobile ad hoc networks. In *Proc. of IEEE ICDCS 2001 Workshop on Wireless Networks and Mobile Computing*, Phoenix, AZ, USA, April 2001.

[15] K. P. Birman. The process group approach to reliable distributed computing. *Communications of the ACM*, 36(12):37–53, December 1993.

[16] Bluetooth. SIG. http://www.bluetooth.com.

[17] Brian Blum, Prashant Nagaraddi, Anthony Wood, Tarek Abdelzaher, Sang Son, and John Stankovic. An entity maintenance and connection service for sensor networks. The First International Conference on Mobile Systems, Applications, and Services (MobiSys), San Francisco, CA, May 2003.

[18] J. Boleng, T. Camp, and V. Tolety. Mesh-based geocast routing protocols in an ad hoc network. In *Proc. of the IEEE Intl. Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing (IPDPS)*, pages 184–193, April 2001.

[19] P. Bose, L. Devroye, W. Evans, and D. Kirkpatrick. the spanning ratio of gabriel graphs and -skeletons, 2001.

[20] Prosenjit Bose and Pat Morin. An improved algorithm for subdivision traversal without extra storage. In *International Symposium on Algorithms and Computation*, pages 444–455, 2000.

[21] Prosenjit Bose, Pat Morin, Ivan Stojmenovic, and Jorge Urrutia. Routing with gurranteed delivery in ad hoc wireless networks. *Wireless Networks*, 7:609–616, 2001.

[22] Linda Briesemeister. Sensor data dissemination through ad hoc battlefield communications. In *Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS)*, January 2003.

[23] Linda Briesemeister and Günter Hommel. Integrating simple yet robust protocol layers for wireless ad hoc intervehicle communications. In *Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS)*, pages 186–192, January 2002.

[24] Linda Briesemeister and Günter Hommel. Localized group membership service for ad hoc networks. In *Proceedings of International Workshop on Ad Hoc Networking (IWAHN)*, pages 94–100, August 2002.

[25] J. Broch, D.B. Johnson, and D.A. Maltz. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks. Internet Draft, October 1999. IETF Mobile Ad Hoc Networking Working Group.

[26] J. Broch, D.A. Maltz, D.B. Johnson, Y.-C. Hu, and J. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proc. of the 4^{th} Annual ACM/IEEE Int. Conf. on Mobile Computing and Networking*, Dallas, TX, USA, October 1998. ACM.

[27] R. Brooks and J. McLurkin. Using Cooperative Robots for Explosive Ordnance Disposal. http://www.ai.mit.edu/projects/microrobots/. MIT Artificial Intelligence Laboratory.

[28] R. R. Brooks, C. Griffin, and D. S. Friedlander. Self-organized distributed snesor network entity tracking. *International Journal of High Performance Computing Applications*, 16(3), 2002.

[29] J. Cartigny, D. Simplot, and J. Carle. Stochastic flooding broadcast protocols in mobile wireless networks. Technical Report 2002-03, IRCICA/LIFL, Univ. Lille 1, 2002.

[30] J. Cartigny, D. Simplot, and I. Stojmenovic. Localized minimum-energy broadcasting in ad-hoc networks. Technical Report 8, IRCICA/LIFL, Univ. Lille 1, Lille, France, 2002.

[31] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao. Habitat monitoring: Application driver for wireless communications technology. In *ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean, Costa Rica, April 2001*, 2001.

[32] Barun Chandra, Gautam Das, Giri Narasimhan, and Jose Soares. New sparseness results on graph spanners. *International Journal of Computational Geometry and Applications*, 5:125–144, 1995.

[33] T. D. Chandra, V. Hadzilacos, S. Toueg, and B. Charron-Bost. On the impossibility of group membership. In *Proceedings of the 15th Annual ACM Symposium on Principles of Distributed Computing (PODC'96)*, pages 322–330, New York, USA, 1996. ACM.

[34] T. D. Chandra and S. Toueg. Unreliable Failure Detectors for Asynchronous Systems. *Journal of ACM*, 43(2):225–267, 1996.

