OPPORTUNISTIC OVERLAYS: EFFICIENT CONTENT DELIVERY IN MOBILE ENVIRONMENTS

A Dissertation
Presented to
The Academic Faculty

by

Yuan Chen
陈 源

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

College of Computing
Georgia Institute of Technology
April 2005

OPPORTUNISTIC OVERLAYS: EFFICIENT CONTENT DELIVERY IN MOBILE ENVIRONMENTS

Approved by:

Dr. Karsten Schwan, Advisor
College of Computing
Georgia Institute of Technology

Dr. Mustaque Ahamad
College of Computing
Georgia Institute of Technology

Dr. Calton Pu
College of Computing
Georgia Institute of Technology

Dr. George Riley
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Dong Zhou
Mobile Software Laboratory
DoCoMo USA Labs

Date Approved: April 6, 2005

To my parents.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor, Professor Karsten Schwan, for his guidance and continuous support throughout my Ph.D studies. His positive attitude and encouragement were crucial to the completion of this dissertation.

I would also like to thank my thesis committee members, Professor Mustaque Ahamad, Professor Calton Pu, Professor George Riley, and Dr. Dong Zhou for their insightful comments that helped improve the quality of this dissertation.

I would like to express my appreciation for the good interaction and collaboration with the CERCS members, especially Dong Zhou, Greg Eisenhauer, Ada Gavrilovska, and Fabian Bustamante. I am also grateful for the friendship and support I received from Yaxin Liu and Li Zou.

I would like to dedicate this dissertation to my parents for their unconditional love, support and encouragement. This work would not have been possible without their devotion and sacrifice. I want to thank my sisters, Chen Lin (陈琳) and Chen Wen (陈文) for all the support they have given me. Finally, I give my deepest love and thanks to my wife Cao Xiaoling (曹小玲), whose love and faith in me have made it all possible.

TABLE OF CONTENTS

LIST OF FIGURES

SUMMARY

Middleware has become a key enabler for the development of distributed applications. Unfortunately, conventional middleware technologies do not yet offer sufficient functionality to make them suitable for mobile environments. This dissertation proposes a novel middleware approach termed *opportunistic overlays* and its dynamically reconfigurable support framework for building efficient mobile applications. Specifically, we address the inefficiency of content delivery introduced by node mobility and by dynamically changing system loads, in the context of publish/subscribe systems. In response to changes in physical network topology, in nodes' physical locations, and in network node behaviors, opportunistic overlays dynamically adapt event dissemination structures (i.e., broker overlays) with the goal of optimizing end-to-end delays in event delivery. Adaptation techniques include the dynamic construction of broker overlay networks, runtime changes of mobile clients' assignments to brokers, and dynamic broker load balancing. Essentially, opportunistic overlays implement a middleware-level analogue of the networking routing protocols used in wireless communications (i.e., Mobile IP, AODV, DSR and DSDV). By thus coordinating network- with middleware-level routing, opportunistic overlays can attain substantial performance improvements over non-adaptive event systems. Such improvements are due to their use of shorter network paths and better balancing of loads across event brokers.

Opportunistic overlays and the adaptive methods they use are realized by a set of distributed protocols implemented in a Java-based publish/subscribe infrastructure. Comprehensive performance evaluations are performed via simulation, emulation, and with two representative applications on actual networks. Experimental results demonstrate that the opportunistic overlay approach is practically applicable and that the performance advantages attained from the use of opportunistic overlays can be substantial, in both infrastructure-based mobile environments and mobile ad hoc networks.

CHAPTER I

INTRODUCTION

With the increased availability of powerful mobile computing devices like laptops and iPAQs, and the widespread deployment and use of wireless data communications, mobile computing as a new computing paradigm is becoming practical, in which the mobile user carrying a portable device can have continuous access to remote services and resources independent of physical location. However, the highly variable computing environments and limited resources that characterize typical mobile platforms present challenges to the application programmers who develop mobile applications. Issues include the limited capacities of mobile devices (slow CPUs, small memory, small displays, and limited battery life), varying availabilities of network resource and poor network connectivities (low bandwidth, high error rate, higher delay, and frequent disconnection), and node mobility.

Middleware has become a key enabler for the development of distributed applications. Unfortunately, conventional middleware technologies do not yet offer sufficient functionality to make them suitable for mobile environments. In response, this dissertation proposes a novel middleware approach and its dynamically reconfigurable support framework for building efficient mobile applications. Specifically, we address the inefficiency of content delivery introduced by node mobility and by dynamically changing system loads, in the context of publish/subscribe systems.

1.1 Motivation

Publish/subscribe is a widely used method for providing anonymous, inherently asynchronous group communications in distributed settings. As shown in Figure 1.1, a publish/-subscribe system has three main elements: (1) an event producer generating (publishing) events (messages), (2) an event consumer who declares its interest in receiving certain events via a subscription, and (3) an event broker responsible for collecting and processing

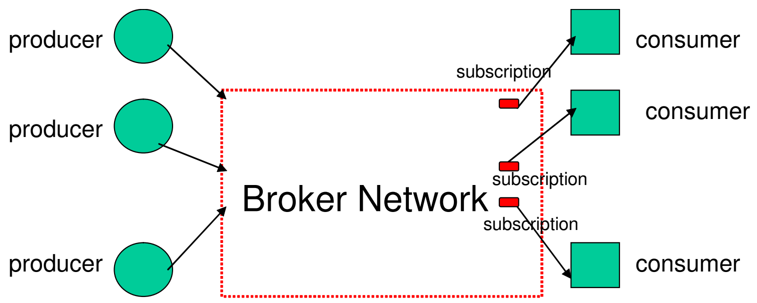


Figure 1.1: Publish/Subscribe Communication

event subscriptions and for routing processed events to corresponding consumers. Most publish/subscribe architectures employ an application-level broker (i.e., overlay) network to perform content-based routing of events at application-level. Broker networks can be organized in multiple ways, ranging from hierarchical tree topologies to acyclic undirected graphs (acyclic peer-to-peer), to generic graph topologies [9].

Past work has created numerous publish/subscribe systems, in industry and in academia [99, 10, 22, 38, 93, 29, 100, 103, 25, 116]. Certain features of publish/subscribe make it well-suited to mobile environments, including asynchronous event delivery, anonymity, multipoint communication and content-based routing [43, 23, 27]. Current systems targeting Internet-based communications, however, commonly assume a fixed network environment in which clients do not move and where the network topology remains relatively stable. Stated more technically, they assume statically deployed broker networks (i.e., overlays) mapped to static network topologies. The resulting problem for mobile environments is a mismatch between static broker network topologies and dynamic underlying network topologies. This mismatch will result in inefficiencies in event delivery, a simple example being a shortest path in the original overlay and physical network turning into an inefficient path when the same logical overlay is used with a different network topology. Another example is a longer event delivery path between client and broker experienced by a client that moves 'far away' (physically) from the broker to which it initially connected.

Depending on the availability of a fixed network, mobile environments can be classified into two types: (1) infrastructure-based mobile environments where brokers reside on fixed networks and a mobile client connects to the fixed network by an access point(base-station

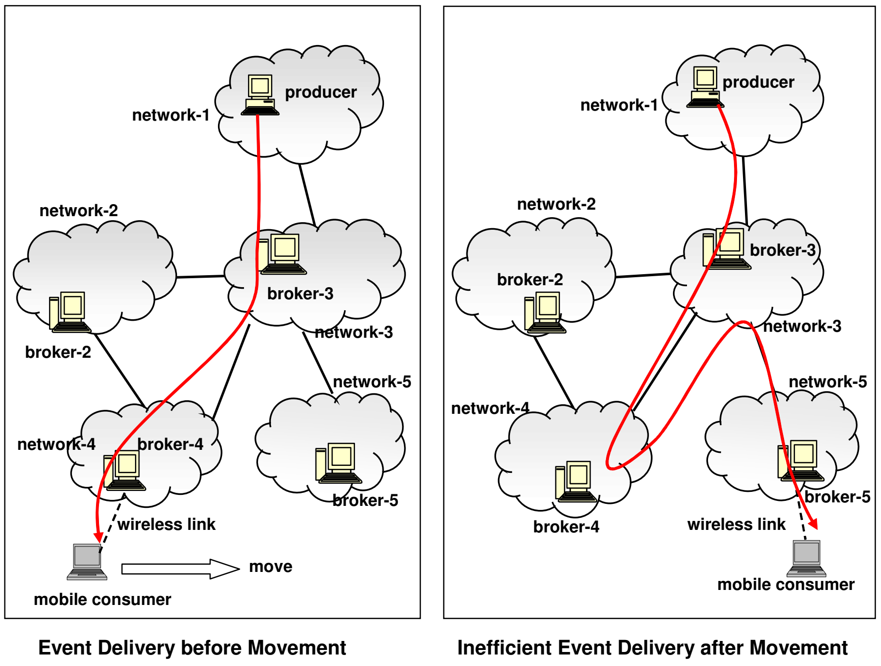
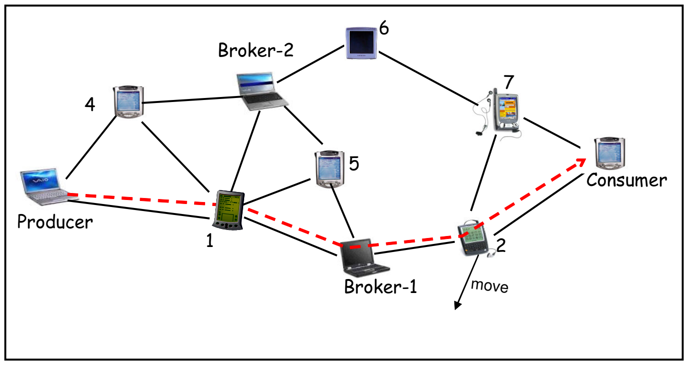


Figure 1.2: An Example of Inefficient Event Delivery with Fixed Broker Assignment in Infrastructure based Mobile Environments

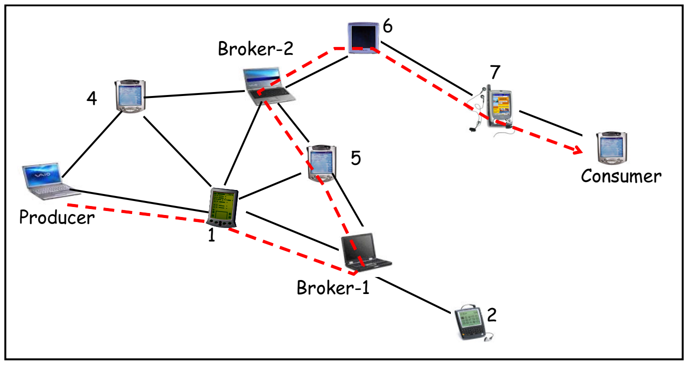
in wireless LAN); and (2) wireless ad-hoc networks where both brokers and clients are mobile and cooperate to form an on-the-fly network without requiring centralized access point or existing communication infrastructure.

- **Producer/Consumer Static Broker Assignments.**

  For infrastructure-based mobile environments, the fixed broker assignment of a mobile producer/consumer will result in inefficient event delivery due to the mobile client's movement. A specific example is one in which a location change by an event producer or consumer results in a consequent change of the network access point used by the underlying Mobile IP protocol [76, 54]. This can lead to inefficiencies in broker communications, as depicted in Figure 1.2, where a mobile consumer's crossing of a network boundary results in a circuitous path from the producer to the consumer's assigned broker to the consumer. Initially, the mobile consumer connects to the event system by broker-4 on network-4, and receives events from a producer on network-1. The resulting delivery path is *network-1(producer)* $\rightarrow$ *network-3(broker-3)* $\rightarrow$ *network-4(broker-4)* $\rightarrow$ *consumer*. At some point time, the consumer moves from



(a) Event Delivery before Node 2's Movement



(b) Inefficient Event Delivery after Node 2's Movement

Figure 1.3: An Example of Inefficient Event Delivery with Fixed Broker Assignment in Mobile Ad Hoc Networks

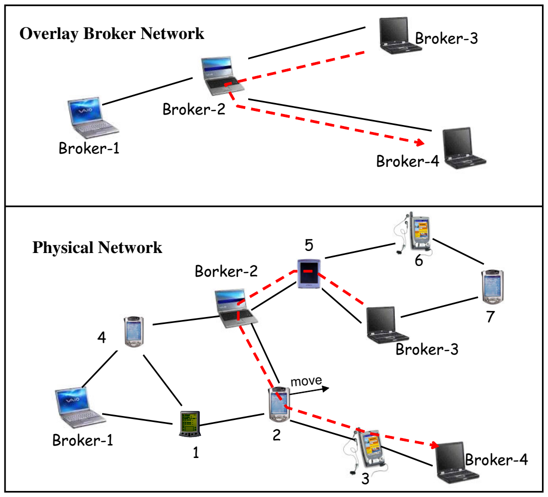
network-4 to network-5. The underlying network protocol such as Mobile IP can automatically connect the mobile consumer to an access point on network-5 and route data to the mobile consumer's new location [76, 54]. Moreover, the optimization of mobile IP routing can send packets to the mobile consumer's new location without going through its initial access point on network-4 [76, 57]. However, if the mobile consumer still uses broker-4 as its attachment to the event system, the content delivery on broker network cannot take advantage of the underlying IP routing optimization and the resulting event delivery path *network-1(producer)* → *network-3(broker-3)* → *network-4(broker-4)* → *network-3* → *network-5* → *consumer* is obviously an inefficient one, since events can be delivered following the much shorter path *network-1(producer)* → *network-3(broker-3)* → *network5(broker-5)*.

As demonstrated by the example above, in infrastructure-based mobile environments, inefficiencies are caused by mobile producer/consumer movements, i.e., when a mobile producer/consumer moves out of range of its old network and into range of new one. In mobile ad hoc networks, any node's movement could affect the underlying network topology and change the distance between a mobile consumer and the assigned broker, hence resulting in sub-optimal event delivery. For example, the original path could become inefficient when the client's assigned broker moves away from the client or when a new broker moves closer to the client. Another example is the movement of an intermediate node on the path between the client and its assigned broker changes the distance between the client and the broker. In a word, the inefficiency caused by fixed broker assignment becomes more frequent and more serious in wireless ad hoc networks due to more frequent node mobility in such environments.

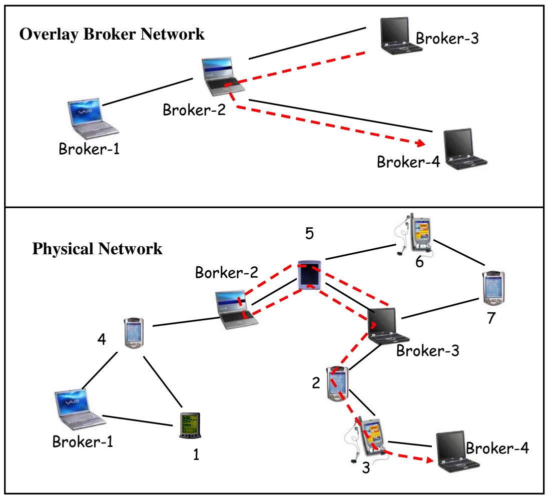
An example in wireless ad hoc networks is shown in Figure 1.3, where an intermediate node's mobility causes a consumer to be 'far away' from its connected broker, resulting in a circuitous path from the producer to the consumer. Specifically, as node *2* moves away from node *7* and *Consumer*, the event delivery path between *Producer* and *Consumer* becomes *Producer* → *1* → *Broker-1* → *5* → *Broker-2* → *6* → *7* → *Consumer*, with 6 intermediate hops. A better path between *Producer* and *Consumer* can be achieved by selecting *Broker-2* as *Consumer*'s new home broker. The new path *Producer* → *4* → *Broker-2* → *6* → *7* → *Consumer* uses only 4 intermediate hops.

- **Static Broker Network Topologies.** Although it is reasonable to assume the interconnections among brokers remain unchanged in an infrastructure-based environment where brokers reside on relatively steady physical network(i.e., Internet), the underlying physical networks change constantly in mobile ad hoc networks, and static broker network topologies will result in inefficient event delivery.

  Figure 1.4 depicts a simple example where a location change by a mobile node results in a consequent change of the underlying physical network topology and therefore, causes a mismatch between the broker network and the physical network. As node *2* moves away from *Broker-2* and node *1* and closer to *Broker-3*, the underlying physical transport infrastructure disconnects the wireless link between *2* and *Broker-2* and the



(a) Initial Topology



(b) Topology after Node 2's Movement

Figure 1.4: An Example of Inefficient Event Delivery with Static Broker Network in Mobile Ad Hoc Networks

link between *2* and *1*, and then creates one new wireless link between *2* and *Broker-3*. As a result, event delivery from *Broker-3* to *Broker-4* based on the old broker network topology follows the path *Broker-3* → *5* → *Broker-2* → *5* → *Broker-3* → *2* → *3* → *Broker-4*, which traverses 7 wireless links. Obviously, this path is highly inefficient, since *Broker-3* could send data to *Broker-4* using the much shorter path *Broker-3* → *2* → *3* → *Broker-4*.

- **Lack of Load Balancing across Brokers.** In addition to inefficient delivery paths, another problem associated in mobile environments is the development of potential load imbalances across brokers due to node mobility and/or due to the use of complex event processing methods. With clients changing locations, broker loads are subject to substantial runtime variation. One reason may be the sudden arrival of large numbers of local users, exemplified by many mobile units converging at a meeting. Another reason is the use of complex event 'modulators' by 'thin' clients, such as modulators that implement the flexible data transcoding required by such clients [116, 108]. As shown in Figure 1.5, many consumers close to broker-4 may overload broker-4, by placing many subscriptions at broker-4. Without overload control, the processing time of a modulator on a highly loaded broker can exceed

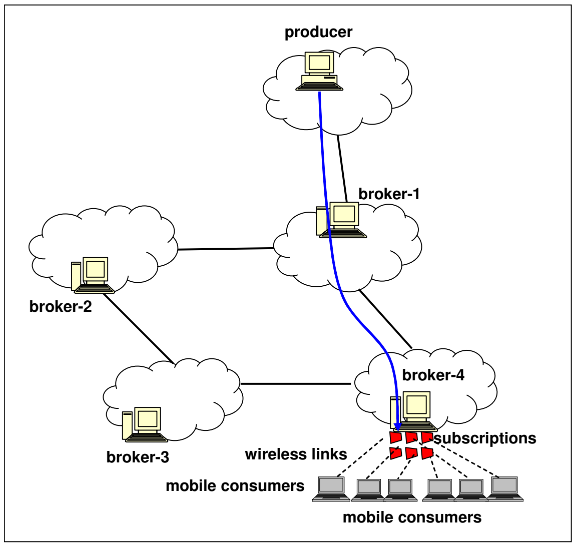


Figure 1.5: An Example of Broker Overloading

network delays and result in poor end-to-end latency even if the event delivery path is optimized.

Most publish/subscribe systems do not take into account broker loads when they distribute or propagate consumers' subscriptions, partly because their subscriptions are simple predicates defined in event fields. As demonstrated in [116], simple predicate-based subscription is not sufficient for implementing the event processing needed by the complex data conversions present in many multimedia, business, and scientific applications. Furthermore, they are unable to realize application-specific tradeoff in order to address resource-limitations existing in mobile systems. To make publish/-subscribe a useful paradigm for mobile systems, it is therefore important to consider the loads implied by event processing.

In a summary, the following challenges have been identified when mapping publish/subscribe infrastructures and more generally, current distributed middleware to mobile environments:

- Producers/consumers' static attachments to brokers.
- Fixed broker network topologies.
- Lack of load balancing across brokers.

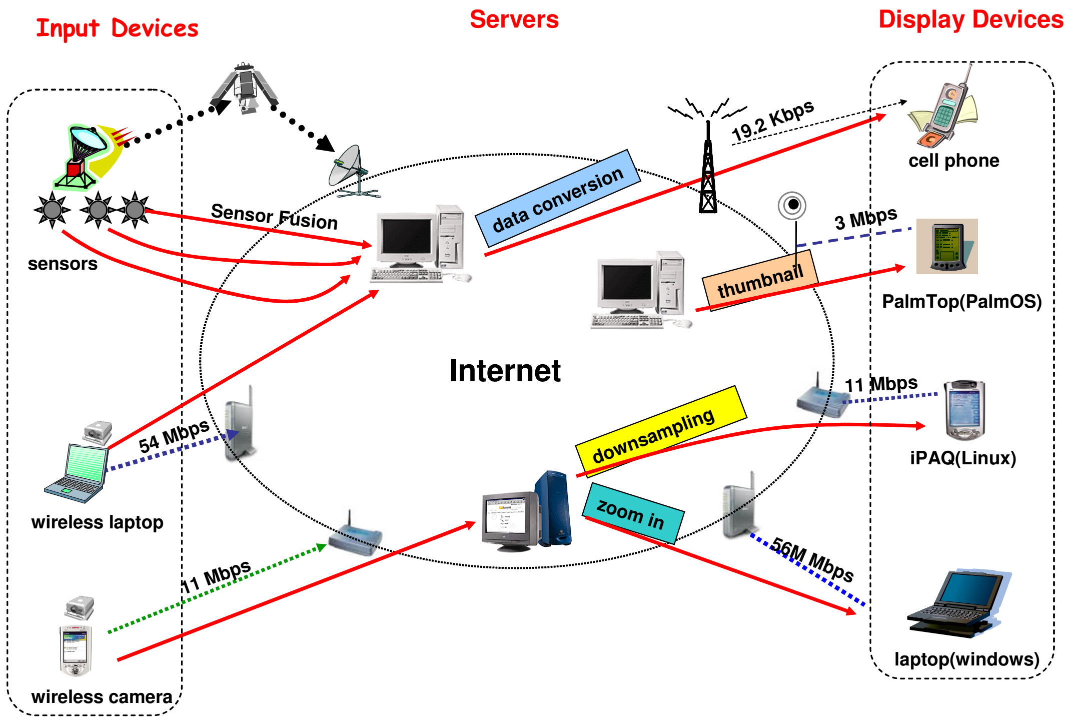


Figure 1.6: A Sample Mobile Application

1.2 Overview of the Opportunistic Overlay Approach

This dissertation proposes the *opportunistic overlay* approach to manage overlays for mobile networks. As shown in Figure 1.6, we target applications in which events are continually emitted by generators and consumed by subscribers, the specific examples used in this thesis concerning data capture, collection, and distribution from remote sensors (e.g., flood data) [32], and a virtual engineering workbench application with the ability of real-time monitoring and controlling remote simulations. Other examples studied in our work include transactional applications like those in Operational Information Systems (OIS) [36] and data-driven scientific applications [107]. The performance metrics applied match our target application domains, focusing on event delivery delays from producers to consumers.