[35] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking, (MobiCom 2001), Rome, Italy, July 16-21*, 2001.

[36] C. Chiang, M. Gerla, and L. Zhang. Adaptive shared tree multicast in mobile wireless networks. Proceedings of GLOBECOM '98, pp.1817-1822, November 1998.

[37] G. Chockler, I. Keidar, and R. Vitenberg. Group communication specifications: A comprehensive study. *ACM Computing Survey*, 33(4):427–469, December 2001.

[38] M. Chu, H. Haussecker, and F. Zhao. Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks. Tech Report P2001-10113, Xerox Palo Alto Research Center, 2001.

[39] Cicerone, Di Stefano, and Handke. Survivable networks with bounded delay: The edge failure case. In *ISAAC: 10th International Symposium on Algorithms and*

Computation (formerly SIGAL International Symposium on Algorithms), Organized by Special Interest Group on Algorithms (SIGAL) of the Information Processing Society of Japan (IPSJ) and the Technical Group on Theoretical Foundation of Computing of the Institute of Electronics, Information and Communication Engineers (IEICE)), 1999.

[40] B. G. F. Cohen. Human aspects in office automation. Technical Report NTIS, PB84-240738, National Institute for Occupational Safety and Health, Division of Biomedical and Behavioral Science, Cincinnati, 1984.

[41] D. Coore, R. Nagpal, and R. Weiss. Paradigms for Structure in an Amorphous Computer. A.I. Memo No. 1614, MIT AI Lab, October 1997.

[42] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. The MIT Press, 1999.

[43] European Transport Safety Council. *Intelligent Transportation Systems and Road Safety*. Brussels, 1999.

[44] F. Cristian. Reaching agreement on processor-group membership in synchronous distributed systems. *Distributed Computing*, 4(4):175–188, 1991.

[45] F. Cristian. Group, majority, and strict agreement in timed asynchronous distributed systems. In *Proceedings of the 26th International Symposium on Fault-Tolerant Computing*, pages 178–187, 1996.

[46] F. Cristian. Synchronous and asynchronous group communication. *Communications of the ACM*, 39(4):88–97, 1996.

[47] F. Cristian. Synchronous and Asynchronous Group Communication. *Communications of the ACM*, 39(4):88–97, April 1996.

[48] F. Cristian, H. Aghali, R. Strong, and D. Dolev. Atomic broadcast: From simple message diffusion to byzantine agreement. In *Proceedings of the 15th Int. Symp. on Fault-Tolerant Computing (FTCS-15)*, pages 200–206, Ann Arbor, MI, USA, 1985. IEEE Computer Society Press.

[49] F. Cristian and C. Fetzer. The timed asynchronous system model. Technical Report CS-97519, University of california - San Diego, January 1997.

[50] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Alpgorithms and Applications*. Springer, second edition edition, 1998.

[51] Mark de Berg, Marc van Kerveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry*. Springer, 1998.

[52] S. Deering. *Multicast Routing in Datagram Inter-network*. PhD thesis, Stanford University, 1991.

[53] Zhidong Deng and Weixiong Zhang. Localization and dynamic tracking using wireless-networked sensors and multi-agent technology: First steps. *IEIEC Transactions on Fundamental Electronics, Communication and Computer Science*, 85-A(11), 2002.

[54] D. Dolev, D. Malki, and R. Strong. A Framework for Partitionable Membership Service. Technical Report CS95-4. The Hebrew University of Jerusalem.

[55] D. Dolev, D. Malki, and R. Strong. An asynchronous membership protocol that tolerates partitions. Technical Report 94-6, Institute of Computer Science, The Hebrew University of Jerusalem, 1994.

[56] D. Dolev, D. Malki, and R. Strong. A framework for partitionable membership service. In *Proceedings of the 15th ACM Symposium on Principles of Distributed Computing*, May 1996.