The mobile environments addressed by this thesis include both infrastructure-based wireless networks and mobile ad hoc networks. An infrastructure-based mobile system consists of brokers residing on fixed networks and consumers residing on mobile devices (e.g., a laptop or an iPAQ handheld device). Broker nodes are connected via the Internet, and mobile consumers connect to the Internet via base-stations. A mobile consumer can change its location and accordingly, its connected base station during the application's execution. Specifically, we focus on wide area mobility (global mobility) where mobile clients move between different sites or domains [37]. In mobile ad hoc networks, brokers, producers, and consumers all reside on mobile nodes, organizing themselves to form an on-the-fly underlying physical network without relying on any fixed network infrastructure. In such environments, two nodes typically communicate with each other via some multi-hop wireless network path.

The idea behind the opportunistic overlay approach is to dynamically optimize content-based event delivery by adapting event dissemination structures (i.e., broker overlays) to changes in physical network topology, in nodes' physical locations, and in network node behaviors. The term 'opportunistic' denotes the fact that the solution is one in which each broker opportunistically acts to improve its relations both with other brokers and with its clients. The key points characterizing opportunistic overlays may be summarized as follows:

- **Resource awareness.** An opportunistic overlay is aware of the underlying network topology used for transporting events from producers to consumers. It is also aware of the respective locations of both and of its current state (e.g., CPU Load, Memory availability).

- **Dynamic construction of overlay broker network.** The broker network topology is dynamically constructed to match the underlying wireless network in mobile ad hoc networks.

- **Dynamic change of broker assignment.** When a mobile client moves out of range of its previous network and into range of a new one in infrastructure-base mobile environment, or when a mobile client detects a broker in its vicinity in wireless ad hoc networks, the client's broker assignment is changed based on the client's physical location and the broker's current capabilities.

- **Dynamic overlay routing.** When broker network topologies or clients' brokers assignments change, overlay routing paths are recalculated, and then the newly computed paths are used.

- **Dynamic load balancing.** Brokers' loads are balanced by periodically relocating modulators from overloaded brokers to more lightly loaded ones.

The implementation of these adaptations uses a set of protocols distributed across clients and brokers. The protocols do not rely on specific network protocol properties, so that they can operate with multiple underlying network protocols.

The opportunistic overlay approach uses overlay networks. Advantages derived from their use are well-documented in the general domain of peer-to-peer computing and for publish/subscribe systems [98, 91, 81, 1, 10, 99]. Benefits include increased scalability, robustness, and generality, the latter in part due to the fact that basic network connectivity is handled by underlying network protocols. The opportunistic overlay approach attains generality in that it exploits the physical communication media and/or protocols existing in pervasive environments to reliably deliver events. The result, of course, is that application-level messages must not only traverse multiple machines and networks (with mobility, an overlay link may well traverse the same physical links multiple times), but also multiple broker processes reside on such machines. Despite these facts, for the publish/subscribe domain, intermediate brokers are a necessity, for reasons of scalability, because of limited capabilities of mobile 'thin' clients, and also, as shown in this paper, because dynamic broker network topologies afford us with the ability to significantly improve system performance compared to static event dissemination structures.

Opportunistic overlays are implemented with the JECho Java-based publish/subscribe infrastructure [116]. A unique attribute of this implementation is that with JECho, dynamic topology adjustments can be coupled with runtime techniques for event filtering, thereby also permitting the system to match event rates and sizes to the currently available levels of bandwidth of physical communication channels. This is particularly effective in JECho because it uses general Java functions for event filtering rather than the predicate-based filters supported elsewhere. That is, JECho subscriptions are based on general functions deployed by event consumers 'into' event producers or brokers. These functions not only implement the event processing needed for the complex data conversions present in multimedia, business, or scientific applications, but they can also realize application-specific tradeoffs in performance vs. desired functionality (e.g., event contents) in order to address some of the resource limitations existing in pervasive systems.

1.3 Thesis Statement

Mobile environments cause inefficiencies in content delivery due to node mobility and dynamically changing system loads. Inefficiencies measured as increases in the end-to-end delay of content delivery may be rectified by dynamic adaptation of the overlay networks used for content acquisition and distribution. For publish/subscribe systems and the broker networks they use for event distribution, this may be carried out by efficient distributed protocols jointly executed by event producers, brokers, and consumers.

1.4 Thesis Contributions

This dissertation focuses on middleware for mobile wireless systems, with specific emphasis on publish/subscribe middleware. To deal with dynamic change in mobile systems, middleware-level adaptation mechanisms are developed and evaluated. The primary contributions of this dissertation that provide evidence to support our thesis statement may be summarized as follows:

- **Opportunistic Overlay Approach.** Opportunistic overlays are a new approach to supporting efficient content delivery in wireless infrastructure-based and ad hoc mobile environments. The approach optimizes end-to-end delay from data producers to consumers by dynamically adapting event dissemination structures (i.e., broker overlays) to changes in physical network topology, in nodes' physical locations, and in network node behaviors. Adaptations are transparent to applications, automatically activated by node mobility and by changes in underlying physical networks or devices.

  By using overlay networks, the opportunistic overlay approach capitalizes on known results to attain high levels of scalability, but then improves on such results to address the specific needs of mobile systems. Network and mobility awareness cause runtime adaptations that improve the performance and robustness of event delivery compared to non-adaptive approaches. By operating at the middleware level, the approach not only applies to multiple underlying network protocols, but it also leverages future protocol developments that will further improve mobile network connectivities.

- **Distributed Adaptation Protocols.** The adaptation techniques used in our approach are realized by a set of distributed protocols cooperatively performed across participating brokers. Those protocols are sufficiently powerful to support a variety of opportunistic adaptations. In addition, they are relatively inexpensive and their overheads are controllable. Finally, they operate across heterogeneous systems that use multiple underlying network routing protocols (e.g., Mobile IP, DSR, DSDV, or AODV [76, 56, 77, 78]) since they do not rely on specific network protocol properties.

- **Opportunistic Overlay Framework.** The opportunistic overlay approach is implemented in a Java-based publish/subscribe infrastructure. The implementation adopts a layered architecture and event-driven paradigm. The 'basic component layer' provides the lower level functionalities of resource monitoring and broker information management necessary for implementing different adaptation techniques. Event-driven adaptations are implemented by defining a set of actions to react to specific events received from basic components. By using services provided by basic components, the adaptation code itself can focus on adaptation methods rather than handling low-level details. The interactions between the basic component and adaptation layers use a set of consistent program interfaces and system events. As a result, it is easy for different brokers to define different adaptations based on their capabilities and requirements. In fact, the implementation of an adaptation protocol within the current system is straightforward, as exemplified by the home broker change adaptation that has less than 50 lines Java code. It is also easy to reconfigure and extend the system with new adaptations, such as those needed to handle physical network partition.

- **Performance Study.** In order to understand the performance of distributed event systems in mobile environments and demonstrate the utility and performance advantages attained from the use of the opportunistic approach, experiments use simulation, system emulation in both infrastructure-based mobile environments and in mobile ad hoc networks., and also involve running two representative applications on testbeds in both types of mobile environments.

1.5 Summary of Performance Evaluation

The performance evaluations reported in this dissertation are performed both on actual hardware, to assess basic performance properties and penalties, and via emulation and simulation, to assess the effects of mobility and to better understand the scalability of our approach.

1.5.1 Infrastructure-based Mobile Environments

Simulation experiments are conducted on 10 physical networks with 100 nodes each, generated using the BRITE topology generator [66]. Microbenchmark programs run on PlanetLab testbed [83] to evaluate the performance of the opportunistic approach on actual hardware and networks. Finally, a virtual workbench application [16] is developed and evaluated on a broker network consisting of 4 computers on PlanetLab. Experimental results demonstrate that the performance advantages attained from the use of opportunistic overlays can be substantial, with the end-to-end latency of the workbench application improved by more than 100%. In microbenchmark experiments, the opportunistic approach is shown capable of dealing with potential broker overloads. Simulation results demonstrate that the overheads experienced by opportunistic overlay are inexpensive and controllable. For example, the routing overhead per node is less than 2Kbps for a moderate update period of 30 seconds.

1.5.2 Mobile Ad Hoc Networks

Simulation techniques are used to evaluate the opportunistic overlay approach under various wireless network configurations. The network consists of 100 mobile nodes that randomly roam in a 1000 x 1000 meter square following the random waypoint mobility model [56]. No link layer details, such as MAC protocols are modeled. In this context, results indicate that the delay of sending a message can be improved significantly. In order to evaluate the effects of load balancing, we conduct experiments on an ad-hoc wireless network emulator. The mobility emulator runs on a Linux cluster of 20 nodes with MobiEmu [111] running on each node. Results show that the opportunistic overlay approach is able to both optimize path lengths and address broker overloads. Measurements on a small testbed comprised of three laptops running an AODV implementation [49] show more than a six-fold improvement in the end-to-end delay experienced by events in the flood watch application. The overheads experienced by opportunistic overlay behavior are moderate. For example, the average per broker bandwidth used for state propagation is less than 2Kbps for a moderate size broker network with 40 broker and for a broker update interval of 100 seconds.

1.6 Thesis Organization

The remainder of this thesis is organized as follows. We present the system model used by opportunistic overlays in Chapter 2. We then describe adaptation protocols and algorithms in Chapter 3. The prototype architecture and implementation details are elaborated in Chapter 4. Chapter 5 presents evaluation results in infrastructure-based mobile environments. Experimental results in mobile ad hoc networks appear in Chapter 6. Related work is described in detail in Chapter 7. Chapter 8 summarizes the conclusions and directions for future work.

CHAPTER II

SYSTEM MODEL

We first outline the system model assumed by opportunistic overlays, followed by descriptions of the event systems in infrastructure-based mobile environments and wireless ad hoc networks, respectively.

2.1 Overview of System Model

As illustrated in Figure 2.1, an event system consists of producers, consumers, and a broker network. The latter is an overlay across the physical network, composed of broker processes connected via links. Each overlay link is a unicast tunnel between a broker node pair in the physical network, either an IP-layer path connecting an overlay node pair in fixed network or a multi-hop wireless path in mobile ad hoc networks. Each producer/consumer (mobile client) connects to one of the brokers (usually the nearest one) via one or multiple wireless links. This broker is called the client's *home broker.* A consumer also provides a content-based subscription function termed *modulator*, which operates on event contents to dynamically tailor them to the consumer's current needs.

A consumer's modulator executes in an intermediate broker's address space on behalf of the consumer. The intermediate broker can be any broker(typically, the homebroker) on the overlay path between producers and the consumer. An event generated by a producer is first sent to the producer's home broker, then routed from the producer's home broker to an intermediate broker, processed using the consumer's modulator, routed to the consumer's home broker, and then finally delivered to the consumer via some wireless network links.

Our research is concerned with applications in both infrastructure-based mobile environment and mobile ad hoc networks. We discuss each of them below.

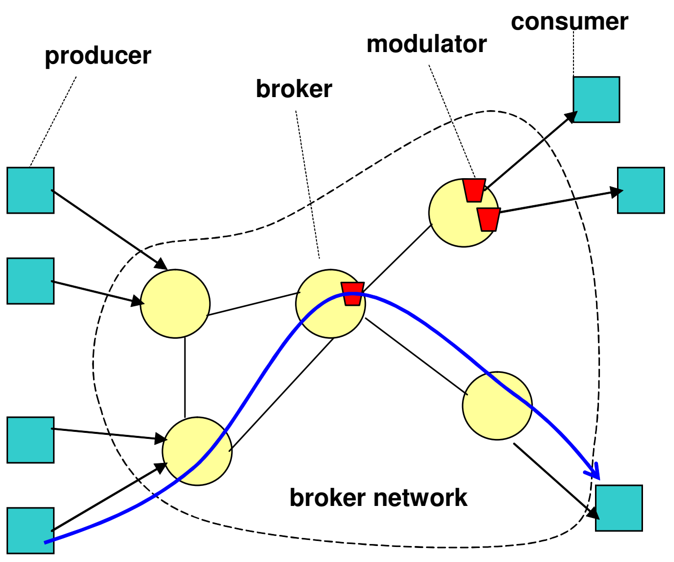


Figure 2.1: System Model

2.2 Infrastructure-based Mobile Systems

An infrastructure-based mobile system consists of brokers residing on fixed networks and consumers residing on mobile devices (i.e., laptop and iPAQ). Broker nodes are connected via the Internet, and mobile consumers connect to the Internet via base-stations. A mobile consumer can change its location and change its connected base station accordingly during its execution. The mobility could be either local mobility where nodes move within a single administrative domain or global mobility where nodes move across domain boundaries [54]. We assume that mobility and wireless network communications are totally handled by the underling network protocols (e.g. Mobile IP [76, 54] and mobile TCP [64, 12]). A mobile client may lose connection unintentionally or intentionally from its home broker during migration, especially when it moves around geographically. If desired, a session layer protocol described in [65] can be used by the broker to maintain the mobile client's TCP connection during the client's migration and supports intermittently connected legacy TCP applications. This can done via home broker's logging events for the client during its disconnection and replaying events on its reconnection.

As shown in our previous work [17], in real scenarios with local mobility such as a wireless campus network, the performance advantages attained from using dynamic home broker

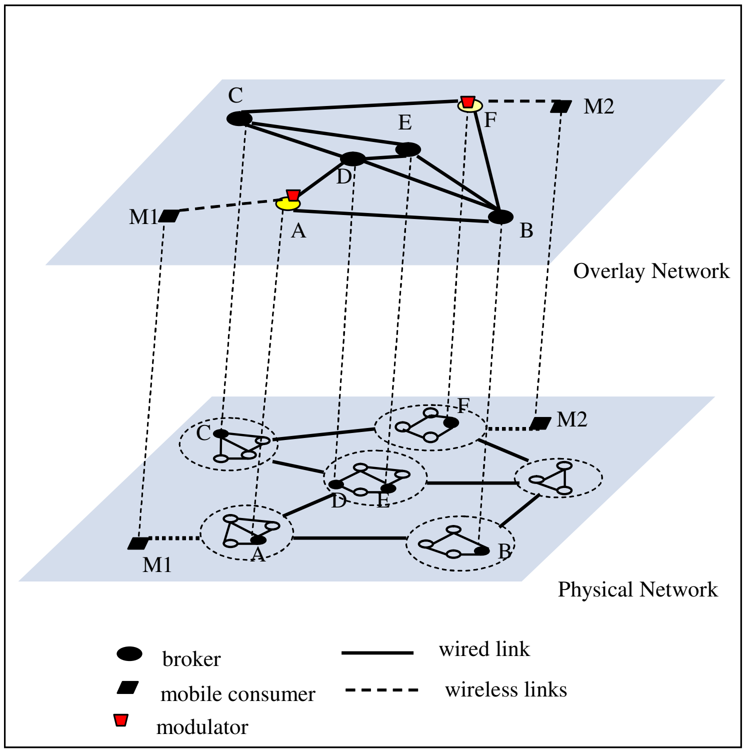


Figure 2.2: A Sample Event System in Infrastructure-based Mobile Environments

change are somewhat small, because the dominant factor in the local mobility scenario is the wireless communication between the broker (access point) and the mobile client. Even with 802.11g, this wireless communication is more than 2 times slower than the wired communication. This will not be the case for wide area mobility or global mobility, with potentially large delays among brokers. Hence, in this thesis, we particularly focus on the scenarios where mobile consumers move across domain boundaries or sites in large Internet topologies. An example is an interactive virtual workbench for remote machine monitoring and control [16] where an engineer perhaps initially inspects an ongoing design simulation from his laptop at his office, then continues his work at home or at a remote location where he is attending a conference. For such applications, timely event delivery is important, since they normally require real-time response. The detailed discussions about the workbench application and its performance appear in Section 5.3.

Figure 2.2 shows an example of an event system in an infrastructure-based mobile environment, which is composed of a set of brokers (A, B, C, D, E and F), and two mobile consumers(M1 and M2). Brokers reside on a fixed IP network and are connected via IP-layer paths. Each mobile consumer connects to a broker via a wireless link (e.g. wireless LAN via base-station). For example, M1 connects to broker A, and A is M1's home broker. M1's modulator runs on A. Actually, M1's modulator can run on any broker on the event path from a producer to M1. A dynamic modulator placement protocol is used in the opportunistic approach to perform load balancing among brokers. The detail is discussed in Section 3.6.

2.3 Mobile Systems in Wireless Ad Hoc Networks

A mobile ad hoc network (MANET) is an autonomous system where all nodes are free to move, organizing themselves to form an on-the-fly underlying physical network without relying on any fixed network infrastructure. Two nodes typically communicate with each other by a multi-hop wireless link between them. In mobile ad hoc networks, all brokers, producers and consumers reside on mobile nodes. A broker link of the broker network corresponds to a multi-hop wireless path on the underlying physical wireless network. A producer/consumer connects to a broker via a multi-hop wireless path, too.

A broker can reside on the same nodes as producers/consumers or on separate nodes. One extreme case is for each node to act as both broker and producer/consumer, and as home broker. Since most nodes in wireless ad-hoc networks have limited resources, however, it is not practical to assume that processors can run arbitrarily complex modulators. Instead, it is often advantageous to use as brokers additional nodes and their resources, such as PCs installed on higher end nearby platforms (e.g., 'command post' vehicles [17, 108]). Unless stated otherwise, this thesis's experimental results assume that brokers run on a set of nodes separate from consumer/producer nodes.

A representative application is used in a flood rescue action. In a flood disaster area, a rescue team is equipped with mobile devices, (e.g., handheld, lightweight laptops or desktops installed on mobile vehicles). All mobile devices have communication hardware (e.g., 802.11b wireless cards) and software (e.g., the AODV protocol [78]) so that they can communicate with each other via the ad-hoc wireless network. Brokers residing on computers

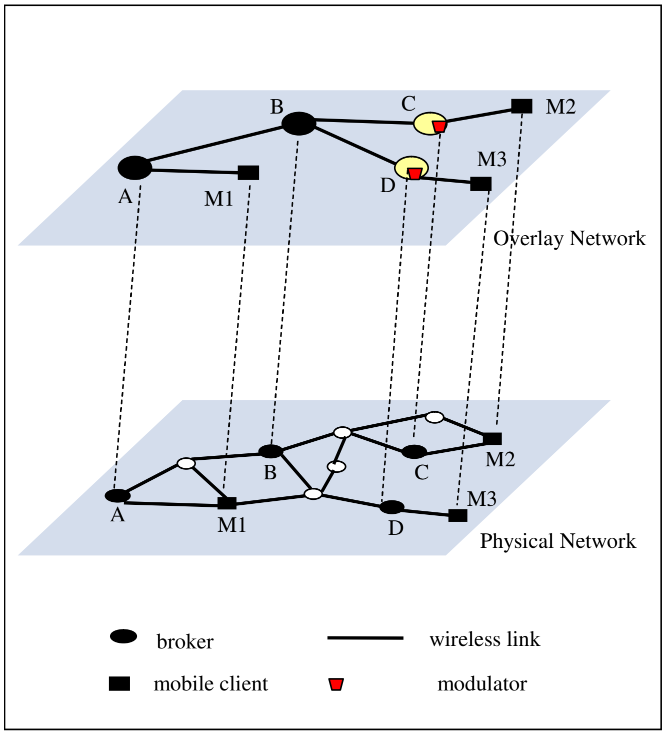


Figure 2.3: A Sample Event System in Mobile Ad Hoc Networks

on mobile vehicles form a virtual network. They receive real weather data either from a weather center via a satellite connection or from sensors distributed across the area. They send collected data to an event channel. The client program running on each team member's mobile computer connects to a broker and receives the data of interest. The client is typically interested in flood information in some specific area, which can be defined by a two-dimensional bounding box. Different people may be interested in receiving different data by running different client programs or by using the same client program with different modulators. The flood application and its performance are discussed in details in Section 6.3.

An event system with four broker nodes (A, B, C and D), one producer M1 and two consumers (M2 and M3) in a wireless ad hoc network is depicted in Figure 2.3. The broker network consists of 3 virtual links (A—B, B—C and B—D). M1's home broker is A and connects to A via the wireless link M1—A. M2 connects to its home broker C via the wireless link M2—C. M3's home broker is D. The figure also shows that mobile clients participate in data routing at the physical network layer, even if their roles in the broker overlay are only to send or receive events. For example, events routed from broker A to broker D are routed through client M1.

2.4 Summary

Our system model adopts an overlay network approach where basic network connectivity is handled by the underlying network protocols. The same approach is used by traditional publish/subscribe systems. By adopting the approach, our model supports publish/subscribe and its semantics, and also extends it to mobile environments.

CHAPTER III

THE OPPORTUNISTIC OVERLAY APPROACH

This chapter first presents the basic idea of the opportunistic overlay approach and then discusses the adaptation protocols underlying it, including three modulator relocation protocols, the broker network topology construction protocol, the dynamic home broker change protocol and the dynamic broker load balancing protocol.

3.1 Basic Idea

The idea behind opportunistic overlays is to continually optimize event delivery, by dynamically changing broker networks and mobile clients' home brokers. Updates occur in response to changes in physical network topology and in nodes' physical locations. Potential broker overloads are avoided by dynamically placing modulators and judiciously choosing clients' home brokers. The key points characterizing opportunistic overlays may be summarized as follows:

- **Resource awareness.** An opportunistic overlay is aware of the underlying network topology used for transporting events from producers to consumers. It is also aware of the respective locations of both and of its current state (e.g., CPU Load, Memory availability).

- **Dynamic construction of broker network topology.** In mobile ad hoc networks, broker network topology is dynamically adapted to changes in the underlying physical network topology. Dynamic broker network topology construction uses a global state routing protocol [15]. Each broker maintains a local view of the broker network topology. At runtime, each opportunistic overlay dynamically monitors client location, physical network topology, and resources (e.g. latency, bandwidth, broker computation load). Periodically, each broker updates its local view of broker network topology, by changing its neighboring brokers and by propagating changes to its neighbors. Neighbor broker information is acquired by querying the network protocols' routing tables or via neighbor discovery operations [40]. When a broker receives propagated information from its neighbors, it updates its broker network topology accordingly.