[57] Steph Durocher and David Kirkpatrick. On the hardness of turn-angle-restricted rectilinear cycle cover problems. In *Preceedings of the Canadian Conference on Computational Geometry*, 2002.

[58] C. Dwork, N.A. Lynch, and L. Stockmeyer. Consensus in the presense of partial synchrony. *Journal of ACM*, 35(2):288–323, April 1988.

[59] David Eppstein. Spanning trees and spanners. Technical Report ICS-TR-96-16, 1996.

[60] David Eppstein. Spanning trees and spanners. In *In J.-R. Sack and J. Urrutia, editors, Handbook of Computational Geometry*, pages 425–461, Amsterdam, 1999. Elsevier Science.

[61] D. Estrin et al. Embedded everywhere: A research agenda for networked systems of embedded computers. National Academy Press, 2001. Computer Science and Telecommunications Board (CSTB) Report.

[62] P. D. Ezhilchelvan, R. A. Macedo, and S. K. Shrivastava. Newtop: A fault-tolerant group communication protocol. In *Proceedings of the International Conference on Distributed Computing Systems*, pages 296–306, 1995.

[63] Alan Fekete, Nancy Lynch, and Alex Shvartsman. Specifying and using a partitionable group communication service. *ACM Transactions on Computer Systems*, 19(2):171–216, 2001.

[64] M. Fischer. A theoretician's view of fault tolerant distributed computing. In B. Simons and A. Spector, editors, *Fault-Tolerant Distributed Computing*, Lecture Notes on Computer Science, pages 1–9. Springer-Verlag, Berlin, Germany, 1990.

[65] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of Distributed Consensus with One Faulty Process. *Journal of ACM*, 32(2):374–382, 1985.

[66] Sally Floyd, Van Jacobson, Ching-Gung Liu, Steven McCanne, and Lixia Zhang. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Transactions on Networking*, 5(6):784–803, 1997.

[67] R. Friedman and R. van Renesse. Strong and Weak Virtual Synchrony in Horus. Tr95-1537, Cornell University, Department of Computer Science, August 1995.

[68] A. Galleni and D. Powell. Towards a unified comparison of synchronous and asynchronous agreement protocols, 1995.

[69] J. Gao, L. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Geometric Spanner for Routing in Mobile Networks. In *Proceeddings of the ACM Internationl Symposium on Mobile ad hoc networking and computing (MobiHoc)*, pages 45–55, Long Beach, CA, USA, 2001.

[70] Jie Gao, Leonidas J. Guibas, John Hershberger, Li Zhang, and An Zhu. Geometric spanner for routing in mobile networks.

[71] David Gelernter. Generative Communication in Linda. *ACM Transactions on Programming Languages and Systems*, 7(1):80–112, 1985.

[72] J. Gray. Notes on Database Operating Systems. In *Lecture Notes in Computer Science*, volume 60, pages 393–481. Spring-Verlag, 1978.

[73] Martin Greenberger, Julius Aronofsky, James L. McKenney, and William F. Massy, editors. *Networks for research and education: Sharing of computer and information resources nationwide*. MIT Press, Cambridge, 1974.

[74] R. Guerraoui and A. Schiper. Consensus: the big misunderstanding. In *Proceedings of the 6th IEEE Computer Society Workshop on Future Trends in Distributed Computing Systems (FTDCS-6)*, pages 183–188, Tunis, Tunisia, 1997. IEEE Computer Society Press.

[75] Sudipto Guha and Samir Khuller. Approximation algorithms for connected dominating sets. In *European Symposium on Algorithms*, pages 179–193, 1996.

[76] Radu Handorean, Jamie Payton, Christine Julien, and Gruia-Catalin Roman. Coordination middleware supporting rapid deployment of ad hoc mobile systems. In *Proceedings of the 23rd IEEE International Conference on Distribiuted Computing Systems Workshops*, pages 362–268, Providence, May 2003. IEEE Computer Society.

[77] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *HICSS*, 2000.