- **Dynamic change of home broker.** When a mobile client moves out of range of its previous network and into range of a new one in infrastructure-base mobile environments, or when a mobile client in mobile ad hoc networks detects a broker in its vicinity by nearest broker search, it identifies to its current home broker the candidate broker closer to its current physical location. Upon receiving such a report from a client, the home broker initiates a broker selection protocol. This protocol uses an approach that combines shortest path selection with overload control methods. Specifically, the home broker first calculates the path length from the producer to the candidate broker, and then determines whether or not to change the client's home broker based on both the network distance and the candidate broker's current capabilities. Preference is giving to the closer broker unless that broker is currently overloaded.

- **Dynamic overlay routing.** Changes in broker network topology and in mobile clients' home brokers will result in rebuilding broker-level routing tables. Opportunistic overlays use source routing for event delivery, where whenever a broker's local view of broker network topology changes or whenever a client receiving events from a broker changes its home broker, new event paths are calculated using a shortest path algorithm.

- **Dynamic load balancing.** Brokers' loads are balanced by periodically moving modulators from heavily loaded brokers to lighter loaded ones. The load balancing algorithm first chooses brokers on the shortest path to run modulators. If all brokers on the shortest path are overloaded, a less loaded broker on a relatively longer path will be used.

The core of dynamic adaptation is a set of modulator relocation operations, which migrate a consumer's modulator from one broker to another broker on the same path or a different path. The relocation operations provide necessary functionality for implementation of adaptations in opportunistic overlays. We next discuss three modulator relocation protocols, and then details on dynamic construction of broker network topology, dynamic change of home broker, and dynamic load balancing.

3.2 Modulator Relocation Protocols

Three relocation operations, which are horizontal relocation, downstream relocation and upstream relocation, perform the tasks of relocating a consumer's modulator from a specified source broker to a specified destination broker. The operations provide necessary functionality for flexible implementation of different adaptation mechanisms. Both dynamic routing and dynamic load balancing will use these operations to complete runtime adaptations. The modulator relocation protocols described below guarantee correctness properties that include (1) in order event delivery, (2) no lost or duplicate events, and (3) consistent modulator state in the presence of migration. These features are useful in many applications deployed in infrastructure-based mobile systems. For applications in mobile ad hoc networks where the protocols might be costly or applications do not need such strict semantics, variations of the above protocols are available. These lightweight protocols can loosen the requirements of either event state consistency or modulator consistency or both.

3.2.1 Horizontal Relocation

Horizontal relocation is illustrated in Figure 3.1. The relocation protocol relocates a client's modulator from its current home broker (source broker) to a new broker (destination broker), asks all producers' home brokers to compute event paths to the new home broker, and switches event delivery from the old to the new paths.

1. The source broker initiates a handoff by sending a HANDOFF request to the destination broker.

2. Upon receiving the handoff request, the destination broker adds the mobile client to

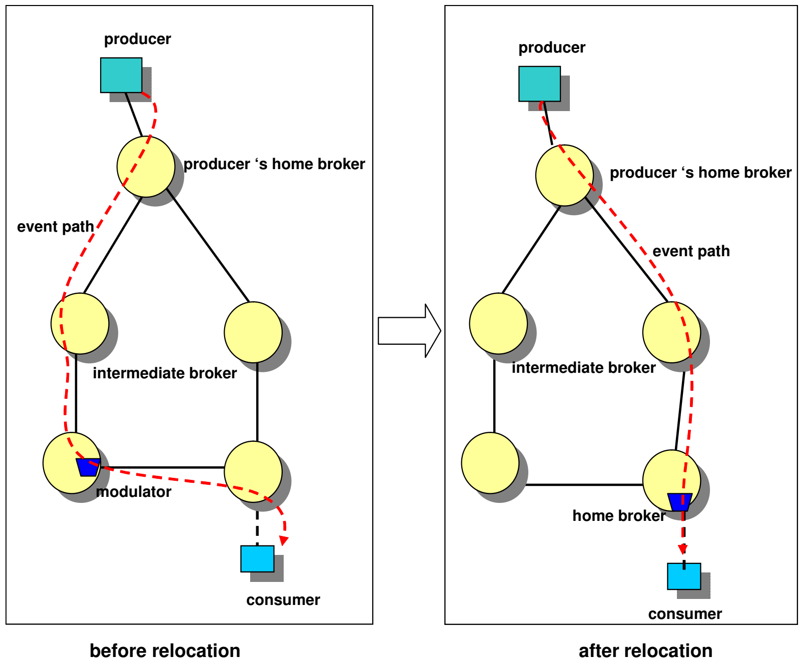


Figure 3.1: Horizontal Relocation

its consumer list and sends an ACK to the source broker.

3. After receiving the ACK from the destination broker, the source broker sends a DETOUR request, which includes the name of the destination broker, to all event producer.

4. Upon receiving the DETOUR request, each producer computes the shortest path to the destination broker, and then atomically switches event delivery from old to new path via changes to its routing table (stop sending events to the source broker along the old path and start sending events to the destination broker along the new path); it then sends ACKS to the source and destination brokers.

5. The source broker receives events from each producer, applies the client's modulator to these events, and forwards them to the next broker, until it receives the ACK from the producer.

6. The destination broker buffers events from each producer after it receives the ACK.

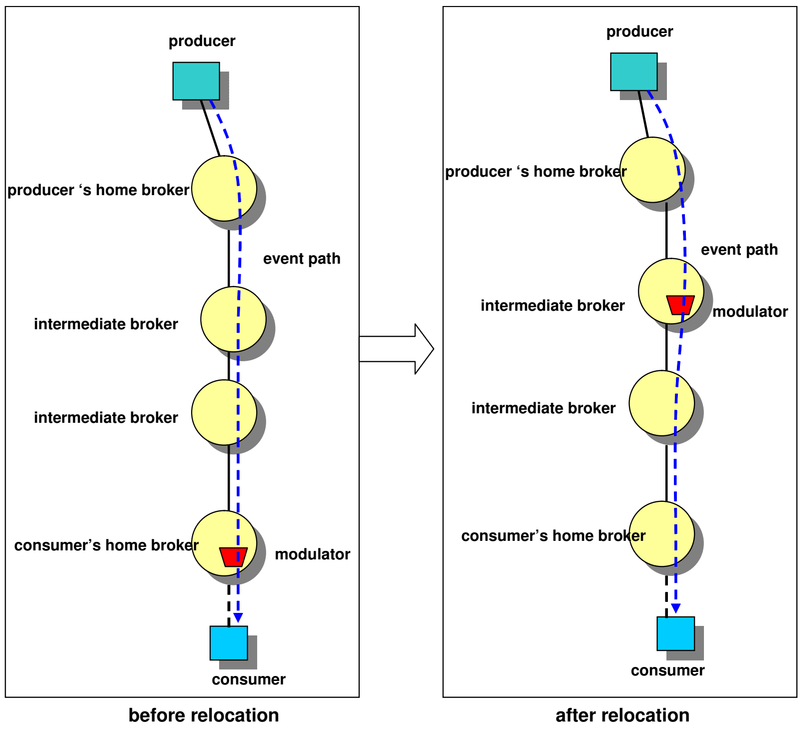


Figure 3.2: Upstream Relocation

7. After ACKs from all producer are received by the source broker, it sends a HANDOFF along with the current modulator to the destination broker and removes the modulator.

8. Upon receiving the HANDOFF, the destination broker applies the modulator received from the source broker to the buffered events and starts forwarding events to the client using the new path.

3.2.2 Upstream Relocation

As depicted in Figure 3.2, upstream relocation is used to relocate a modulator from a broker closer to the consumer to another broker closer to the producer on the event delivery path. It's used in load balancing protocol to dynamically move a client's modulator from an overloaded broker to a less loaded one.

1. The source broker initiates a handoff by sending a HANDOFF request to the destination broker.

2. Upon receiving the HANDOFF request, the destination broker stops forwarding events to next broker downstream, buffers events received from brokers upstream, and sends an ACK to the source broker.

3. The source broker receives events from each producer, applies the client's modulator to these event and sends to next broker, until it receives the ACK from the destination broker.

4. After the ACK is received by the source broker, it sends a HANDOFF along with the current modulator to the destination broker and removes the modulator.

5. Upon receiving the HANDOFF, the destination broker applies the modulator received from the source broker to the buffered events and starts forwarding events to the next brokers.

3.2.3 Downstream Relocation

Figure 3.3 shows the downstream relocation operation, which relocates a modulator from a broker closer to the producer to another broker closer to the consumer on the event delivery path.

1. The source brokers sends a HANDOFF along with the current modulator to the destination broker; and forwards events to the next broker without applying the modulator to these events after that.

2. Upon receiving the HANDOFF, the destination broker applies the modulator received from the source broker to received events and forwards processed events to next broker downstream.

3.2.4 Discussion

The above protocols ensure that no events are lost or duplicated during relocations, since the old broker processes all events from a producer before receiving the producer's ACK mark, while the new broker processes only those events after the ACK message. The order of events from the same producer is maintained, since both the old broker and new

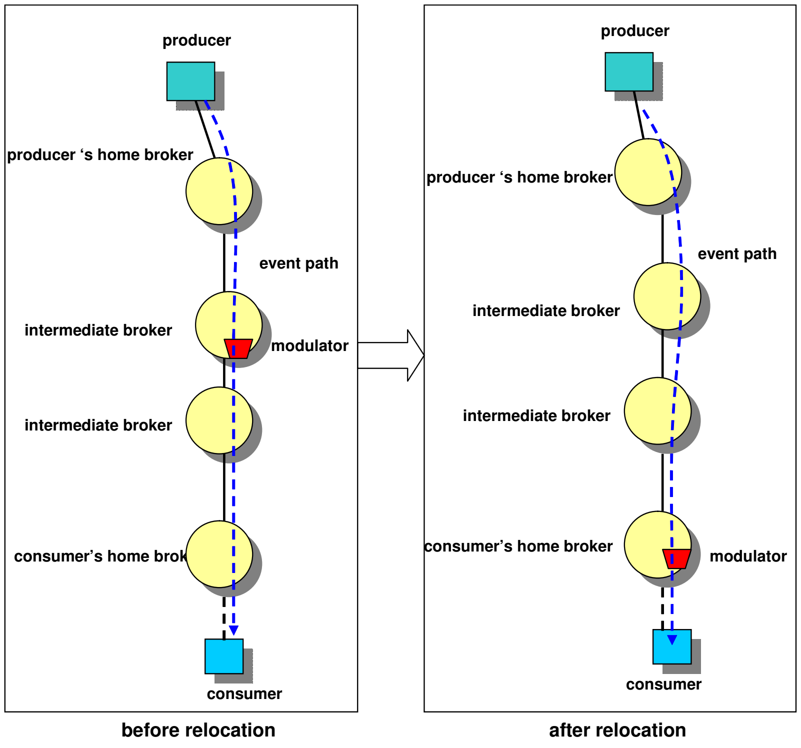


Figure 3.3: Downstream Relocation

broker process the events in the order delivered by the producer. Since modulators are stateful and since event processing may change their states, the protocol described above also ensures the consistency of modulator state. Finally, the algorithm is fairly non-disruptive, as the relocation procedure does not directly affect the other consumers of the producer handled by a certain broker. This is because it requires neither the producer nor the brokers to temporarily stop event delivery. For applications that do not need such strict semantics, variations of the above protocols are possible. Lightweight protocols can loosen the requirements of event state consistency or modulator consistency or both.

3.3 Construction of Broker Networks in Fixed Networks

In infrastructure-based mobile environments, all brokers reside on static nodes and connect with each other by fixed networks. Because the locations of brokers and network connections among brokers on fixed networks are relatively steady, we assume that the overlay broker networks remain unchanged once constructed. Over the past several years, a considerable amount of work has been dedicated to building optimized overlay networks [61]. To connect

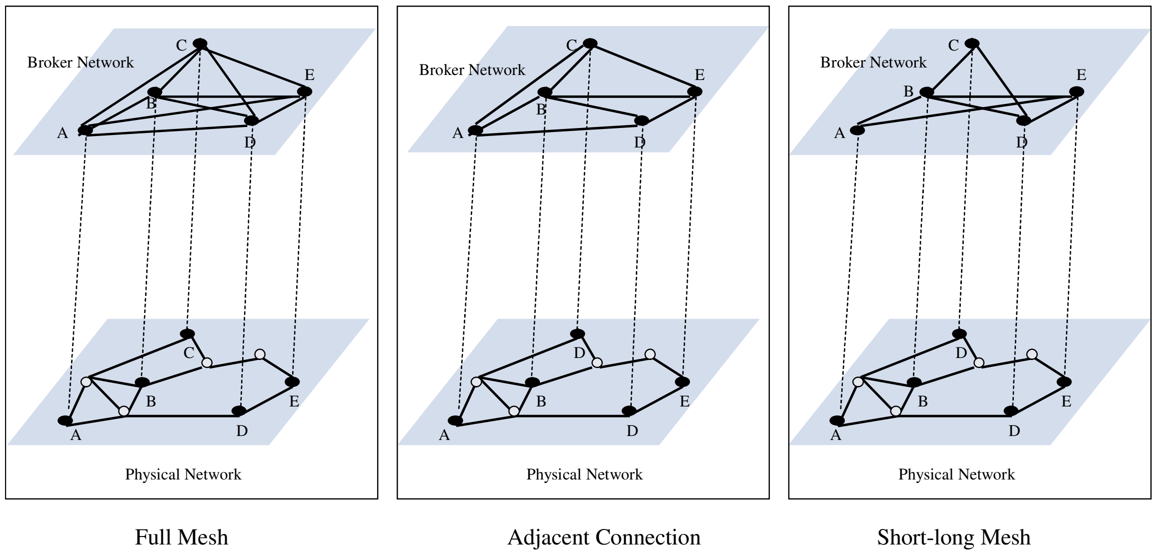


Figure 3.4: Examples of Broker Network Topologies in Fixed Networks

broker nodes, there are many candidate topologies such as full mesh, adjacent connection, regular mesh, K-Minimum-spanning tree, and topology-aware K-minimum-spanning tree and etc. [61]. We briefly discuss each of them below. Detailed research on overlay topologies can be found in [61].

- Full Mesh — there's a broker link between each pair of overlay nodes. Thus each pair of overlay nodes could be neighbors at the overlay layer. It's used in RON (Resilient Overlay Network) [1]. It has been shown that it cannot scale beyond more than 50 nodes [1].

- Adjacent Connection — if there is no other broker on the network layer shortest path between two broker nodes, there is an overlay link between these two brokers [69].

- Short-Long Regular Mesh — each broker picks its k neighbors by picking k/2 brokers closest to itself and then picking another k/2 brokers at random to keep the broker network connected.

- Random Regular Mesh — each broker picks k brokers as its neighbors randomly. This topology is used as a comparison basis for performance evaluation only.

- K-Minimum Spanning Tree — the overlay is composed of K minimal disjoint minimum spanning trees in the full mesh topologies. The K trees have the minimal overlaps of overlay links. Different values of K can be chosen based on the different cost-performance tradeoffs.

- Topology-aware K-Minimum Spanning Tree — a K-Minimum spanning tree is first constructed, then if two overlay links pass through a common physical link, one of them is removed. Thus, the resulting topology has the least overlap of the physical links.

The broker network topology has significant impact on overlay routing in terms of routing performance and routing overhead [61]. For example, full mesh can route events without intermediate overlay nodes, but it incurs routing information maintenance cost and becomes unfeasible for a large overlay network. A random mesh without any network topology knowledge will result in inefficient broker links with high redundancy in physical network layer. The opportunistic overlay approach does not rely on specific properties of a broker network topology, and can operate on top of any broker network topology. Hence we can apply these reported techniques to construct different overlay broker network topologies for event systems in infrastructure-based mobile environments. Four broker network topologies have been implemented in our prototype, and their performance is studied in Chapter 5, including full mesh, adjacent connection, short-long mesh and random mesh. Some examples of broker network topologies in fixed networks are shown in Figure 3.4.

3.4 Dynamic Construction of Broker Networks in Mobile Ad Hoc Networks

In wireless ad hoc networks, both clients and brokers keep changing their locations, dynamic construction of broker networks is required in order to keep broker network topologies congruent with the underlying physical network topology. This is achieved by periodically re-constructing global knowledge about the broker network, using a global state routing protocol [15]. Toward this end, each broker maintains its knowledge about the current broker network topology in a topology table T. Periodically, each broker receives its neighboring broker's T, updates its own T, and then propagates found topology updates to its neighbors.

Each broker keeps track of the other brokers in its vicinity by either querying the routing table maintained by the wireless network routing protocol used in each broker machine, or via a neighbor discovery protocol like the Expanding Ring Search described in [40].

3.4.1 Dynamic Broker Network Construction Protocol

The broker network topology update protocol can be summarized as follows.

- **Step 1: Broker Neighbor Discovery.** Each broker periodically updates its neighboring brokers using the Expanding Ring Search. If a neighbor broker moves too far away from a broker, then the original overlay link between the broker and that neighbor is removed from the broker network. If a broker moves into the vicinity of another broker, a new broker link between them is created. The detail of neighbor discovery is discussed in Section 3.4.2.

- **Step 2: Broker Network Topology Propagation.** Once a broker completes updating its topology table by neighbor discovery, the broker sends to its neighbors those items in its topology table that have changed since the prior propagation period. A sequence number is associated with each such update.

- **Step 3: Broker Network Topology Update.** When a broker receives updated information from its neighbor, it compares the sequence number of the incoming message with its topology table's corresponding items, replaces old items with new ones, and marks the items changed if the incoming items have a higher sequence number.

- **Step 4: Broker Routing Table Rebuilding.** A broker's topology table T changes either due to its own execution of the periodic neighbor update or due to the receipt of topology propagation from its neighbors. When such changes occur, the broker rebuilds its routing table by recalculating its shortest paths to other brokers henceforth uses the new routing table for delivering events.

3.4.2 Dynamic Neighbor Discovery in Wireless Ad Hoc Networks

The key points of dynamic neighbor discovery using expanding ring search can be summarized as follows.

- Each broker starts neighbor discovery by sending an ND message. This message is propagated to those nodes that are up to R hops away from the original node. R is current search radius starting from R=1, and R is less than the maximum radius of the discovery ring.
- When a broker B receives a ND message from A, it simply returns an ACK message to A.
- When A receives the ACK from B, it records node B as its neighbor along with the hop distance to broker B.
- When the number of neighbors reaches the upper limit or the search radius reaches the maximum radius, the broker stops the neighbor discovery process. Otherwise, increase the search radius by 1.
- If a node fails to find any neighbor within its maximum ring search radius, the neighbor discovery procedure continues until it finds at least one neighbor.

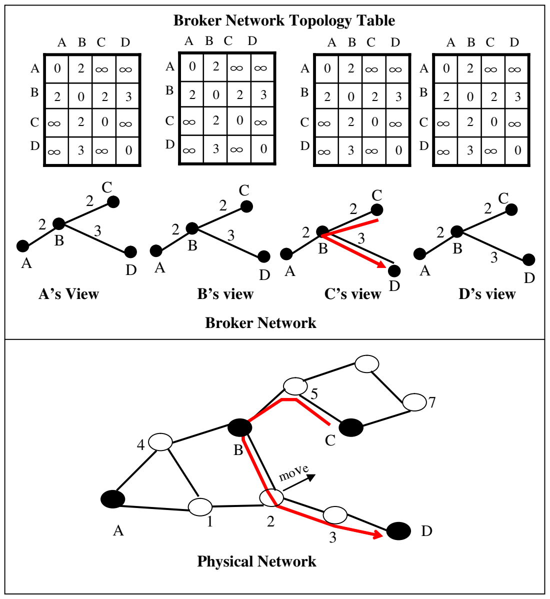
The neighbor discovery protocol used by opportunistic overlays differs from the PAST-DM protocol [40] in that each broker updates its neighbors completely independently from other brokers. Specifically, if broker A discovers that broker B is close enough to it, then A adds B as its neighbor without requiring B to do the same. Instead, A's update is propagated to B in the next topology propagation period. Furthermore, to control the overheads of broker network topology management, we impose some static limit on the maximum number of neighbors for each broker. In the prototype implementation, to avoid a storm of link state exchanges, each node can make the interval between two consecutive exchanges as the period plus a small random offset. The update period can be specified by each broker in a policy file discussed in Section 4.6.

3.4.3 An Example of Dynamic Broker Network Construction

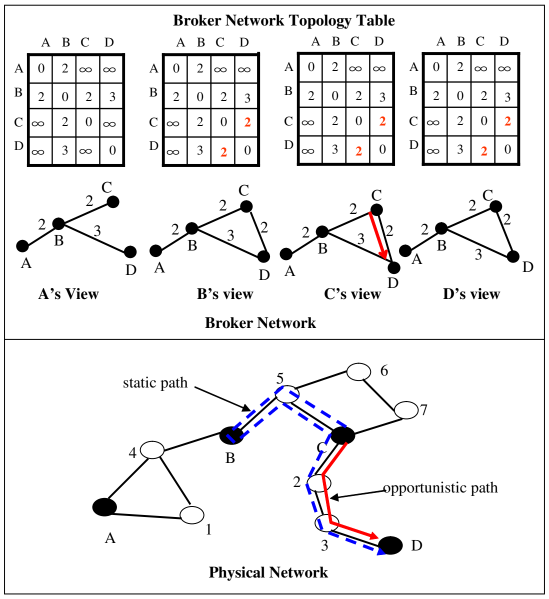
Figure 3.5 depicts an example. The broker network topology table is the adjacency matrix representation of the broker network graph. The number associated with a broker link indicates the corresponding physical network path length in terms of number of wireless links. At the beginning (Figure 5(a)), all four brokers (A, B, C and D) have the same view of the global broker network topology A—B, B—C and B—D. At some point, Node 2 moves away from B and 1, and closer to C. As a result, two old wireless network links 2—B and 2—1 are removed, and one new wireless link 2—C is created. Let's assume C is the first to start its update period (Figure 5(b)). C adds D as its new neighbor. Then C updates it topology table accordingly and propagates the change to its neighboring brokers B and D. After receiving updated information from C, each of B and D updates its topology table by adding the broker link C—D. B, C and D rebuild their overlay routing tables based on the new broker network topology A—B, B—C, B—D and C—D. As a result of C's routing table update, opportunistic overlays deliver events from C to D using the wireless paths C→2→3→D, compared with the static broker approach's C→5→B→5→C→2→3→D. Let's assume B is the next to start its update period (Figure 5(c)). B removes D from its neighbor list and updates its broker network topology knowledge accordingly. B then sends its changes since previous period to its neighbors A and C. Upon receiving updated information from B, both A and C update their topology knowledge by removing the broker link B—D. A also adds the broker link C—D to its topology table. At this time, each of A, B and C has the latest broker network topology knowledge. D's topology knowledge is outdated, and D's topology table will be updated either through C's propagation or via D's own running of the neighbor updates protocol, whichever comes first.