[78] Shannon Hetrick. Examination of Driver Lane Change Behavior and The Potential Effectiveness of Warning Onset Rules for Lane Change Or "Side" Crash Avoidance Systems. Master's thesis, Virginia Polytechnic Institute and State University, March 1997.

[79] Christopher Ho, Katia Obraczka, Gene Tsudik, and Kumar Viswanath. Flooding for reliable multicast in multi-hop ad hoc networks. In *Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 64–71, Seattle, WA, 1999.

[80] Qingfeng Huang, Chenyang Lu, and Gruia-Catalin Roman. Mobicast: Just-in-time multicast for sensor networks under spatiotemporal constraints. WUCS-TR 42, Washington Unievrsity in Saint Louis, 2002.

[81] Qingfeng Huang, Chenyang Lu, and Gruia-Catalin Roman. Design and analysis of spatiotemporal multicast protocols for wireless sensor networks. WUCSE-TR 45, Washington Unievrsity in Saint Louis, 2003.

[82] Qingfeng Huang, Chenyang Lu, and Gruia-Catalin Roman. Mobicast: Just-in-time multicast for sensor networks under spatiotemporal constraints. In *Proc. of the 2nd International Workshop on Information Processing in Sensor Networks*, pages 442–457, Palo Alto, CA, USA, April 2003.

[83] Qingfeng Huang, Chenyang Lu, and Gruia-Catalin Roman. Mobicast: Just-in-time multicast for sensor networks under spatiotemporal constraints. In *Proc. of the 2nd International Workshop on Information Processing in Sensor Networks*, pages 442–457, Palo Alto, CA, USA, April 2003.

[84] Qingfeng Huang, Chenyang Lu, and Gruia-Catalin Roman. Spatiotemporal multicast in sensor networks. WUCSE-TR 18, Washington Unievrsity in Saint Louis, 2003.

[85] Qingfeng Huang, Chenyang Lu, and Gruia-Catalin Roman. Spatiotemporal multicast in sensor networks. to appear in ACM SenSys '03, November 2003.

[86] Qingfeng Huang, Ronald Miller, Perry McNeille, David Dimeo, and Gruia-Catalin Roman. Development of a peer-to-peer collision warning system. *Ford Techincal Journal*, 5(2), March 2002.

[87] IBM. TSpaces, web page. http://www.almaden.ibm.com/cs/TSpaces.

[88] Tomasz Imielinski and Julio C. Navas. Gps-based addressing and routing. RFC2009, Computer Sciece, Rutgers University, March 1996.

[89] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann. Impact of network density on data aggregation in wireless sensor networks. International Conference on Distributed Computing Systems (ICDCS-22), 2001.

[90] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Mobile Computing and Networking*, pages 56–67, 2000.

[91] ITS. Problem Area Descriptions, Motor Vehicle Crashes - Data Analysis and IVI Program Emphasis. USDOT and ITS Joint Program, November 1999.

[92] Richard Jackson. Running down the up-escalator: Regional inequality in papua new guinea. *Australian Geographer*, 14:175–84, May 1979.

[93] J.W. Jaromczyk and G.T. Toussaint. Relative neighborhood graphs and their relatives. *Proc. of IEEE*, 80(9):1502–1517, 1992.

[94] JavaSpaces. The JavaSpaces Specification web page. http://www.sun.com/jini/specs/js-spec.html, 1999.

[95] X. Jiang and T. Camp. Review of geocasting protocols for a mobile adhoc network. In *Proceedings of the Grace Hopper Celebration(GHC)*, 2002.

[96] D.B. Johnson. Routing in Ad Hoc Networks of Mobile Hosts. In *Proc. of the Workshop on Mobile Computing Systems and Applications*, pages 158–163, 1994.

[97] Tejas karkhanis, James E. Smith, and Pradip Bose. Saving energy with just in time instruction delivery. In *Proceedings of the 2002 international symposium on Low power electronics and design*, pages 178–183, August 2002.

[98] B. Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2000)*, pages 243–254, 2000.

[99] Y. Ko and N. Vaidya. Geocasting in mobile ad hoc networks: Location-based multicast algorithms. In *Proc. of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, pages 101–110, New Orleans, LA, February 1999.

[100] Y. Ko and N. Vaidya. Geotora: A protocol for geocasting in mobile ad hoc networks, 2000.

[101] Y.B. Ko and N.H. Vaidya. Location-Aided Routing (LAR) in Mobile Ad Hoc Networks. In *Proc. ACM/IEEE MOBICOM '98*, October 1998.

[102] Young-Bae Ko and Nitin H. Vaidya. Location-based multicast in mobile ad hoc networks. Technical Report TR98-018, Texas A&M University, 3, 1998.

[103] Evangelos Kranakis, Harvinder Singh, and Jorge Urrutia. Compass routing on geometric networks. In *Proc. 11 th Canadian Conference on Computational Geometry*, pages 51–54, Vancouver, August 1999.

[104] F. Kuhn, R. Wattenhofer, and A. Zollinger. Geometric ad-hoc routing for unit disk graphs and general cost models. Technical Report 373, ETH Zurich,Department of Computer Science, Zurich, 2002.

[105] Fabian Kuhn and Roger Wattenhofer. Asymptotically optimal geometric mobile ad-hoc routing.

[106] Fabian Kuhn, Roger Wattenhofer, Yan Zhang, and Aaron Zollinger. Geometric Ad-Hoc Routing: Of Theory and Practice. In *Proceedings of the 22nd ACM International Symposium on the Principles of Distributed Computing (PODC)*, Boston, Massachusetts, USA, 2003.

[107] S. Lee, W. Su, and M. Gerla. On-demand multicast routing protocol in multihop wireless mobile networks. *ACM/Baltzer Mobile Networks and Applications*, 2000. special issue on Multipoint Communication in Wireless Mobile Networks.

[108] D. Li, K. Wong, Y.H. Hu, and A. Sayeed. Detection, classification and tracking of targets in distributed sensor networks. *IEEE Signal Processing Magazine*, 19(2), March 2002.

[109] Xiang-Yang Li, Gruia Calinescu, and Peng-Jun Wan. Distributed construction of planar spanner and routing for ad hoc wireless networks. In *Proceedings of IEEE INFOCOM*, 2003.

[110] W.-H. Liao, Y.-C. Tseng, K.-L. Lo, and J.-P. Sheu. Geogrid: A geocasting protocol for mobile ad hoc networks based on grid. *Journal of Internet Technology*, 1(2):23–32, 2000.

[111] Annegret Liebers. Planarizing graphs - a survey and annotated bibliography. *Journal of Graph Algorithms and Applications*, 5(1):1–74, 2001.

[112] H. Lim and C. Kim. Multicast tree construction and flooding in wireless ad hoc networks. In *Proceedings of 3rd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2000.

[113] Juan Liu, Dragan Petrovic, and Feng Zhao. Multi-step information-directed sensor querying in distributed sensor networks. In *Proceedings of the International Conference in Acoustics, Speech and Signal Processing (ICASSP)*, 2003.

[114] Juan Liu, James Reich, and Feng Zhao. Collaborative in-network processing for target tracking. *EURASIP Journal on Applied Signal Processing*, 2003(4):378–391 month =, 2003.

[115] Nancy Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, Inc., 1996.

[116] M. Gerla and C. Chiang and L. Zhang. Tree multicast strategies in mobile, multihop wireless networks. *ACM/Baltzer Mobile Networks and Applications*, 4(3):193–207, October 1999. special issue on Mobile Ad Hoc Networking.

[117] Samuel Madden, Michael Franklin, Joseph Hellerstein, and Wei Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. OSDI 2002, Boston MA.

[118] Michael J. Markowski and Adarshpal S. Sethi. Analysis of a soft real-time random access protocol. CIC TR 96-02, University of Delaware, Newark, DE 1996.

[119] M. Mauve, J. Widmer, and H. Hartenstein. A survey on position-based routing in mobile ad hoc networks, 2001.

[120] Sun Microsystems. Jini. http://www.sun.com/jini.