3.5 Dynamic Home Broker Change

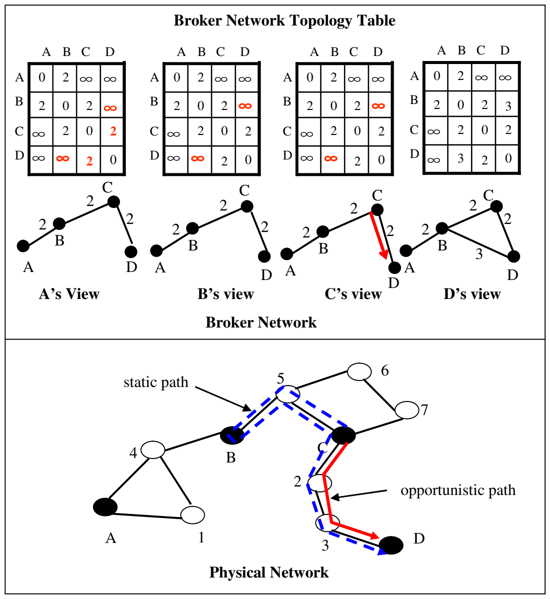
End-to-end latency not only depends on the network distance between a producer's home broker and a consumer's home broker, but also on the distance between producers/consumers and their home brokers. By dynamically constructing a broker network, we aim to optimize the former. By dynamically changing home brokers, we improve the latter.



(a) Initial Topology



(b) Topology after Node 2's Movement(after B's update)



(c) Topology after Node 2's Movement(after C's update)

Figure 3.5: An Example of Dynamic Broker Network Construction in Mobile Ad Hoc Networks

Toward these ends, opportunistic overlays act as described next.

When a client subscribes to a broker network for the first time, it must connect to some home broker that receives events (via the broker network) on behalf of the client and delivers received events to the client via some wireless network link. Intuitively, we should choose the nearest broker as the client's home broker, thereby optimizing the delay between the client and its home broker. In infrastructure-based mobile environments, when a mobile client moves out of range of its previous network and into range of new one, a new home broker need to be chosen. In ad-hoc mobile environments, both brokers and clients changes their locations frequently, therefore, home brokers must be chosen repeatedly, whenever nodes substantially change their locations. The procedure used by opportunistic overlays may be described as follows. Each client, periodically or in response to changes indicated by the underlying physical network protocol, executes a protocol that searches for the broker nearest its current location. If the nearest broker is not its home broker, it notifies the current home broker of its discovery. Upon receiving this news from its client, the home broker selects new home broker based on average path length between producers and the client and the CPU load of candidate brokers. If the home broker must be changed, the modulator relocation protocol is performed.

An interesting aspect of our approach is overload control, which is particularly important because end-to-end event delay from a producer to a consumer depends not only on the length of the network path, but also on event processing times at brokers. Processing times are determined by how fast modulators can be executed on home brokers which in turn depends on the home brokers' loads and capabilities. With clients changing locations, broker loads are subject to substantial runtime variation. One reason would be the sudden arrival of large numbers of local users, exemplified by many mobile units converging at a meeting. Another reason is the use of complex modulators by 'thin' clients, such as modulators that implement the flexible data transcoding required by such clients [116, 108]. In fact, the processing time of a modulator on moderately to highly loaded brokers can exceed network delays by an order of magnitude.

3.5.1 Dynamic Home Broker Change Protocol

The protocol followed to change home brokers can be summarized as follows.

- **Step 1: Nearest Broker Search.** Each client searches the broker nearest its location periodically using the nearest neighbor discovery algorithms. When a client finds the nearest broker that is not its current home broker, it shares with its current home broker the newly discovered broker along with its distance to that broker.

- **Step 2: Home Broker Selection.** When a broker receives a nearest broker discovery message from its client, it executes the following home broker selection algorithm.

  1. Check current home broker's (its own) and the discovered broker's CPU loads.
  2. If only one of the brokers CPU load exceeds the specified threshold, the other broker is selected as the client's new home broker.
  3. If both brokers are overloaded, the broker with the lighter load will be used as the new home broker.
  4. If neither broker is overloaded, calculate the average path from each producer to the client through each of two candidate brokers; then select the broker on the shorter path.
  5. If the home broker must be changed, perform the modulator relocation protocol.

- **Step 3: Horizontal Modulator Relocation.** The horizontal relocation operation is used to migrate the client's modulator from its current home broker to the selected broker, asks all producers' home brokers to compute paths to the new home broker, and switches event delivery from the old to the new paths.

3.5.2 Nearest Broker Search in Infrastructure-based Mobile Environments

The simplest approach to find a nearest broker is to probe each broker by the mobile client and choose the broker with lowest latency. If the number of brokers is too large, this method may become costly and impractical. If network topology is known, a cost-effective approach as described in [69] could be applied. The mobile client narrows down the set of potential candidate broker according to network topology. Then, the client or the proxy actively probes the set of candidate brokers and chooses the broker with the lowest latency. [89] proposes an approach which put a node into a "bin" by measuring its distance to a set of landmark nodes and then using the latencies to the landmarks represents the "bin" the node belongs. The nodes in the same bin have closer inter-node distance. Using the approach, instead of probing all brokers, the mobile client can only probe those brokers in its own bin or similar bin. Another similar approach is to compute an absolute coordinate (Global Network Positioning [70]) of mobile client's location based on its latencies to the landmarks. Then, its distance to each broker can be estimated by simply computing the Cartesian distance between their coordinates [70]. The number of landmarks (10-15) is usually much less than the number of brokers. The above procedures may be performed by a proxy residing on the fixed network (i.e., the base-station which the mobile client is connecting to) on behalf of the mobile client.

3.5.3 Nearest Broker Search in Mobile Ad Hoc Networks

The nearest broker search in mobile ad hoc networks can use the similar expanding ring search algorithm used in broker neighbor discovery, as described in Section 3.4.2. The idea is summarized as follows.

- A mobile client starts nearest broker search by sending BROKER-REQ message. The message will be propagated to those broker which is R hops away from the client. R is current search radius, and R is less than the maximum radius of the ring.
- When a broker B receives a BROKER-REQ message from a mobile client, it sends back to the mobile client an BROKER-REP message along with the distance between them.
- When a client receives a BROKER-REP messages for the first time, it records the broker as the nearest broker and stops the discovery procedure. Otherwise, it increases its search radius R by 1 and repeats the above procedure.

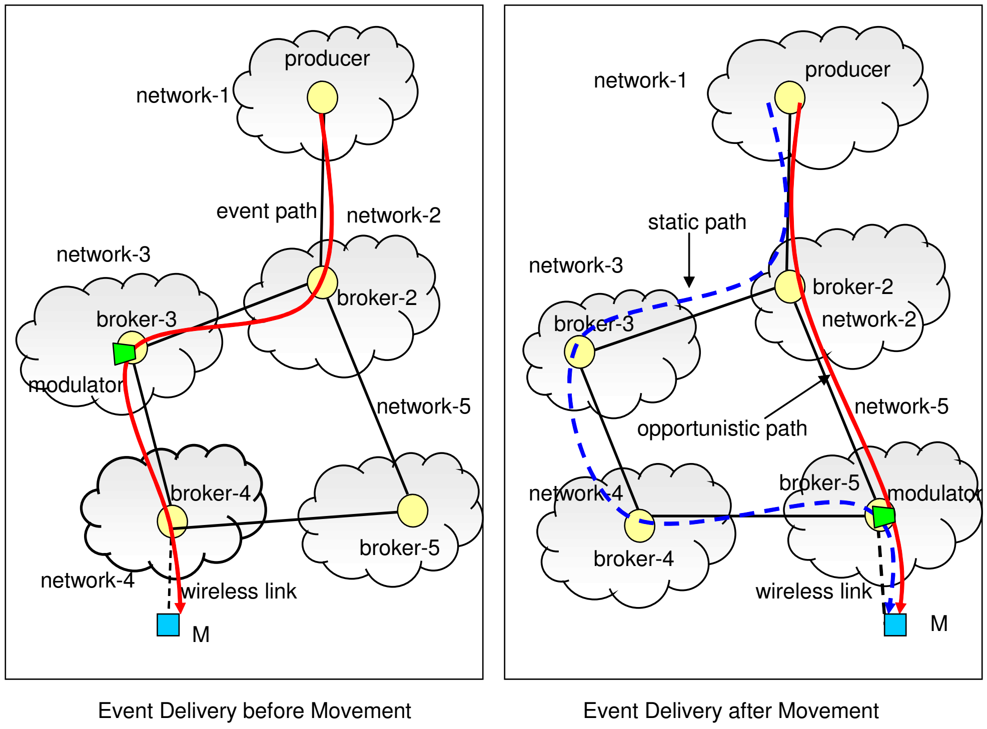


Figure 3.6: An Example of Dynamic Home Broker Change in an Infrastructure-based Mobile System

3.5.4 Examples of Dynamic Home Broker Change

An example of event system in an infrastructure-based mobile environment is illustrated in Figure 3.6. M is a mobile host which originally receives events from a producer through path producer→broker-2→broker-3→broker-4→M, corresponding to physical network path network-1→network-2→network-3→network-4→M. Broker-4 is M's home broker. M's modulator is also placed at its home broker and executes there. At some point in time, M moved from network-4 to network-5. The opportunistic overlay will automatically changes M's home broker from broker-4 to broker-5 and then the event delivery path becomes network-1(producer)→network-2(broker-2)→network-5(broker-5)→M. The new delivery path is shorter than the path of network-1(producer)→network-2(broker-2) →network-3(broker-3)→network-4(broker-4)→network-5→M, if M still used broker broker-4 as its home broker.

Figure 3.7 shows another example of home broker change in a mobile ad hoc network. A consumer that originally receives events from broker C via the path producer → A → C →

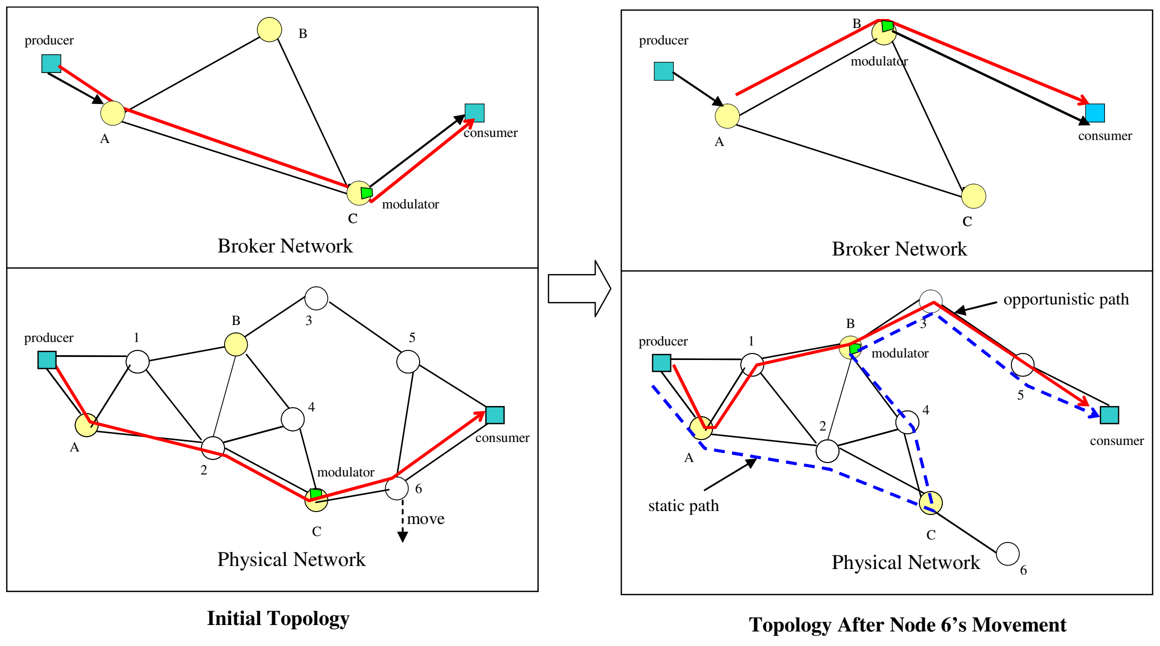


Figure 3.7: An Example of Dynamic Home Broker Change in Mobile Ad Hoc Networks

consumer corresponding to the physical network path producer→A→2→C→6→consumer. Broker C is the consumer's home broker. The consumer's modulator is also placed at C and executes there. At some point in time, with node 6 moving away from the consumer and node 5, during the consumer's nearest broker discovery period, it discovers that it is closer to B than C. When the consumer detects this, it sends a home broker change request along with B's information to C. C will choose B as the consumer's new home broker, since A→B→consumer is shorter than A→C→consumer under the new network topology. C then performs the modulator relocation protocol. After the change, the event delivery path from the producer to the consumer becomes producer→A→1→B→3→5→consumer. In contrast, without changing home brokers, the old path producer→A→C→consumer corresponds to the physical path producer→A→2→C→4→B→3→5→consumer, which is 2 hops longer than the path used by opportunistic overlays.

3.6 Dynamic Load Balancing

As discussed in the last section, end-to-end latency not only depends on network distance, but also on event processing time. With clients joining, leaving and changing locations, broker loads are subject to runtime variation, including overloads caused by home broker change. A modulator running on a heavily loaded broker will have a larger execution time, resulting in worse end-to-end latency. Two mechanisms are used in the opportunistic approach to control overloads. The first mechanism is to avoid selecting overloaded brokers as home brokers in dynamic home broke change protocol, as described in Section 3.5. The other one is to periodically perform a dynamic load balancing protocol. The load balancing protocol dynamically distributes modulators from heavily loaded brokers to lightly loaded ones.

3.6.1 Local Load Balancing Protocol

We first describe a load balancing protocol, which performs load balancing among brokers on the shortest path. Each broker checks its load periodically and performs the following protocol if it has been overloaded.

1. For each mobile client it's currently serving, the broker finds the least loaded broker on current event path to the client.

2. If the least loaded broker on current path is not overloaded, a modulator relocation operation is performed to migrate the modulator to the least loaded broker via either upstream or downstream operation.

3. If all brokers on the current path are overloaded, it simply moves the modulator to the least loaded broker found in the first step.

4. Repeat the above steps until either the broker is not overloaded anymore or the broker itself becomes the least loaded broker.

3.6.2 Global Load Balancing Protocol

Our evaluation in chapter 5 shows that for a broker network with a large number of brokers, this local load balancing algorithm works well since the average number of brokers on a path is relatively large. For more general broker network configurations, an advanced global load balancing algorithm is proposed. The global algorithm distributes modulators among brokers on the shortest path first; if all brokers on the shortest path are overloaded, it will try to find another feasible path with relatively longer network distance but having some less heavily loaded brokers. The global load balancing protocol can be summarized as follows.

1. For each mobile client it's currently serving, the broker finds out the least loaded broker on the shortest event path to the client.

2. If the least loaded broker on the shortest path is not overloaded, a modulator relocation operation is performed to migrate the modulator to the least loaded broker. If the broker itself is on the shortest path, the relocation is an upstream or downstream operation. Otherwise, it's a horizontal relocation.

3. If all brokers on the shortest path are overloaded and the current event path is not the shortest path, the broker searches the least loaded broker on the current path.

4. If the least loaded broker on current path is not overloaded, a modulator relocation operation is performed to move the modulator to the least loaded broker via either upstream or downstream operation.

5. If all brokers on shortest path and current path are overloaded, the broker searches an under loaded broker not on the shortest path and not on the current path in the order of the distances to itself. If such a broker is found, a horizontal relocation operation is performed to relocate the modulator to new found broker. The resulting event path consists of the shortest path from a producer's home broker to the new found broker plus the shortest path from the new found broker to the consumer's home broker,

6. If all brokers in the system are overloaded, it simply migrates the modulator to the least loaded broker on current path found in the third step.

7. Repeat the above steps until either the broker is not overloaded or the broker itself becomes the least loaded broker.

The above protocol always tries to balance broker loads among brokers on the shortest delivery path first. Only when all brokers on the shortest path become overloaded, it uses

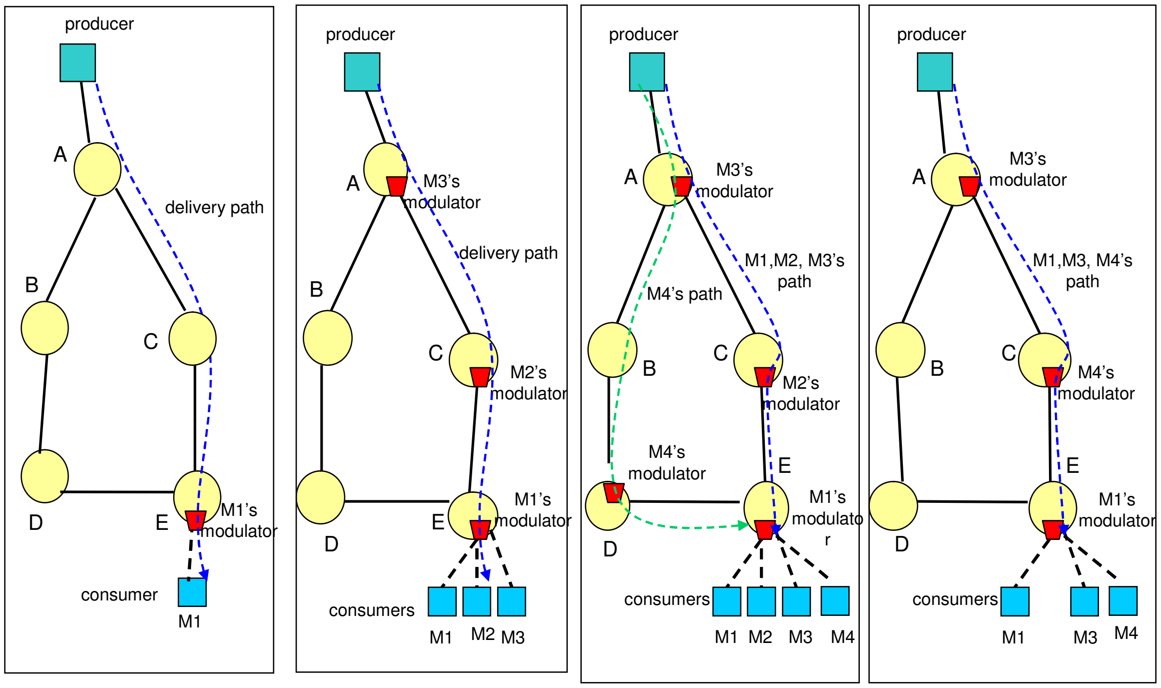


Figure 3.8: An Example of Dynamic Broker Load Balancing

brokers on relatively longer paths. Since the protocol always checks brokers on the shortest path first when it's looking for a target broker to offload modulator processing, the shortest path will be used whenever any broker on the shortest path becomes available.

3.6.3 An Example of Broker Load Balancing

Figure 3.8 shows an example of how global load balancing dynamically places modulators and changes event delivery paths. There are 5 brokers (A,B,C,D and E) in this network. A producer sends event via broker A. Mobile consumers receive event via E and E is the home broker. We assume each broker will reach its maximum computation load after executing one modulator. Initially, only one mobile consumer M1 connects to the broker network and its modulator executes at E. The events are routed using the shortest path A→C→E→M1. Then two more mobile consumers (M2 and M3) join the event system from the same location and use E as their home brokers. E is overloaded with running more than one modulator. Dynamic load balancing protocol moves M2's modulator to C and M3's modulator to A. All of three mobile consumers currently receive events through the shortest path A→C→E. At some point, the fourth mobile consumer (M4) joins the broker network.

Since all brokers (A,C and E) on the shortest path have been already overloaded, broker D is used to run M4's modulator. Though the resulting event path A→B→D→E→M4 is longer than A→C→E→M4, with running M4's modulator on a less loaded broker D, the new path can achieve better end-to-end delay of event delivery. Later, when M2 leaves the location and B becomes available, M4's modulator is migrated to B and the shortest path is used to deliver events to M4.

3.7 Summary of Adaptation Protocols

The proposed protocols in this chapter are totally distributed executed by each broker/client without any central control. The resulting advantages include increased scalability. As demonstrated by examples, these protocols are simple but powerful enough to support a variety of opportunistic adaptations. Moreover, the protocols do not assume any specific broker network topology and do not rely on any specific properties of the underlying physical network so that they can operate on top of different broker network topologies and across heterogeneous systems using multiple underlying routing protocols (e.g., Mobile IP, DSR, DSDV, or AODV [76, 56, 77, 78]). Finally, as shown in Chapter 5 and Chapter 6, the protocols are inexpensive and have controllable overheads.

The main limitation of the current protocols is that they do not handle network partition. There are multiple solutions to this problem. One such solution is described next. Consider a broker that discovers that the next broker or the client on the event delivery path is not reachable, using for example, the network partition detection algorithm described in [41]. If the destination is still reachable in some way, then one solution is to simply compute an alternative path to the destination. The result is that opportunistic overlays continue to operate correctly within the same partition. For those destinations that cannot be reached, other solutions have to be found. Our current thoughts are to store undeliverable events for later delivery and/or ask applications to provide functions that implement event discard. For example, for emergency-and-rescue applications, the desirable semantics is buffer and redelivery since information could be a lifesaver and must therefore not be lost. When an unreachable broker becomes reachable during networking merge later, the buffered events for that broker will be sent to it. The event ordering issues caused by dynamic routing and network partition can be handled at application-level using higher-level (e.g., JECho's) synchronization support [114]. We'd like to leave to future work topics like disconnection, reconnection, and related reliability issues.