[121] Ronald Miller and Qingfeng Huang. An Adaptive Peer-to-Peer Collision Warning System. In *Proceedings of the IEEE Vehicle Technology Conference (Spring)*, Birmingham, Alabama, May 2002.

[122] R. Milner. *Communicating and Mobile Systems: The π-Calculus*. Cambridge University Press, 1999.

[123] L. E. Moser, P. M. Melliar-Smith, D. A. Agarwal, R. K. Budhia, and Colleen A. Lingley-Papadopoulos. Totem: A fault-tolerant multicast group communication system. *Communications of the ACM*, 39(4):54–63, 1996.

[124] A. L. Murphy, G. P. Picco, and G.-C. Roman. Lime: A middleware for physical and logical mobility. In *Proceedings of the 21st International Conference on Distributed Computing Systems*, pages 521 – 533, 2001.

[125] A. L. Murphy, G. P. Picco, and G.-C. Roman. Lime: A coordination middleware supporting mobility of agents and hosts. Technical Report WUCSE-03-21, Washington University Department of Computer Science and Engineering, St Louis, 2003.

[126] Raimundo Macedo Nadjib Badache, Michel Hurfin. Solving the consensus problem in a mobile environment.

[127] Giri Narasimhan and Michiel H. M. Smid. Approximating the stretch factor of euclidean graphs. *SIAM J. Comput.*, 30(3):978–989, 2000.

[128] Julio C. Navas and Tomasz Imielinski. Geocast - geographic addressing and routing. In *Proc. of the 3rd Annual Intl. Conf. on Mobile Computing and Networking (MobiCom)*, pages 66–76, 1997.

[129] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping Sheu. The broadcast storm problem in a mobile ad hoc network. In *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pages 152–162, August 1999.

[130] Katia Obraczka, Gene Tsudik, and Kumar Viswanath. Pushing the limits of multicast in ad hoc networks. International Conference on Distributed Computing Systems(ICDCS), 2001.

[131] U.S. Department of Agriculture. *Will there be enough food? The 1981 yearbook of agriculture*. Government Printing Office, Washington, D.C., 1981.

[132] University of Chicago Press. *The Chicago Manual of Style*. University of Chicago Press, Chicago, 13th edition, 1982.

[133] V. Park and S. Corson. Temporally-Ordered Routing Algorithm (TORA) Version 1 Functional Specification. Internet Draft, October 1999. IETF Mobile Ad Hoc Networking Working Group.

[134] Vincent D. Park and M. Scott Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *INFOCOM (3)*, pages 1405–1413, 1997.

[135] S. Pattem, S. Poduri, and B. Krishnamachari. Energy-quality tradeoffs for target tracking in wireless sensor networks. In *The 2nd Workshop on Information Processing in Sensor Networks (IPSN 2003), April 2003*, 2003.

[136] W. Peng and X. Lu. On the reduction of broadcast redundancy in mobile ad hoc networks. In *Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, 2000.

[137] C. Perkins. IP Mobility Support. RFC 2002, IETF Network Working Group, 1996.

[138] C.E. Perkins, E.M. Royer, and S.R. Das. Ad Hoc On Demand Distance Vector (AODV) Routing. Internet Draft, October 1999. IETF Mobile Ad Hoc Networking Working Group.

[139] G.P. Picco and A.L. Murphy. `lighTS` web page.

[140] G.P. Picco, A.L. Murphy, and G.-C. Roman. LIME: Linda meets mobility. In *Proceedings of the 21st International Conference on Software Engineering (ICSE)*, pages 368–377, May 1999.

[141] R. Prakash and R. Baldoni. Architecture for Group Communication in Mobile Systems. In *Proceedings of the IEEE Symposium on Reliable Distributed Systems (SRDS)*, pages 235–242, October 1998.

[142] R. Prakash, M. Raynal, and M. Singhal. An Adaptive Causal Ordering Algorithm Suited to Mobile Computing Environments. *Journal of Parallel and Distributed Computing*, pages 190–204, March 1997.