CHAPTER IV

THE OPPORTUNISTIC OVERLAY FRAMEWORK AND IMPLEMENTATION

4.1 Overview of JECho

Opportunistic overlay networks are realized with the JECho distributed event system [116]. JECho implements a publish/subscribe communication paradigm, providing services to distributed, concurrently executing components via event channels. JECho's implementation is in pure Java, its group-cast communication layer is based on Java Sockets, and it runs with both standard and embedded JVMs. JECho's efficient implementation enables it to move events at rates higher than other Java-based event system implementations [116, 114]. The basic communication model in JECho is shown in Figure 4.1.

Using JECho's *modulators* [116, 114], individual event consumers can dynamically tailor event flows to their own needs, thereby adapting to runtime changes in component behaviors and needs and/or changes in platform resources. Modulators are implemented as Java objects, executed in a source's or broker's address space on behalf of clients. Client-defined customization with modulators, including event conversions or transformation, is performed prior to delivering the event to the consumer. Event conversion may reduce their sizes and hence reduce network traffic. Modulators can also be used for offloading computation

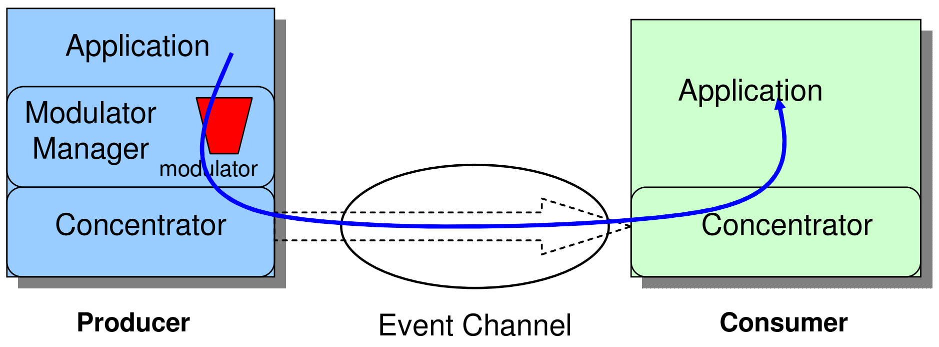


Figure 4.1: JECho Communication Model

from resource-constrained mobile devices. An example is a modulator that pre-converts events to the forms needed by some specific client's graphical displays, thereby eliminating the costs of data conversion in the client. In fact, it is sometimes impossible to display data appropriately without performing such conversions, as with rendering OpenGL-based graphical data on a PalmTop or when applying server-resident business rules to data prior to its display on cellphones [16]. Other reasons for such conversions include the delivery of data with certain Quality of Service or to conserve power on battery-limited handheld devices [85].

Two core components of JECho are the concentrator and the modulator manager.

- **Concentrator.** The concentrator is a hub for all incoming/outgoing events. Each Java virtual machine(JVM) has one concentrator. More details about concentrator can be found in [116, 114].

- **Modulator Manager.** The modulator manager provides an environment that permits the modulator to carry out computation with necessary access to local resources, and it provides facilities for adding, removing and changing modulator and to provide information about selected modulator state (e.g., modulator execution time).

4.2 Overview of Software Architecture

Opportunistic overlays are implemented as depicted in Figure 4.2. The architecture is able to support flexible adaptation implementations, and the architecture itself is easy to reconfigure and extend.

The basic component layer provides the lower level functionalities of resource monitoring and broker information management necessary for implementing different adaptations. Event-driven adaptations are implemented by defining a set of actions to react to specific events received from basic components. By using services provided by basic components, adaptation codes execute on high level protocols rather than needing to handle lower level details. The interaction between basic component layer and adaptation layer uses a set of consistent program interfaces and system events. As a result, it is easy for different brokers to define different adaptations based on their capabilities and requirements. In fact, the

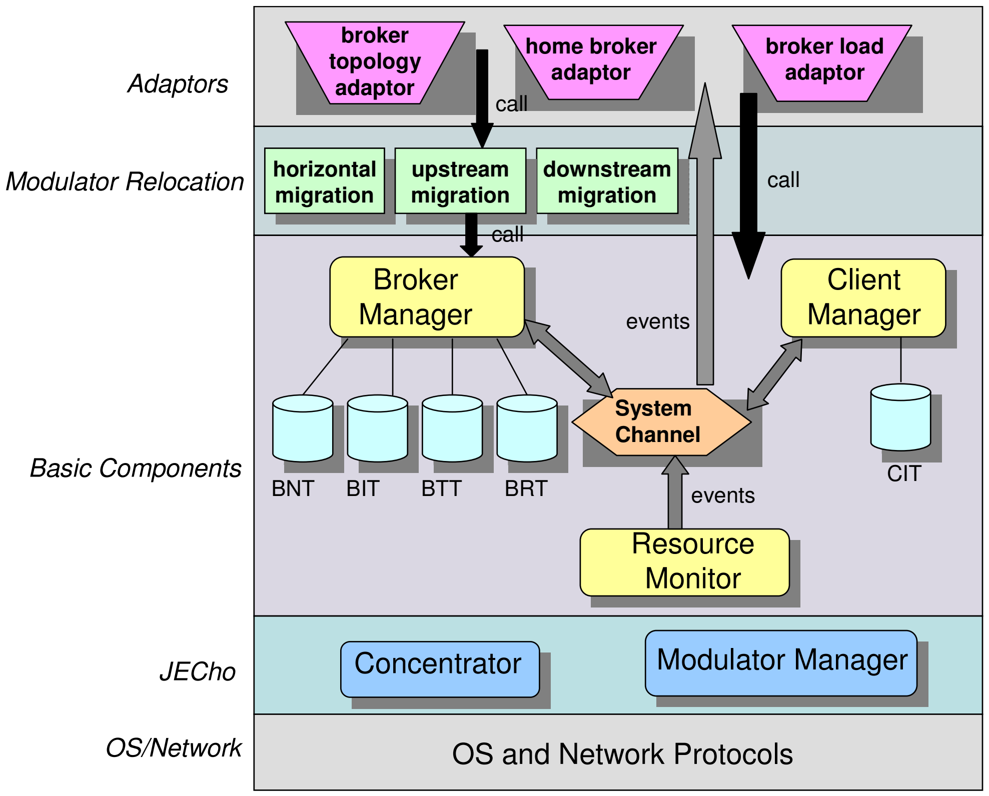


Figure 4.2: Opportunistic Overlays Software Architecture

implementation of an adaptation protocol within the current system is straightforward, as exemplified by the home broker adaptation that has less than 50 lines Java code. It is also easy to reconfigure and extend the system with new adaptations, such as those needed to handle physical network partition. Finally, a policy file can be associated with each broker or mobile client. The policy file is used to customize the parameters used by opportunistic overlays to meet the application's needs. The parameters include broker network topology, broker update period, nearest broker discovery period, load balancing protocols and modulator relocation semantics. For future work, we are considering adding a more advanced policy layer that permits user to define high-level policies concerning the adaptations being carried out, somewhat like the ideas in autonomic computing presented in [59].

4.3 Basic Component Layer

The basic component layer layer is composed of Resource Monitor, Broker Manager, and Client Manager. Components in this layer provide the core functionality implementing the adaptation protocols described in Chapter 3. Each component in this layer defines a set of program interfaces for other components to access its services (i.e., a set of 'get' and 'set' functions). Each component notifies other components and high level protocols by sending events containing the relevant information to an internal 'system' channel. Other components receive this information by registering their interests about certain events.

4.3.1 Resource Monitor

The Resource Monitor collects, aggregates, processes and delivers data about the current execution environment. There are two basic functions for each broker's Resource Monitor:

1. Collect local resource information, including CPU load, memory availability, and modulator execution time. CPU loads are extracted from the operating system, and the execution of each modulator is provided by the Modulator Manager. In infrastructure-based environments, communication costs to other brokers are measured by probing each other explicitly or using lower level network services [69, 109]. The current implementation measures the delays to its neighbors by periodically pinging them. In wireless ad hoc networks, communications costs are collected by querying the routing table maintained by the underlying network routing protocols or by periodically performing the expanding ring search protocol described in Section 3.4.

2. Notify other components of the resource information. Resource information is contained in events, other components and high level protocols can selectively register their interests in data being captured with the Resource Monitor.

4.3.2 Broker Manager

The Broker Manager maintains four tables, which are the Broker Neighbor Table (BNT), the Broker Information Table (BIT), the Broker Network Topology Table (BTT), and the Broker Routing Table (BRT). The Broker Manager provides a set of program interfaces to higher level protocols, including functions for accessing and changing broker-related information, and operations that propagate its broker network topology to neighboring brokers. The Broker Manager is also responsible for sending notifications to higher level components when it receives them.

- **Broker Neighbor Table.** The BNT table stores the information about its neighboring brokers, including their names and the up-to-date network latency to each of them. In fixed network environments, the BNT table is created at the time broker network is constructed, and latency information in the BNT is updated every time the Broker Manager receives latency update information from the Resource Monitor. In wireless ad hoc networks, the BNT is dynamically constructed using the neighbor discovery protocol discussed in Section 3.4.2.

- **Broker Information Table.** The BIT maintains information about each broker, including its name, IP address, current physical location, CPU and memory loads. The Broker Manager updates its BIT through information collected by the Resource Monitor, and it periodically propagates its BIT to neighboring brokers, which then adjust their own BITs.

- **Broker Network Topology Table.** Broker network topology knowledge, including connectivity among brokers and network distances for broker links, is contained in the BTT.

  For infrastructure-based systems, the BTT table is created at the time the event system is deployed and remains fixed. Our prototype implements four different broker network topologies in fixed network, including full mesh, adjacent connection, long-short regular mesh, and random regular mesh.

  In mobile ad hoc networks, the broker network topology is dynamically constructed and maintained by high level's dynamic broker network construction protocol as described in Section 3.4. The Broker Manager itself does not directly change the BTT. Instead, it sends notifications to higher level components when it receives broker network topology propagations from its neighbors.

- **Broker Routing Table.** The last table managed by the Broker Manager is the BRT, which contains shortest paths to other brokers. The BRT is computed based on the broker network topology stored in BTT. Every time the BTT changes, the BRT is recalculated. High level components can access the shortest path information in the BRT using the interface provided by the Broker Manager.

4.3.3 Client Manager

The Client Manager maintains information about each client for which the broker is currently acting as home broker, including its name, IP address, physical location, as well as related path information (e.g., current routing path and communication overhead of the path). This information is stored in the Client Information Table (CIT). The data in the CIT is collected from mobile clients. When the Client Manager receives a nearest broker discovery message from a client, it sends that message to the higher level adaptation protocol.

4.4 Modulator Relocation Layer

On top of the basic component layer are three relocation operations: horizontal relocation, upstream relocation, and downstream relocation. Relocation operations perform the task of relocating a client's modulator from current broker to another broker, and changing event delivery path accordingly, as described in Section 3.2. Relocation operations provide necessary functionality for supporting home broker change and dynamic load balancing, and will be called by those adaptations. The relocations are implemented by calling functions provided from the basic component layer. The prototype provides several variations of implementations for each operation. These different implementations vary in relocation semantics from strictly guaranteeing both event consistency and modulator state consistency to guaranteeing neither. Different applications can specify their desired semantics when they subscribe to an event system in the policy files discussed in Section 4.6.

4.5 Adaptation Protocol Layer

The adaptation protocol layer implements a variety of protocols, including 'dynamic broker network construction', 'dynamic home broker change' and 'dynamic load balancing'. Each such protocol is implemented with a Java object called an *adaptor*. An adaptor can register with the system event channel by specifying its interests in certain events delivered by the Resource Monitor, Broker Manager, and Client Manager. For the broker network topology adaptor, interesting events are a time event and a broker propagation event. The interesting event for the home broker adaptor is a broker discovery message received from a client. The code in the adaptor is implemented in the event handler method “process( )”, which is invoked whenever an interesting event is received. This code implements changes, such as reconfiguring the broker network, changing the home broker, or rebuilding a routing table. Using adaptors and the services provided by basic components, system developers can create potentially complex adaptation policies. Our prototype implementation has three types of adaptors: broker network topology construction adaptor, and home broker change adaptor and broker load balancing adaptor. Each adaptor performs the task for which it is named.

4.5.1 Home Broker Adaptor

The home broker adaptor is activated when a broker receives a broker discovery message from its client. It then evaluates the network path to new broker and determines whether or not to change the home broker based on path length and the potential new broker’s CPU load. If a change is necessary, the modulator relocation protocols is called. Network path length information is collected by sending queries to both the Broker and Client Managers. The skeleton code of the home broker adaptor is shown in Figure 4.3. For simplicity, this code fragment assumes that there is only one producer. It is apparent that the implementation is a straightforward Java realization of the high-level protocol described in Section 3.5.

4.5.2 Broker Network Topology Adaptor

The broker network topology adaptor implements dynamic broker network topology construction protocol described in Section 3.4. This adaptor periodically adjusts broker network topology based on updated neighbors information in its broker neighbor table and then sends updated broker network topology to its neighbors by calling broker manager’s propagation operation. When a broker receives a propagation event from its neighbor, the adaptor replaces outdated items in broker network topology table with new items received

```
public class HomeBrokerAdaptor implements BrokerAdaptor {

  //subscrite to receive nearest broker discovery message
  public void subscribe( ) {
    registerToEvents(NEAREST_BROKER_DISCOVERY);
  }

  //adaptation code
  public void process(Object e) {

        BrokerDiscovery bd = (BrokerDiscovery)e.getContent( );
        client = bd.clientName;
        candidateBroker = bd.broker;
        delay = bd.delay;

        //control overload
        load1 = BrokerManager.getLoad(this);
        load2 = BrokerManager.getLoad(candidateBroker);
        if(load1 < MAXIMUM_LOAD && load2 >= MAXIMUM_LOAD)
           newHomeBroker = currentHomeBroker;
        else if(load1 >= MAXIMUM_LOAD && load2 < MAXIMUM_LOAD)
            newHomeBroker = candidateBroker;
        else if(load2 >= MAXIMUM_LOAD && load2 >= MAXIMUM_LOAD) {
             if(load1 <= load2)
                newHomeBroker = currentHomeBroker;
             else
               newHomeBroker = candidateBroker;
        }

        //select home broker based on network delay
        else {
           sourceBroker = ClientManager.getSourceBroker(client);
           delay1 = ClientManager.getLatency(client);
           delay2 = BrokerManger.getPath(sourceBroker, candidateBroker) + delay;
           if(delay2 < delay1)
               newHomeBroker = candidateBroker;
           else
               newHomeBroker = currentHomeBroker;
       }

       //perform modulator relocation and path switch
       if(newHomeBroker != currentHomeBroker)
            RelocationAdaptor.migrate(currentHomeBroker,newHomeBroker,client);
  }
}
```

Figure 4.3: Skeleton Code for Home Broker Adaptor

```
public class BrokerNetworkTopologyAdaptor implements BrokerAdaptor {

  //subscrite to receive time event and broker topology update event
  public void subscribe( ) {
    registerToEvents(BROKER_UPDATE);
    registerToEvents(TIMER,period);
  }

  //adaptation code defined in event handler
  public void process(Object e) {

      //it's time to update neighbors and broker network topology
      if(e instanceof Timer) {
        //perform neighbor discovery
        BrokerManager.discoverNeighbors(radius,degree);

        BrokerNeighborTable bnt = BrokerManager.getBrokerNeighbors( );
        BrokerTopologyTable btt = BrokerManager.getBrokerTopology( );

        //update broker network topology table
        for(int i = 0; i < BROKERS; i++) {
             if(btt.getDistance(myid,i) != bnt.getDistance(i)) {
                 btt.setDistance(myid, i, bnt.getDistance(i));
                 btt.setChanged(myid,i) ;
                 btt.setSeq(myid,i, currentSeq);
             }
        }
        BrokerManager.setBrokerToplogy(btt);

        //update routing paths based on new broker topology
        BrokerManager.computeBrokerPath(bnt);

        //propogate the updated topology to neighbors
        BrokerManager.propagate(bnt,btt);
     }

     //received updated broker network topology from neighbors
     else if (e instance of BrokerUpdate) {
         BrokerTopologyTable btt1 = BrokerManager.getBrokerTopology( );
         BrokerTopologyTable btt2 = (BrokerUpdate)(e.getContent());

        //update broker network topology
        for(int i = 0; i < BROKERS; i++)
          for(int j = 0; j < BROKERS; j++) {
            //compare sequence number and replace the old item with the incoming one
            if(btt1.getSeq(i,j) < btt2.getSeq(i,j)) {
               btt1.setDistance(i,j, btt2.getDdistance(i,j));
               btt1.setSeq(i,j,btt2.getSeq(i,j));
               btt1.setChanged(i,j);
            }
            BrokerManager.setBrokerToplogy(btt1);

            //recalculate broker paths based on new topology
            BrokerManager.computeBrokerPath(btt1);
         }
     }
   }
}
```

Figure 4.4: Skeleton Code for Broker Network Topology Adaptor

```
public class BrokerLoadAdaptor implements BrokerAdaptor {
   //subscribe to receive time event
   public void subscribe( ) {
     registerToEvents(TIMER,period);
   }

   //adaptation code defined in event handler
   public void process(Object e) {
     load1 = BrokerManager.getLoad(this);

     //perform load balacning
     while(load1 > MAXIMUM_LOAD) {
       consumerList = ClientManager.getClients( );
       if(!consumerList.hasNext( ))
          break;

       //check each modulator running by the broker
       if(consumerList.hasNext( )) {
         client = consumerList.next( );

         //search the least loaded broker on shortest path to the client
         path = BrokerManager.getPath(client);
         brokerList1 = getBrokerList(path,TRUE);
         broker = findBroker(brokerList1,LEASTED_LOADED);
         load2 = BrokerManager.getLoad(broker);

         //relocate modulator to the leastest loaded broker
         if(load2 < MAXIMUM_LOAD) {
            Relocation.migrate(this,broker,client);
            status = SUCCESS;
         }

         //search other brokers if all brokers on the shortest path are overloaded
         else {
           //search the nearest broker not on the shortest path
           brokerList2 = getBrokerList(path,FALSE);
           broker = findBroker(brokerList2, NEAREST);
           load2 = BrokerManager.getLoad(broker);

           //perform modulator relocation and path switch
           if(load2 < MAXIMUM_LOAD) {
             Relocation.migrate(this,broker,client);
             status = SUCCESS;
           }
           else
             status = FAILURE;
         }

         //if all brokers in the system are overloaded
         //relocate the modulator to the least load broker on the shortest path
         if(status != SUCCESS) {
           broker = findBroker(brokerList1,LEASTED_LOADED);
           Relocation.migrate(this,broker,client);
         }
         load1 = BrokerManager.getLoad(this);
       }
     }
   }
}
```

Figure 4.5: Skeleton Code for Broker Load Adaptor

from the neighbor. This is implemented by associating each item in the BTT with a sequence number, comparing the sequence number of the incoming message with its topology table's corresponding items, and replacing old items with new ones if the incoming items have a higher sequence number. After updating the broker network topology table, it calls the broker manager's routing table rebuilding function to recalculate the routing path to each broker, using the new broker network topology table. The skeleton code for the broker network topology adaptor is shown in Figure 4.4.

4.5.3 Broker Load Adaptor

The broker load adaptor implements the load balancing protocol proposed in Section 3.6. It periodically checks its load and dynamically distributes its modulators to other less loaded brokers by calling modulator relocation operations when it's overloaded. Brokers on the shortest path are preferred as an offloading target. If all brokers on the shortest path are loaded, brokers on other paths are chosen. The load balancing protocol is performed by each home broker periodically, and the period and the overload threshold can be specified when a broker is deployed. Figure 4.5 shows the skeleton code of the broker load adaptor implementing the global load balancing algorithm.

The local load balancing protocol is also implemented in our prototype. That adaptor limits load balancing among brokers on the shortest path and never changes the delivery path.

4.6 Policy File

A policy file can be associated with each broker and mobile client. The policy file is used to customize the adaptation parameters used by opportunistic overlays to meet the application's needs. When a broker or client is deployed, each component reads the policy file and customizes its parameters according to the values specified in the policy file. We describe the policy file below. The detailed performance evaluation using different policies can be found in Chapter 5 and Chapter 6.

- **Broker Network Topology.** For fixed networks, the broker network topology can be specified as full mesh, short-long regular mesh, random regular mesh, and adjacent connection. This is a global parameter, which means all brokers must specify the same topology. For regular mesh, each broker can specify its own degree (number of neighbors). For a small scale system, a full mesh can achieve better performance since the broker path is optimal in this case. For a large system, short-long mesh and adjacent connection might be appropriate.

For mobile ad hoc networks, each broker can specify the maximum search radius and the maximum number of neighbors used in neighbor discovery in its policy file. There's a tradeoff between the maximum radius, maximum broker node degree, and the path length. With larger search radius and broker degree, the broker can have more neighbors and hence shorter broker paths to other brokers. However, more broker links also means more overhead for broker network update. In order to avoid network partition, the radius and node degree should not be too small. Our experience via simulation shows that a search radius of 3 and broker degree of 4 is good enough to achieve good performance as well as maintain network connectivity for a broker network consisting of 40 brokers on a 100-node physical network.

- **Broker Update Period.** The broker update period specifies the time interval used by each broker to send updates to its neighbors. More frequent updates lead to better up-to-date broker knowledge and better performance at the cost of additional bandwidth usage. Experimental results in our performance evaluation show that an update period of 60 seconds in both fixed and ad-hoc networks can achieve good performance with moderate costs.

- **Relocation Semantics.** Different relocation semantics can be specified by brokers. Maximum semantics guarantees both modulator state consistency and no lost events and no duplicated events. Other available options include modulator state consistency only, no lost event only, and minimum semantics with no guarantee. The cost-effective lightweight relocation operations may be more practical for applications in wireless ad hoc networks where both network and computing resources are scarce..

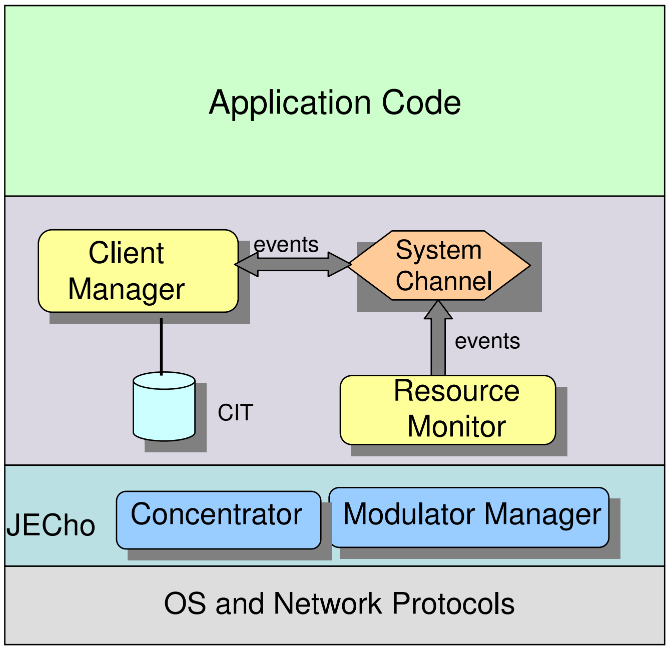


Figure 4.6: Client Software Architecture

- **Load Balancing.** The load balancing protocol options used by each broker include no balancing, local load balancing, and global load balancing. Also, the threshold of the maximum load of each broker can be specified. A zero threshold means that the broker is unable to execute any modulator, and its only task is to relay events. This provides flexibility for mobile applications in mobile ad hoc networks where mobile computing devices are highly heterogeneous.

- **Nearest Broker Discovery Period.** The nearest broker discovery period specifies how frequently each client performs the nearest broker discovery. The value is specified by each mobile client in its policy file. The more frequent nearest broker discovery can lead to better performance. As shown in our performance evaluation, a nearest broker discovery period of 100 second is good enough.

4.7 Client Component

The final element of opportunistic overlays are client-resident components that interact with the broker overlay. The client software architecture is shown in Figure 4.6. The components include a Resource Monitor and Client Manager. A Client's Resource Monitor performs a task simpler than its counterpart in a broker. It monitors the client's location and measures the distances to its home broker and other brokers. These items are reported to its Client Manager. Whenever the client location changes, the Client Manager notifies its home broker's Client Manager, and it also periodically executes the nearest broker discovery protocols in Section 3.5.3 and Section 3.5.2.

CHAPTER V

PERFORMANCE EVALUATION OF INFRASTRUCTURE-BASED MOBILE SYSTEMS

5.1 Simulation

For the purpose of simulation, 10 physical networks with 100 nodes each are generated using the BRITE's Flat-AS model (an Internet Topology Generator developed at Boston University [66]). Each node represents an autonomous system (site or domain) in the Internet. Each broker is randomly attached to one of the physical nodes. Unless stated otherwise, there is one event producer, which is randomly chosen from among 100 nodes, and one mobile consumer, which randomly travels through all nodes. Three broker network topologies are used in the simulation, including adjacent connection (AC), short-long regular mesh (Short-Long) and random regular mesh (Random). A short-long regular mesh is constructed with the following rules: each broker chooses 3 neighbors by picking 2 nearest brokers, and a remote broker randomly chosen from brokers at farther locations. A random regular mesh is constructed by each broker randomly choosing 4 brokers as its neighbors. We run each experiment on top of each physical network topology generated by BRITE, and the result reported is the average over 10 runs.

5.1.1 Average Latency in Simulation

First, we compare the average latency from the producer to the consumer across different broker networks with vs. without opportunistic overlay protocols. We vary the number of brokers from 10 to 100 and compute the average latency from the producer to the mobile consumer. Specifically, the latency from the producer to each location of the mobile consumer is computed and averaged over all locations. In order to establish a basis for comparison, we also measure the average latency from the producer to the consumer across physical network.

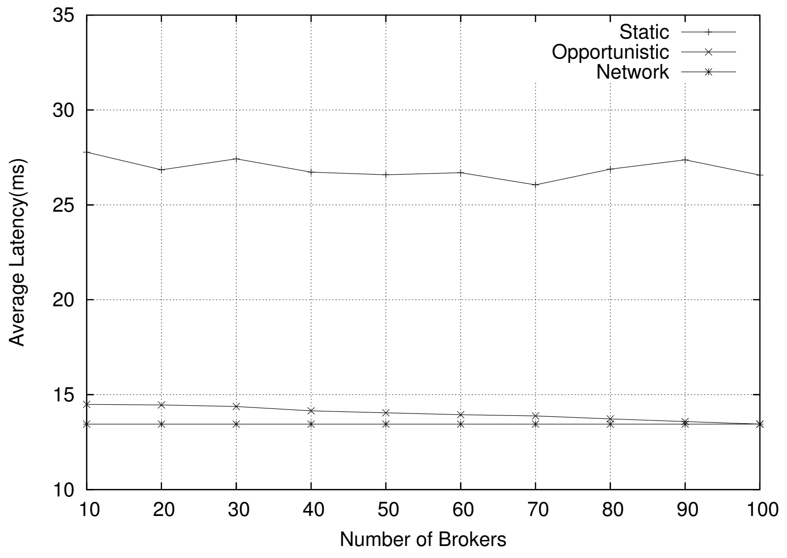


Figure 5.1: Average Latency between a Producer and a Mobile Consumer (Adjacent Connection)

The results with broker networks using adjacent connection (AC) are shown in Figure 5.1. As evident from the figure, opportunistic overlays improve performance significantly for all broker network sizes. Compared with the static approach's 26.5ms, the opportunistic approach has an average latency of only 14.5ms. Furthermore, the latency of the opportunistic approach is very close to network latency, 14.5ms vs. 13.4ms. This is due to the fact that the adjacent connection technique uses knowledge about physical network topology for broker overlay construction, and the resulting broker network topology is a good approximation of the physical network topology.

The results with the short-long broker mesh are depicted in Figure 5.2. They show that the opportunistic approach can deliver events much more efficiently than the static approach. The average latency of the opportunistic approach is less than one half of the static one. Compared with adjacent connection, the performance differential between the broker delivery and physical network is slightly larger. This is because the broker network using the adjacent connection can match the physical network better than the short-long mesh. However, the adjacent connection also leads to more routing overhead compared to the short-long mesh, as shown in Section 5.1.2.

Figure 5.3 reports the results with random mesh. Although the latency is obviously

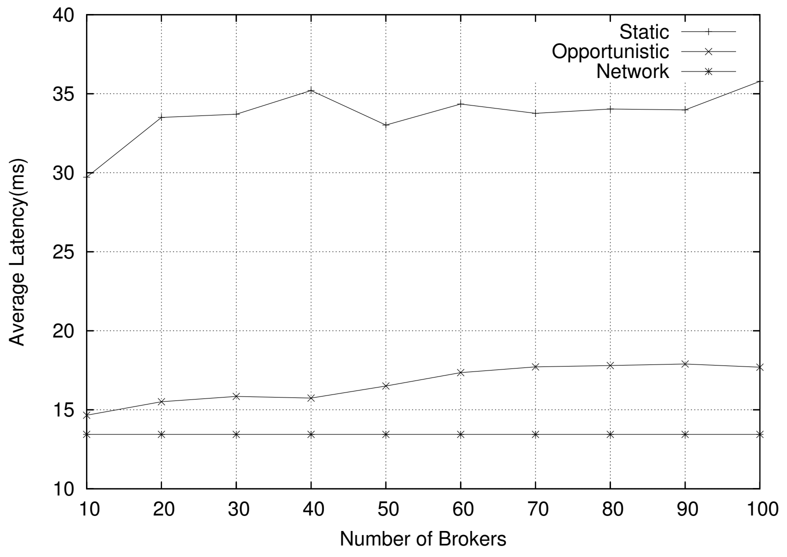


Figure 5.2: Average Latency between a Producer and a Mobile Consumer (Short-Long Regular Mesh)

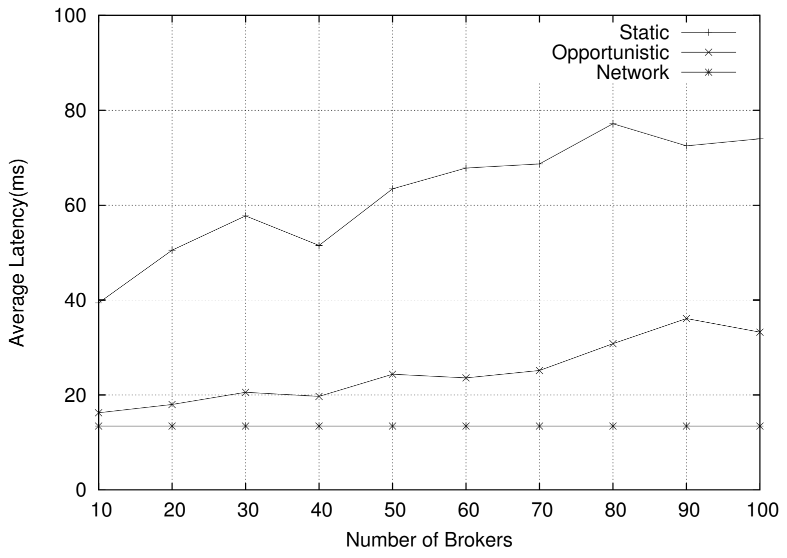


Figure 5.3: Average Latency between a Producer and a Mobile Consumer (Random Regular Mesh)

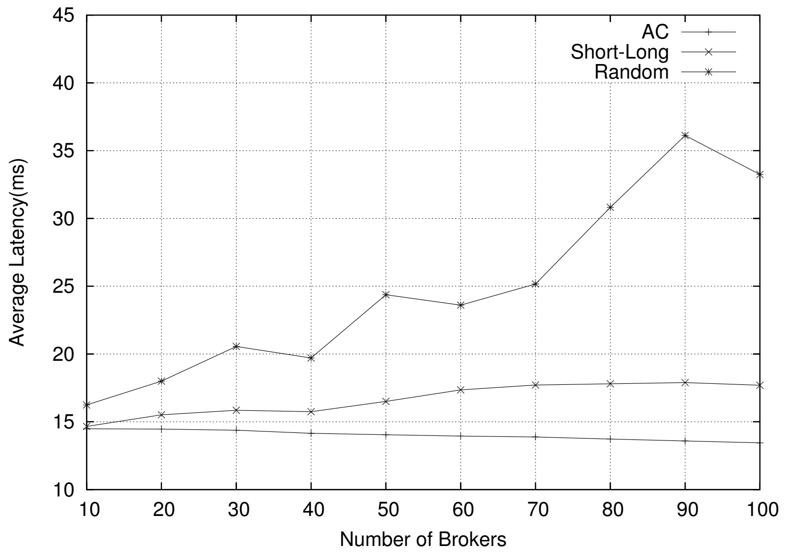


Figure 5.4: Comparison of Average Latency With Different Broker Network Topologies

worse than with adjacent connection and short-long mesh, the performance improvement from the opportunistic overlay approach is still substantial, 24.4ms vs. 63.5ms for the broker network of 50 brokers. With increased sizes of broker networks, the mismatch between random mesh broker network and the underlying physical network becomes worse. This degrades the performance of event deliver across broker network substantially, as shown in the figure.

The comparison of opportunistic delivery with three different broker network topologies is shown in Figure 5.4. Among the three broker network topologies, adjacent connection has the best performance and random mesh has the worst one. In addition, both adjacent and short-long topology are not sensitive to broker network size, while the random mesh's latency increases quickly with an increased number of brokers.

Figure 5.5 shows the average latency between 10 producers and a mobile consumer. The short-long mesh is used to construct broker network topology. In this experiment, a mobile consumer receives events from 10 producers each running at a different node randomly chosen from 100 physical nodes. The latency from each producer to the consumer is measured and averaged. Compared with the static approach's 30ms, the opportunistic approach has an average latency of 16ms.

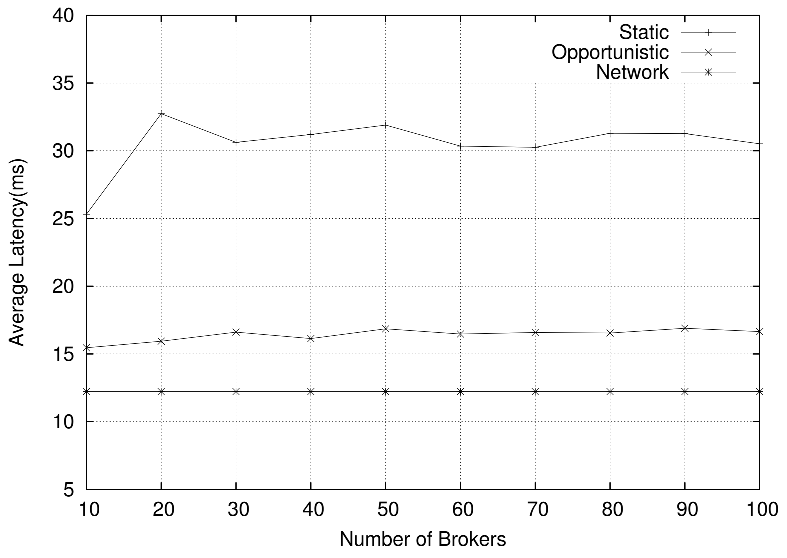


Figure 5.5: Average Latency between 10 Producers and a Mobile Consumer (Short-Long Mesh)

5.1.2 Broker Routing Overhead

This set of experiments evaluates routing overhead, which is computed as the average bandwidth each broker consumes for monitoring broker link performance and exchanging its broker link information with its neighbors. We vary the number of brokers from 10 to 100 and measure the bandwidth used by each broker.

The broker routing overheads using different broker network topologies with an update period of 30 seconds are shown in Figure 5.6. The adjacent connection achieves the best performance, as shown in Figure 5.4, but it also has the worst (though still moderate) routing overheads under most broker network sizes. Compared with the short-long mesh's 1Kbps bandwidth requirement with a broker network of 50 nodes, the adjacent connection's routing uses about 3Kbps.

The broker routing overheads on top of short-long mesh with update periods of 15 seconds, 30 seconds, and 60 seconds are shown in Figure 5.7. For the broker network with 50 nodes, an average of 1.9 Kbps bandwidth is used by each broker with an update period of 15 seconds. This constitutes a very small bandwidth usage in most modern network infrastructures.

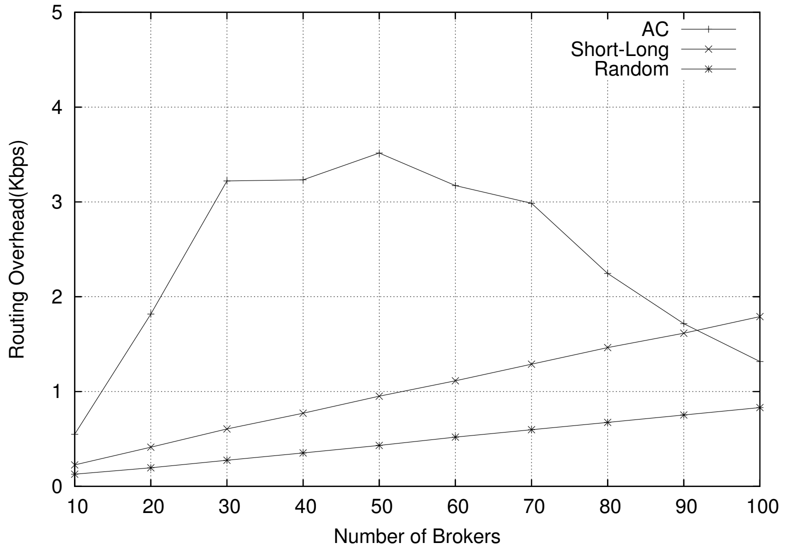


Figure 5.6: Broker Routing Overhead with Different Broker Network Topologies (30 Seconds Update)

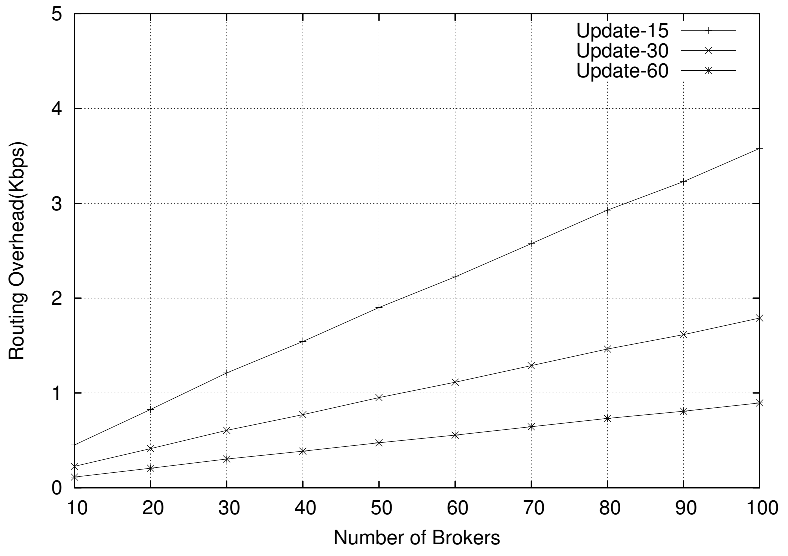


Figure 5.7: Broker Routing Overhead Using Different Update Periods (Short-Long Mesh)

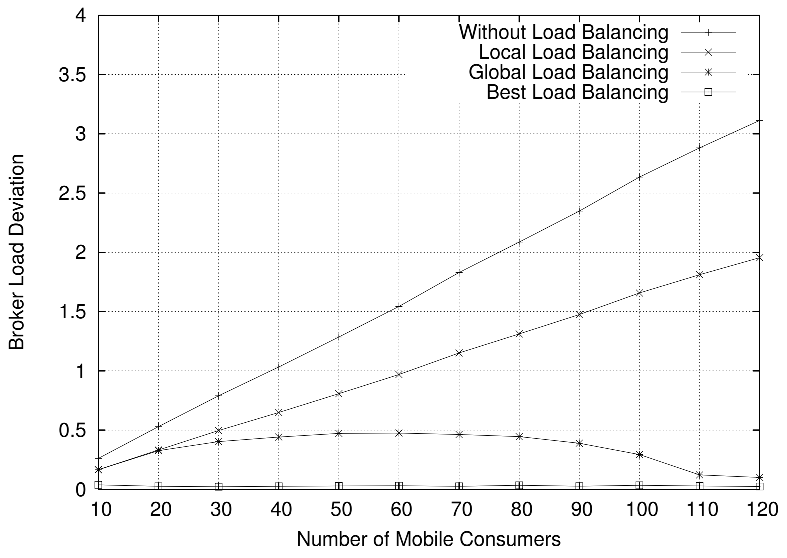


Figure 5.8: Broker Load Deviation versus Number of Mobile Consumers

5.1.3 Load Balancing

In order to evaluate the effects of load balancing, we conduct a set of experiments in which we vary broker loads by dynamically increasing the number of mobile consumers connecting to the event system from the same location. Four different approaches are evaluated in terms of the resulting broker load deviation and maximum broker loads: (1) *Without Load Balancing*, which always uses the closest broker as the mobile consumer's home broker; (2) *Local Load Balancing*, which performs dynamic load balancing among brokers on the shortest path between the producer and the consumers; (3) *Global Load Balancing*, which performs load balancing among brokers on the shortest path first, and then uses brokers on alternative paths when all brokers on the shortest path are overloaded; and (4) *Best Load Balancing*, which always chooses the broker with the minimum load without taking into account the network path length.

The broker load deviations and maximum broker loads for the varying number of mobile consumers are shown in Figure 5.8 and Figure 5.9 respectively. Improvements due to load balancing are significant. Without load balancing, broker loads become increasingly unbalanced with increased numbers of mobile clients. The load balancing algorithms ameliorate this problem by relocating modulators from overloaded brokers to lightly loaded brokers.

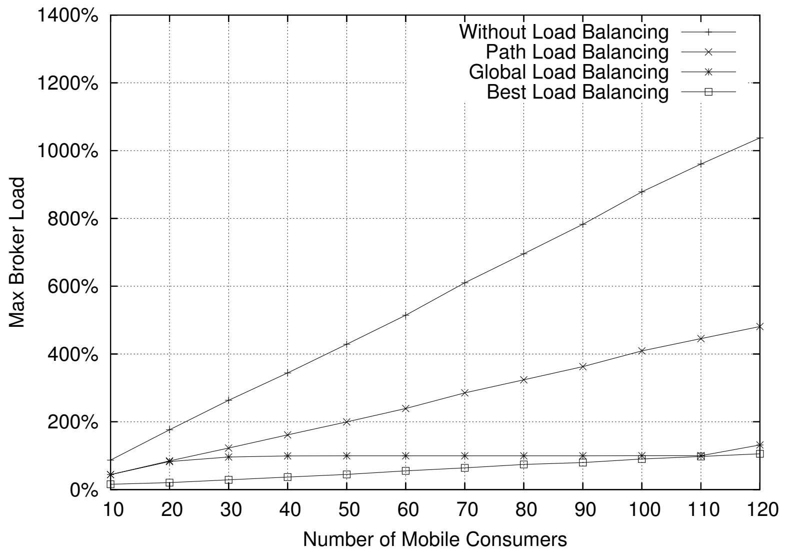


Figure 5.9: Max Broker Load versus Number of Mobile Consumers

Local load balancing and global load balancing approaches perform exactly the same and have the same performance when the load is light(i.e., the number of mobile consumers is less than 20). At this time, broker loads are distributed among brokers on the shortest path. When the number of mobile consumers continues to increase and all brokers on the shortest path become overloaded, the global approach starts optimizing broker loads by using less loaded brokers on relatively longer paths, and obviously performs better than local load balancing approach whose load balancing is limited to those brokers on the shortest path. Furthermore, these figures also show that the outcomes of global load balancing are comparable with the best load balancing approach, which optimizes broker load only and does not take into account path length between the producer and consumers.

5.2 PlanetLab Experiments

This set of experiments involves running a set of Java programs on the PlanetLab testbed [83]. Planetlab is a wide area testbed which provides the capacity of running actual distributed experiments. PlanetLab currently consists of 508 machines, hosted by 235 sites, spanning over 25 countries. All of the machines are connected to the Internet.