[143] R. Prakash and M. Singhal. A Dynamic Approach to Location Management in Mobile Computing Systems. In *Proceedings of the 8^{th} Int. Conf. on Software Engineering and Knowledge Engineering (SEKE'96)*, pages 488–495, June 1996.

[144] R. Prakash and M. Singhal. Dynamic Hashing + Quorum = Efficient Location Management for Mobile Computing Systems. In *Proc. of the 16^{th} Annual ACM Symp. on Principles of Distributed Computing (PODC)*, page 291, Santa Barbara, CA, USA, August 1997. ACM Press.

[145] Amir Qayyum, Laurent Viennot, and Anis Laouiti. Multipoint relaying: An efficient technique for flooding in mobile wireless networks. Technical Report Research Report RR-3898, INRIA, February 2000.

[146] R. Alur and T.A. Henzinger. Logics and Models of Real-Time: A Survey. In *Real Time: Theory in Practice*, volume 600, pages 74–106. Springer-Verlag, 1991.

[147] R. Friedman R. Baldoni and R. van Renesse. The Hierarchial Daisy Chain Architecture for Causal Delivery. In *Proc. of the 17^{th} IEEE Int. Conf. on Distributed Computing Systems*, pages 570–577. IEEE Press, 1997.

[148] A. Ricciardi and K. Birman. Process Membership in Asynchronous Environments. Technical Report 93-1328, Cornell University, Department of Computer Science, February 1993.

[149] G.-C. Roman, Q. Huang, and A. Hazemi. Consistent group membership in ad hoc networks. In *Proceedings of the 23^{rd} International Conference on Software Engineering (ICSE)*, May 2001.

[150] Gruia-Catalin Roman, Qingfeng Huang, and Ali Hazemi. On Maintaining Group Membership Data in Ad Hoc Networks. Wucs-00-09, Washington University in St. Louis, Department of Computer Science, April 2000.

[151] E. Royer and C. Toh. A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks. *IEEE Personal Communications*, pages 46–55, April 1999.

[152] Elizabeth M. Royer and Charles E. Perkins. Multicast operation of the ad-hoc on-demand distance vector routing protocol. In *Mobile Computing and Networking*, pages 207–218, 1999.

[153] Jeffrey Salowe. On euclidean spanner graphs with small degree. ACM Symposium on Computational Geometry 1992, 186-191, 1992.

[154] Peter Seiler, Bongsob Song, and J. Karl Hedrick. Development of a Collision Avoidance System. In *SAE Conference Preceedings*, pages 97–103, 1998.

[155] I Stojmenovic. Voronoi diagram and convex hull based geocasting and routing in wireless networks. TR TR-99-11, University of Ottawa, December 1999.

[156] Ivan Stojmenovic, Mahtab Seddigh, and Jovisa D. Zunic. Internal nodes based broadcasting in wireless networks. In *HICSS*, 2001.

[157] William Jr. Strunk and E. B. White. *The Elements of Style*. MacMillan Publishing Co., New York, 3rd edition, 1979.

[158] Min te Sun, Wu chi Feng, Ten-Hwang Lai, Kentaro Yamada, Hiromi Okada, and Kikuo Fujimura. Gps-based message broadcasting for inter-vehicle communication. In *Proceedings of the 2000 International Conference on Parallel Processing*, Toronto, Canada, August 2000.

[159] Kate L. Turabian. *A Manual for Writers of Term Papers, Theses, and Dissertations*. University of Chicago Press, Chicago, 5th edition, 1987.

[160] R. van Renesse, K.P. Birman, and S. Maffeis. Horus, a flexible Group Communication System. *Communications of the ACM*, 39(4):76–83, April 1996.

[161] R. Vitenberg, I. Keidar, G. Chockler, and D. Dolev. Group communication specifications: A comprehensive study, 1999.

[162] B. Williams and T. Camp. Comparison of broadcasting techniques for mobile ad hoc networks. In *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, pages 194–205, 2002.

[163] Tina Wong, Thomas Henderson, and Randy H. Katz. Tunable reliable multicast for periodic information dissemination. *Mobile Network and Applications*, 7(1):21–36, January 2002.

[164] Jie Wu and Hailan Li. A dominating-set-based routing scheme in ad hoc wireless networks. *Telecommunication Systems*, 18(1-3):13–36, 2001.

[165] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking, (MobiCom 2001), Rome, Italy, July 16-21*, 2001.

[166] Zhengrong Yang, Takashi Kobayashi, and Tsuyoshi Katayama. Development of an Intersection Collision Warning System Using DGPS. In *Intelligent Vehicle Systems(SP-1538)*. SAE World Congress, Detroit, Michigan, March 2000.

[167] Ellen W. Zegura, Kenneth L. Calvert, and Samrat Bhattacharjee. How to model an internetwork. In *IEEE Infocom*, volume 2, pages 594–602, San Francisco, CA, March 1996. IEEE.

[168] Hu Zhou and Suresh Singh. Content based multicast (cbm) for ad hoc networks. Mobihoc 2000, Boston, MA, August 2000.

[169] Hu Zhou and Suresh Singh. Time-space multicast for integrated sensor/battlesite networks. IEEE MILCOM 2000, Los Angelos, CA, October 2000.

Vita

Qingfeng Huang

Date of Birth January 15, 1970

Place of Birth CangNan, China

Degrees
D.Sc. Computer Science, Washington University, August 2003
M.S. Computer Science, Washington University, August 2000
A.M. Physics, Washington University, August 1998
M.S. Physics, Fudan University, July 1995
B.S. Physics, Fudan University, July 1992

Publications

Huang, Q., Lu, C. and Roman, G.-C. (2003). Spatiotemporal Multicast for Sensor Networks, to appear in *Proc. of the First ACM Conference on Embedded Network Sensor Systems (SenSys)* Los Angeles, California.

Veltri G., Huang, Q., Qu, G. and Potkonjak, M. (2003). Minimal and Maximal Exposure Path Algorithms for Wireless Embedded Sensor Networks to appear in *Proc. of the First ACM Conference on Embedded Network Sensor Systems (SenSys)* Los Angeles, California.

Huang, Q., Lu, C. and Roman, G.-C. (2003). Mobicast: Just-in-Time Multicast for Sensor Networks under Spatiotemporal Constraints. In *Proc. of the 2nd International Workshop on Information Processing in Sensor Networks* Palo Alto, California.

Huang, Q., Julien, C. and Roman, G.-C. (2003). Relying on Safe Distance to Achieve Strong Group Membership in Ad Hoc Mobile Environments. (accepted for publication in IEEE Transactions on Mobile Computing).

Huang, Q., Miller, R., MacNeille, P., Roman, G.-C. and DiMeo, D (2002). Development of a Peer-to-Peer Collision Warning System, *Ford Technical Journal* **5**(2), March 2002

Miller, R. and Huang, Q., (2002). An Adaptive Peer-to-Peer Collision Warning System. In *Proc. of the IEEE Vehicular Technology Conference (VTC) Spring 2002*, Birmingham, Alabama.

Roman, G.-C., Julien, C. and Huang, Q. (2002). Network Abstractions for Context Aware Mobile Computing. In *Proceedings of the 24th International Conference on Software Engineering (ICSE)*, pages 363–373. Orlando, Florida, May 2002.

Roman, G.-C., Julien, C. and Huang, Q. (2002). Formal Specification and Design of Mobile Systems. In *Proceedings of the 7th International Workshop on Formal Methods for Parallel Programming: Theory and Applications*, Fort Lauderdale, Florida, March 2002.

Roman, G.-C., Huang, Q. and Hazemi, A. (2001).Consistent Group Membership in Ad Hoc Networks. In *Proceedings of the 23rd International Conference in Software Engineering (ISCE)*, Toronto, Canada, May 2001.

Professional Societies

IEEE Computer Society

August 2003