We randomly choose up to 30 nodes to run brokers from all PlanetLab machines with edu domain (hosted by universities in the United States). A broker network is constructed

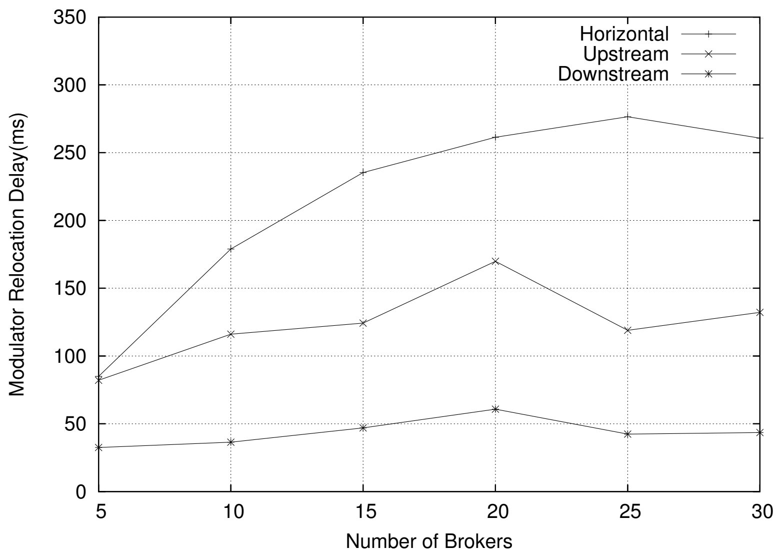


Figure 5.10: Time Required to Complete Modulator Relocation

using the same short-long mesh approach as the one used in our simulation. The resulting broker network has an average degree of 3.5. In this set of experiments, a mobile consumer roaming randomly among broker nodes receives events from a producer running on one of the nodes. The events being passed are arrays of 1000 floats, and a for() loop is used to emulate the running times of complex modulators. Mobility is simulated by dynamically killing the consumer program at a location and starting it at a new location. The latency from the producer to the consumer is computed as the latency from the producer to the mobile consumer's current location plus a wireless network latency. Wireless latency is achieved by measuring the delay of delivering the same events from planet1.cc.gt.atl.ga.us, one of the PlanetLab machines at Georgia Tech, to the same consumer program running on a laptop across Georgia Tech wireless campus network (LAWN).

5.2.1 Modulator Relocation Delay

This set of experiments evaluate the modulator relocation delay of three different relocation operations: horizontal relocation, downstream relocation, and upstream relocation. The mobile consumer moves randomly, and the elapsed real time required to complete each modulator relocation is measured at each location and averaged over all locations. For horizontal relocation, the time of completing a relocation from the consumer's home broker

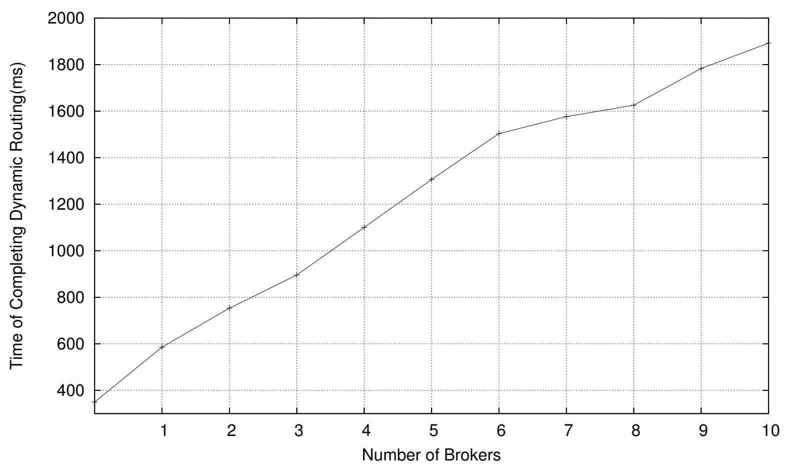


Figure 5.11: Time Required to Complete Dynamic Path Change

to the nearest broker is measured. For upstream and downstream relocations, the delay is measured as the time to complete a relocation between two randomly chosen brokers on current event path. The results in Figure 5.10 demonstrate that modulator relocation delays remain small even for a moderate number of brokers (e.g., 30). Among three relocation methods, horizontal relocation has the largest delay since its relocation protocol involves more brokers, requires more steps, and most importantly, the network distance between the source broker and destination broker could be large; the latter is because the mobile consumer can move across large network topologies. The downstream operation is the simplest one among the three relocation operations, and the delay to complete this relocation under all broker network sizes is less than 50ms.

Figure 5.11 shows the times required to complete a path change. The times shown here are the elapsed real times from when the home broker receives the consumer's location change message to when all producers start delivering events to the consumer via the new home broker. As shown in the figure, delay increase linearly with increases in the number of event producers.

5.2.2 Average Latency in PlanetLab

The average latency between the producer and the mobile consumer with different broker network sizes appears in Figure 5.12. The opportunistic approach delivers events much more

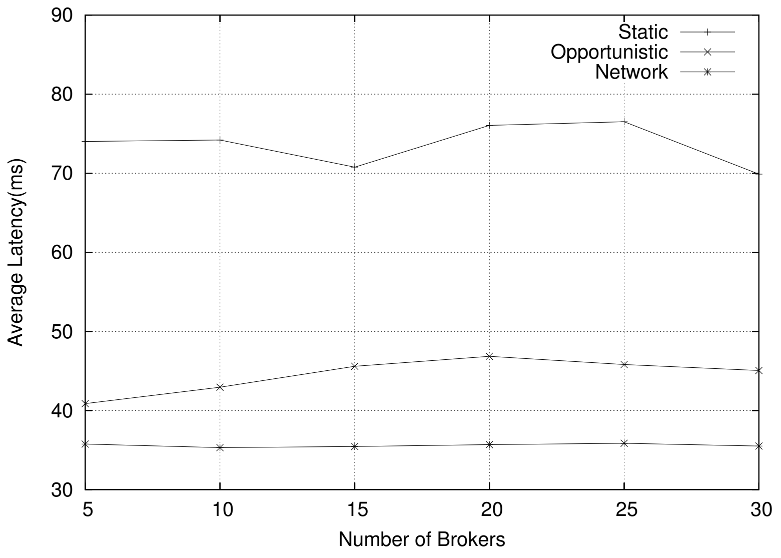


Figure 5.12: Average Latency between a Producer and a Consumer in PlanetLab

efficiently than the static one, 45ms vs. 72ms. As a comparison, the latency of delivering events via the physical network path is 35ms.

5.2.3 Latency with Load Balancing

In order to evaluate the effect of the opportunistic approach with load balancing management, we conduct a set of experiment in which we vary the average broker load and compare the latency from the producer to the mobile consumer using 4 different approaches: (1)*Static without Load Balancing*, which always uses the same home broker and doesn't do any load balancing; (2)*Opportunistic Without Load Balancing*, which performs dynamic home broker change, but doesn't perform broker overload control; (3)*Opportunistic with Local Load Balancing*, which perform the dynamic home broker change and local load balancing protocols; and (4)*Opportunistic with Global Load balancing*, which performs dynamic home broker change and uses global load balancing protocol.

Figure 5.13 depicts the results of event delivery latency versus average broker load. With increased broker loads, modulator execution times increase, and the latencies without load balancing increase rapidly. However, the dynamic approach still continues to outperform the static one, and the latency using the dynamic approach with load balancing increases very slowly with increased broker loads. Compared with the static approach's 110ms, the

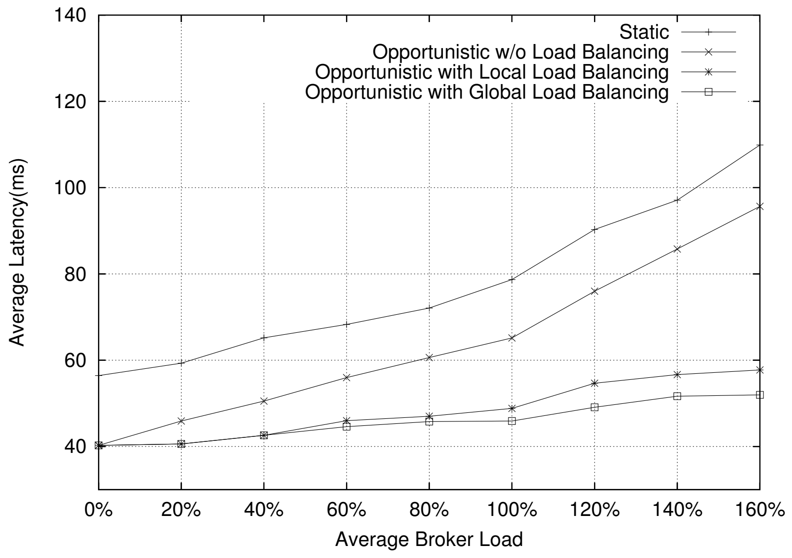


Figure 5.13: Average Latency versus Average Broker Load

opportunistic approach with global load balancing can deliver events in an average of 52ms when average broker load reaches 160%. Global load balancing performs better than local load balancing when average broker load is more than 40%. This is because the global approach can use brokers with lighter loads on relatively longer paths when all brokers on the shorter paths have been overloaded, resulting in decreased modulator execution times and more efficient event delivery.

5.3 Virtual Workbench Application

The last set of experiments demonstrates the practical utility of opportunistic overlays, by running a sample application on PlanetLab. The experiments being performed run a virtual design workbench developed in our previous work [16]. The workbench enables scientists to remotely interact with ongoing design simulations and with the software packages implementing those simulations. Here, timely delivery of events is essential due to the interactive nature of the application. Our experiment involves a target application component and a monitor (i.e., display) component. The application component runs several product design and analysis packages and sends intermediate results to an event channel. In our current implementation, all components used are 'wrapped' into objects that contain all relevant workbench experiment attributes and status information. The client component initiates certain experiments and observes the ongoing simulations by subscribing to the event channels. Intermediate results are typically quite large. However, at any one time, an engineer is typically interested in viewing some subset of these results. This is achieved by providing a modulator that analyzes intermediate results and transforms interesting ones into appropriate, viewable forms.

We envision a scenario in which scientists and engineers working in geographically different locations remotely execute large-scale product designs and simulations such as the Rapid Tooling TestBed (RTTB). While simulations are run on the higher end machines in the engineering design center, engineers wish to remotely monitor real-time outputs from their laptop or handheld devices. Similar scenarios occur in the oil industry, where engineers at drill sites wish to interact with large-scale subsurface simulations being run in analysis centers. In any case, intermediate results are sent to a broker close to the user's current physical location via the Internet and then forwarded to the monitor program running on his laptop via wireless network. Most such experiments run for a long time from several hours to several months. When the user changes his location, e.g., leaves for home from his office or attends a meeting at another city, he will wish to continue to monitor the ongoing experiments from his laptop at the new location.

The basic experiment configuration shown in Figure 5.14 uses a broker network consisting of 4 machines running at the University of Michigan (planetlab1.eecs.umich.edu), the University of Illinois at Urbana-Champaign (planetlab1.cs.uiuc.edu), Washington University in St. Louis (vn2.cs.wustl.edu), and Georgia Tech (plant1.cc.gt.atl.ga.us). [1] There are 5 broker links in the broker network, which are UMICH—UIUC, UMICH—GATECH, UIUC—WUSTL, UIUC—GATECH and WUSTL—GATECH. A RTTB experiment is running at UMICH, and a monitor program initially runs on a laptop close to WUSTL and will later move to GATECH.

In our experiment, we evaluate the performance of event delivery using multiple approaches: static, opportunistic without load balancing, and opportunistic with global load

[1] For simplicity, we refer to each broker with the name of the site where it resides (UMICH, UIUC,WUSTL and GATECH).

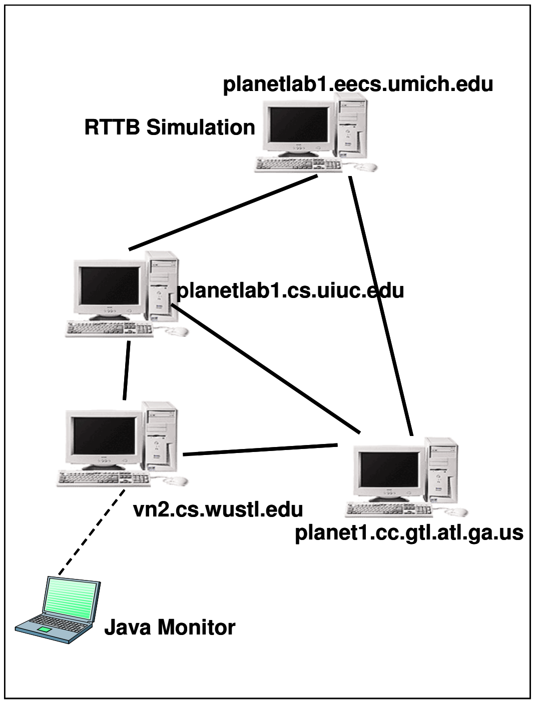


Figure 5.14: Basic Experiment Configuration

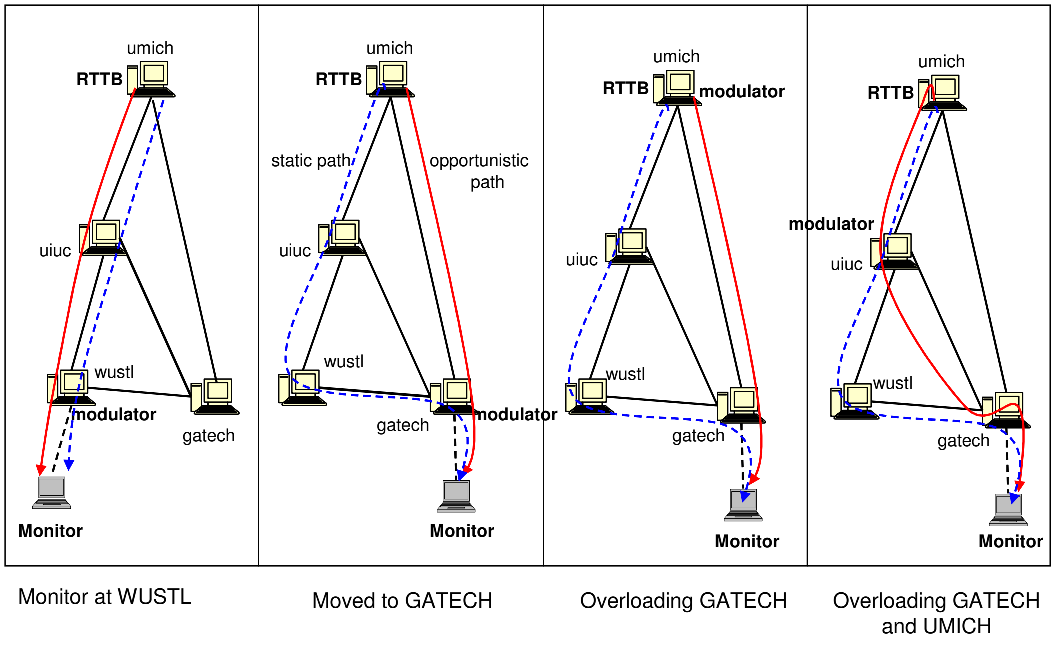


Figure 5.15: Experiment Configurations under 4 Scenarios

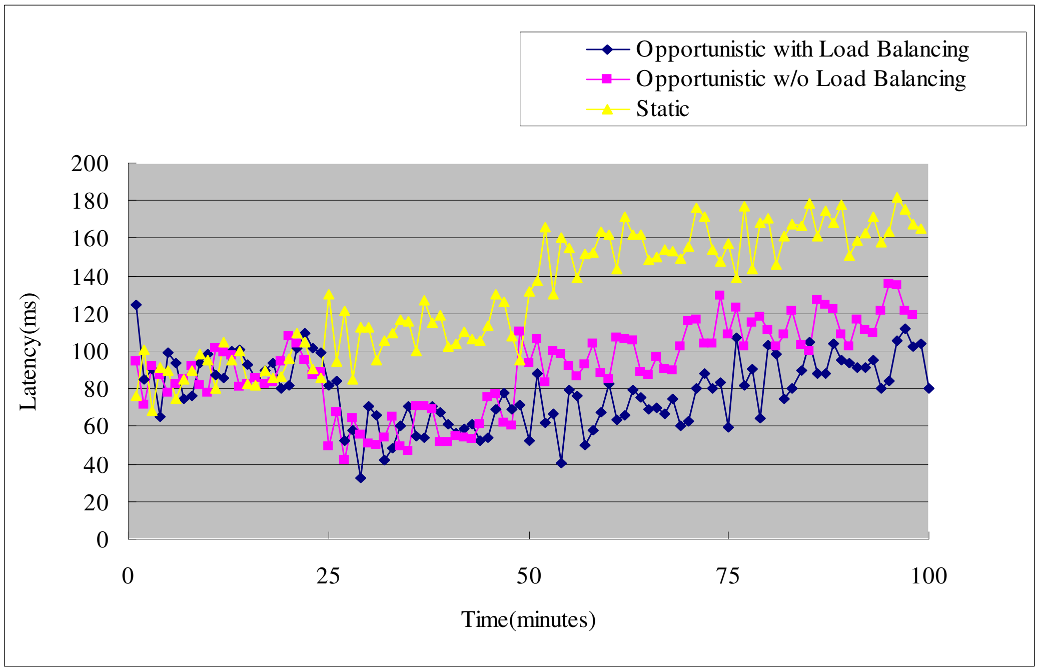


Figure 5.16: Timeline of Latency of RTTB Workbench

balancing, under 4 different scenarios. In the first scenario, the monitor runs at WUSTL. In the second one, the monitor moves to GATECH. Third, the monitor runs at GATECH, and GATECH and WUSTL are overloaded. The last scenario has a monitor at GATECH, and GATECH, WUSTL, and UMICH are overloaded. The last two scenarios are used for the purpose of evaluating the effect of the opportunistic approach's load balancing mechanism. The experiment configuration changes used under the 4 different scenarios are shown in Figure 5.15.

Latency results along our timeline are shown in Figure 5.16. The average latency results appear in Figure 5.17. As depicted in these figures, the opportunistic approach can delivery events more efficiently than the static one when a monitor changes location. Furthermore, the opportunistic approach with load balancing can further improve event delivery by handling broker overloads.

We discuss experimental results in more detail. Initially, the monitor program runs close to WUSTL and its home broker is WUSTL. The event delivery path is UMICH $\rightarrow$ UIUC $\rightarrow$ WUSTL $\rightarrow$ monitor. When the monitor program moves close to GATECH, the opportunistic approach changes the home broker to GATECH and relocates the modulator

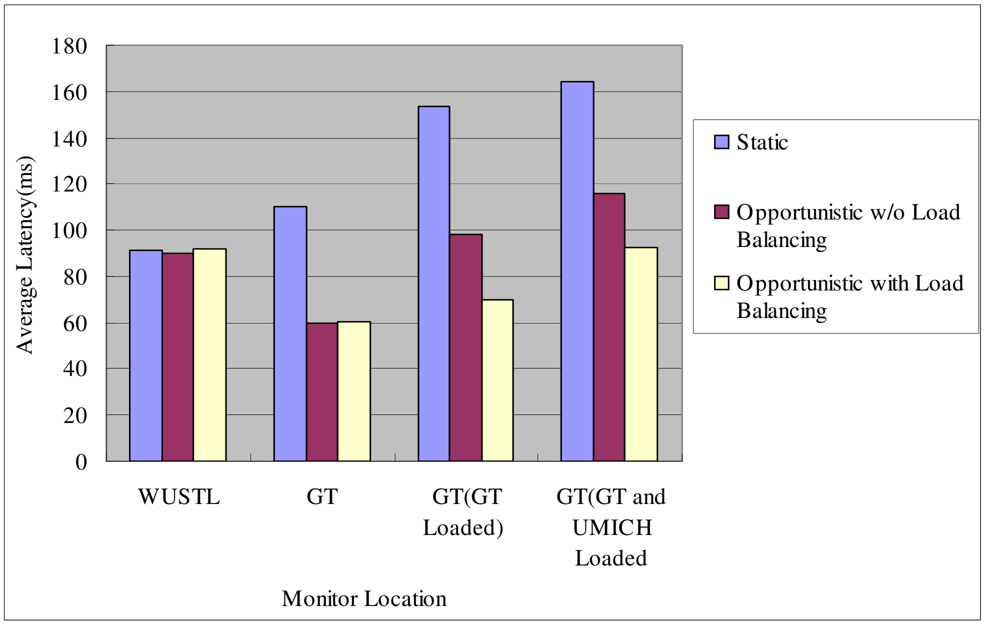


Figure 5.17: Average Latency of RTTB Workbench

from WUSTL to GATECH. The resulting delivery path of UMICH → GATECH →monitor is shorter compared to the path UMICH → UIUC → WUSTL → GATECH → monitor used by the static approach (the latter still uses WUSTL as the monitor's home broker). We next increase GATECH's load beyond some threshold and for fairness, for the static approach, we overload WUSTL. The opportunistic approach handles GATECH's overload by relocating the modulator to UMICH. As shown in the figures, because the modulator run faster on the less loaded UMICH node, the resulting event delivery used by the approach that combines opportunistic overlays with load balancing is more efficient than the approach without load balancing. We then increase UMICH's load and also overload it. Now both brokers (UMICH and GATECH) on the shortest path are overloaded. The opportunistic approach chooses the lightly loaded UIUC to run the modulator. After relocation, although the new broker path of UMICH → UIUC → GATECH → monitor is longer than the shortest path UMICH → GATECH → monitor, event delivery delay is actually lower than on the old path used by the approach without load balancing. This is because the modulator runs faster at UIUC and the performance gains attained by running the modulator at UIUC exceed the performance loss of using a longer delivery path.

The results in these experiments show that the opportunistic approach is practically applicable in actual networks and is able to improve the application's performance in terms of end-to-end latency.

CHAPTER VI

PERFORMANCE EVALUATION IN MOBILE AD HOC NETWORKS

6.1 Simulation

Simulation techniques are used to evaluate the opportunistic overlay approach under various wireless network configurations in mobile ad hoc networks. In all experiments reported in this section, the network consists of 100 mobile nodes that roaming in a 1000 x 1000 meter square. The random waypoint mobility model [56] is used with a pause time of 10 seconds. The radio transmission range of each node is 250 meters. Each simulation spans 600 seconds of simulated time. The model is designed to be similar to comparable research, such as PAST-DM [40].

It is important to note the limitations of the simulation results presented in this section. First, simulations do not consider link layer details, such as MAC-layer protocol functionality, link errors, or multiple-access interference. Second, they do not take into account physical layer characteristics (e.g., radio properties). As a result, we cannot model actual control overheads or the effects of link contention like increased message delay due to packet losses. Consequently, in actual systems, (a) overlay routing, (b) broker network topology creation, and (c) broker neighbor discovery may experience larger delays than those reported here, most notably under high packet loads or in ‘noisy’ environments. In addition, since we do not model the overheads of interactions across the overlay and protocol layers, the performance results achieved in our simulations are somewhat optimistic, indicating the strong potential for performance improvements rather than demonstrating specific improvements for specific platforms. An example is that at high mobility speeds, overlay adjustments may not be able to react to all changes in the underlying network layer. Finally, our simulation uses a “best-case” ad-hoc routing protocol, which computes the shortest paths between each pair of nodes from the entire connectivity topology. Future work should address these limitations by constructing a more comprehensive simulator with a MAC layer model (e.g., IEEE 802.11 MAC).

The point of this section's simulation results is to compare the relative performance of the opportunistic approach to overlay routing vs. static approaches. Results hold for moderately loaded networks, based on the following observations. First, as demonstrated with measurements in actual systems, the bandwidth usage implied to opportunistic overlay control messages is small, e.g., 2.4 Kbps per broker with an update period of 50 seconds. Hence, the additional contention at data-link layer caused by control packets (i.e., broker network topology propagation) is low. Second, for a data link layer protocol like 802.11 MAC, the chance of overlay routing packets being lost due to collisions during periods of high channel contention is small (packets may experience additional delays, however). Third, the addition of delays and overheads at lower layers does not negate simulation results. Instead, essentially, such delays increase the broker network update period, thereby causing opportunistic overlays to react more slowly to changes in physical network topology. Fortunately, simulation results shown that the opportunistic approach is fairly robust, that is, performance does not decrease rapidly with increases in update periods from 10 seconds to 100 seconds. In fact, even with an update period of 100 seconds, the opportunistic approach can still use much shorter paths than the static approach. Finally, the additional delays due to packet loss experienced at the network layer are small compared to overlay delays, i.e., they are several orders of magnitude smaller than the 10 second minimum update period used in our simulations. Therefore, the changes in path length that may be experienced if more precise MAC-layer models were used are likely to be small.

6.1.1 Performance of Broker Network

The first set of experiments evaluate the average lengths of network paths across broker overlays with vs. without opportunistic overlay protocols. Measurements consider only brokers, not clients. Unless stated otherwise, 40 nodes among 100 are randomly chosen as brokers. Experimental configurations are varied (1) with different broker network topology update intervals, (2) different mobility speeds, and (3) different numbers of brokers. The

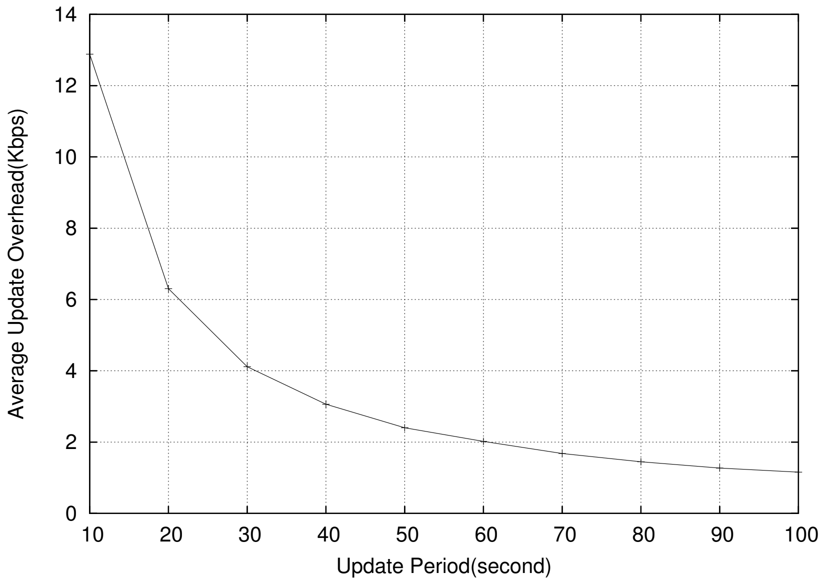


Figure 6.1: Update Overhead versus Update Period

path length from broker B1 to broker B2 is the corresponding physical network distance of the shortest broker path between B1 and B2, which is computed based on B1's local knowledge of current broker network topology. The average path length from each broker to all other brokers is computed and averaged over all brokers. In order to establish a basis for comparison, we also measure the average length of the physical network path between each pair of brokers. In addition, the overheads of broker network topology update with different update intervals are evaluated.

Update Overhead versus Broker Update Period. The overheads of broker network updates are shown in Figure 6.1. Overhead is computed as the average bandwidth each broker requires for propagating its topology knowledge to its neighbors. The argument is that network resources tend to be scarce in pervasive systems, which means that solutions like ours cannot require undue amounts of network resources. Results show that update overheads are modest, requiring a total of 2.4 Kbps bandwidth with an update period of 50 seconds. These results also imply that broker updates will cause only small levels of additional contention at the data link layer.

Average Path Length versus Broker Update Period. Figure 6.2 shows the average path length versus the update interval, the latter varying from 10 to 100 seconds. As expected, with increased update intervals, path length increases since larger update intervals

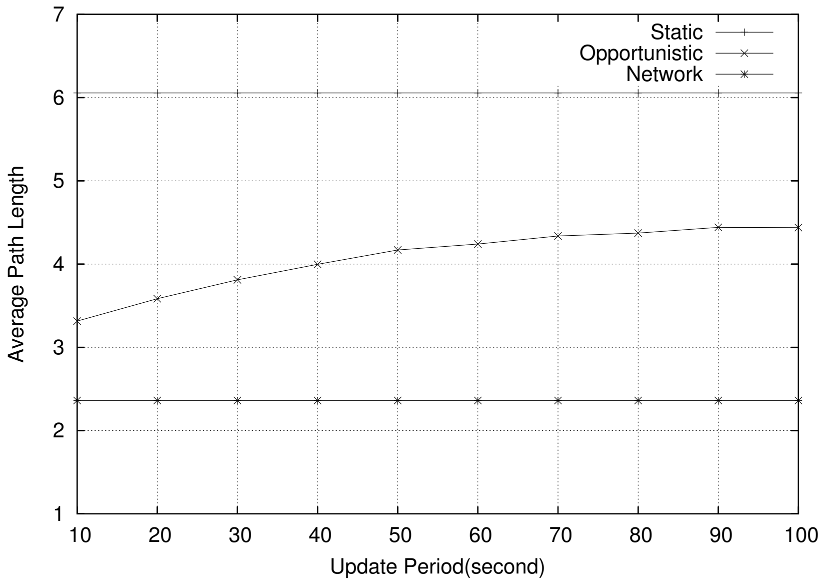


Figure 6.2: Average Path Length among Brokers versus Update Period

imply slower reactions to changes in physical network topology. However, as shown in the figure, the change in path length is not rapid with the increase of update periods. Even with a relatively low update period of 100 seconds, the opportunistic approach still uses much shorter network paths. Since the addition of delays at lower layers has the similar effect of increasing the broker update period and the delay is normally orders of milliseconds, one important insight is that simulation results will not be substantially different if we also modeled lower layer overheads. More frequent update will result in larger performance difference of opportunistic approach between our simplified simulation model and the simulation model including MAC layer detail since more frequent update requires more bandwidth usage and hence creates higher contention at data link layer.

Path Length versus Mobility. Figure 6.3 reports performance for different mobility speeds. The obvious result is that the opportunistic approach performs better at all speed ranges. Increased mobility speeds like those attained by cars, for instance, imply more rapid changes in physical network topology which in turn requires more frequent updates of broker networks.

Update Overhead versus Number of Brokers. Figure 6.4 shows the overheads of broker network update versus the number of brokers. Overhead increases with broker network size, of course, since larger broker networks imply larger data exchanges during

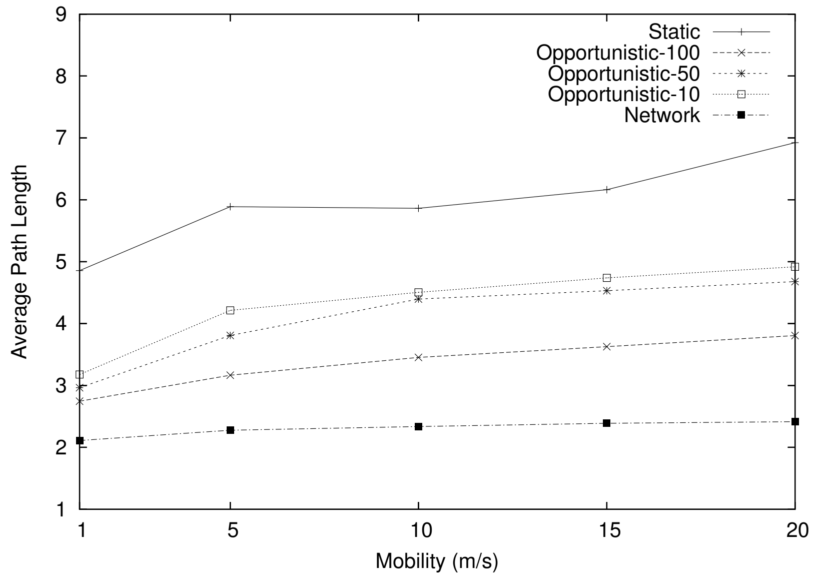


Figure 6.3: Average Path Length among Brokers versus Mobility

update. As shown in the figure, although overhead increases rapidly with an update interval of 10 seconds, it increases more slowly for less frequent updates. For instance, the average bandwidth usage of 2.55Kbps with 100-second update for a 60-broker overlay should be acceptable. Overhead may be reduced further by using a better state routing protocol like the one reported in [74].

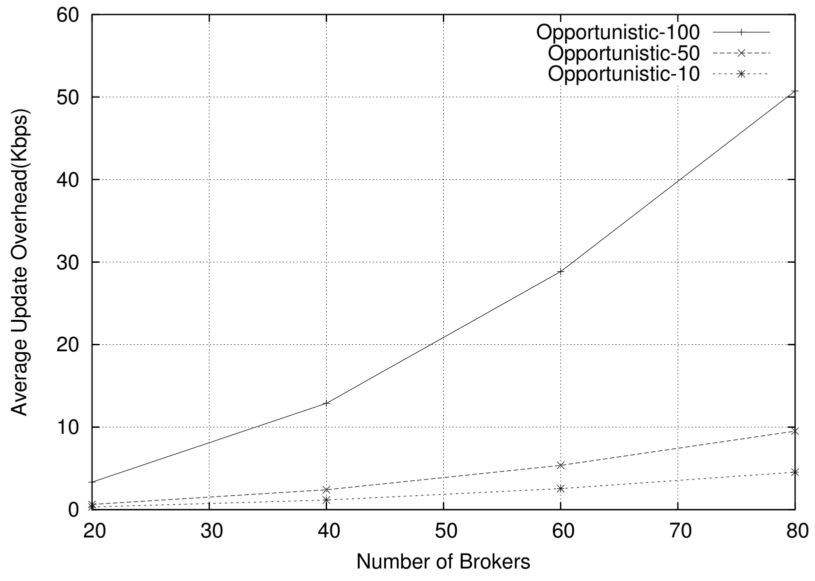


Figure 6.4: Update Overhead versus Number of Brokers

Average Path Length versus Number of Brokers. Results reported here concern how average path length changes with the number of brokers. We vary the number of brokers and measure average path length. The results in Figure 6.5 show that the opportunistic overlay approach outperforms the static one for all broker network sizes.

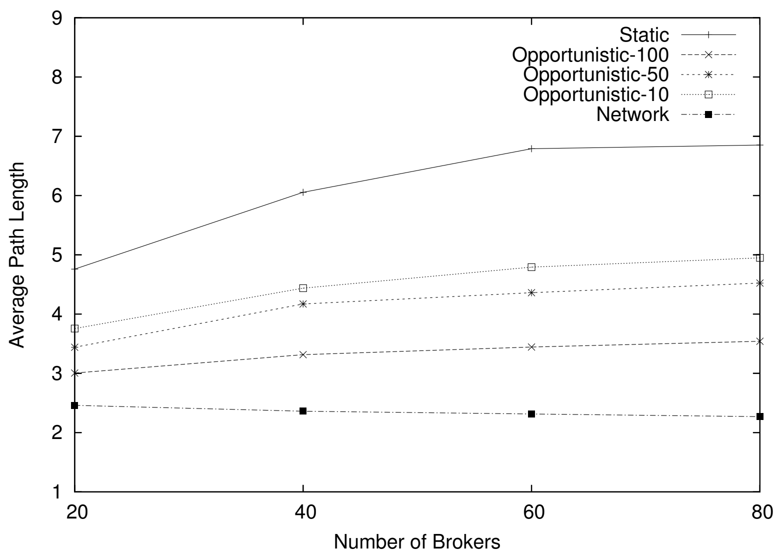


Figure 6.5: Average Path Length among Brokers versus Number of Brokers

Timeline of Path Length. Figure 6.6 shows how path length changes in a simulation with an update interval of 50 seconds and mobility speeds between 1 m/s and 20m/s. As shown in the figure, the opportunistic approach can deliver events more efficiently than the static approach at almost all time points. At the beginning, both the static and the opportunistic approaches have similar path lengths, since the initial broker network matches physical network topology. As time passes, the path lengths of the static approach increase rapidly because the initial broker network topology cannot reflect changes in physical network topology caused by node mobility.

6.1.2 Performance of Event Delivery between Mobile Clients

Most relevant to our work, of course, is the end-to-end performance experienced by end users, i.e., clients. In the following experiments, we randomly choose 20 brokers, 10 event producers, and 60 event consumers from 100 mobiles nodes. In this set of experiments, the broker network update interval is fixed at 20 seconds, and we vary mobile clients' home broker discovery periods as well as mobility speeds. Four different approaches are evaluated in terms of the resulting average path lengths between each pair of producer and

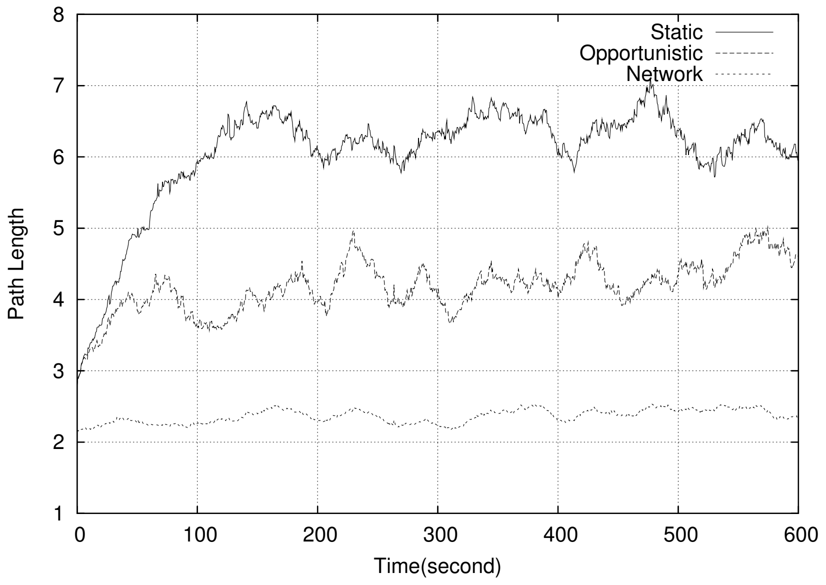


Figure 6.6: Timeline of Path Length among Brokers

receiver: (1) the static approach changes neither the broker network topology nor the home brokers of mobile clients; (2) the static-opportunistic approach changes mobile clients' home brokers only; (3) opportunistic-opportunistic changes both the broker network topology and mobile clients' home brokers; and (4) the 'best' approach keeps updating the broker network whenever the physical network changes; it calculates shortest broker paths based on up-to-date physical network topology data. Although the best approach is not practical, we have included it to establish a basis for comparison.

Timeline of Path Length. Figure 6.7 shows the performance results of a simulation with a home broker discovery period of 10 seconds. We can see that the opportunistic-opportunistic approach performs best among the three realistic approaches. Even the static-opportunistic approach can improve event delivery significantly compared with the static approach.

Average Path Length versus Nearest Broker Discovery Period. The results of path length versus home broker discovery period are reported in Figure 6.8. With increased closest broker discovery periods, path length increases slightly. As discussed in Section 3.5.3, the mobile client can find the nearest broker either by querying its routing table or by using an expanding ring protocol. Either way, the cost is small compared with the overheads of broker update operation. More frequent closest broker discovery results in more frequent home broker changes, hence more frequent modulator relocations. Since the costs of

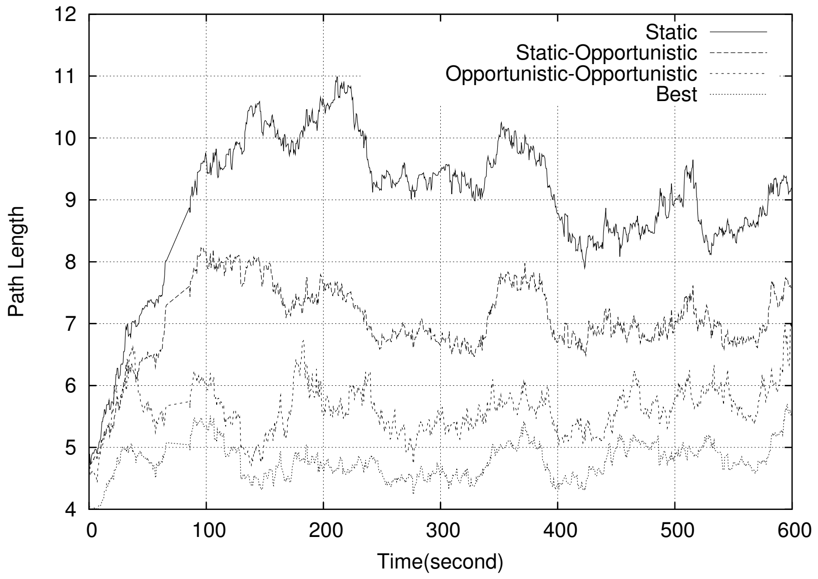


Figure 6.7: Timeline of Average Path Length between Producers and Consumers

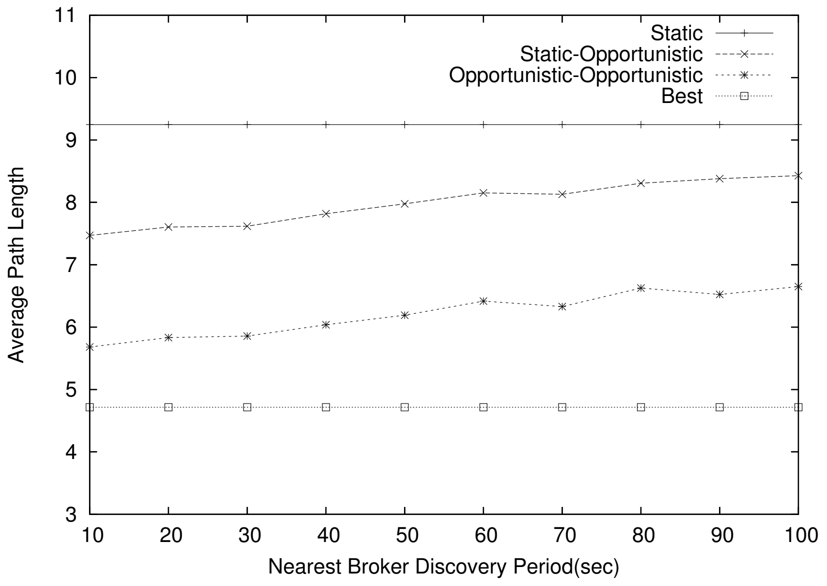


Figure 6.8: Average Path Length between producers and consumers versus Nearest Broker Discovery Period

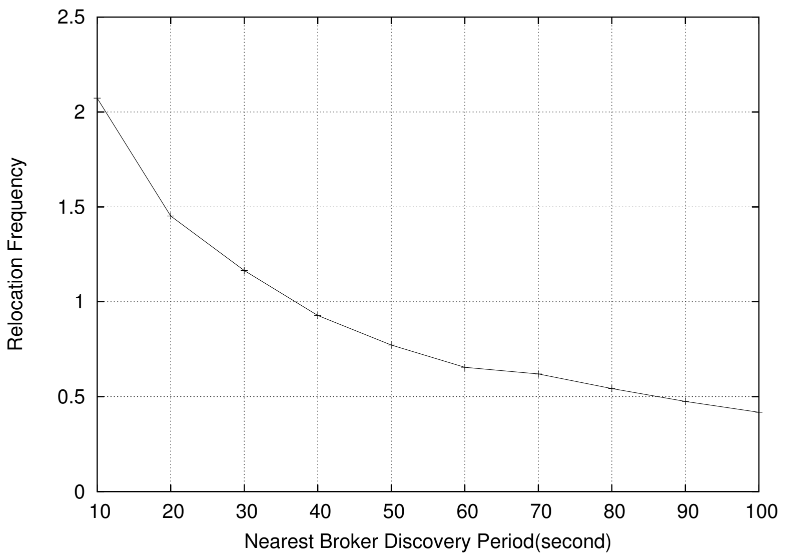


Figure 6.9: Modulator Relocation Frequency versus Nearest Broker Discovery Period

modulator relocation depend on modulator complexity and state, we do not explore those further.

Home Broker Change Frequency versus Nearest Broker Discovery Period. This experiment evaluates the relationship between home broker change frequency and the periodicity of nearest broker discovery. Figure 6.9 shows the modulator relocation frequency versus nearest broker discovery period. As shown in the figure, there are about 2 relocations per second with a discovery period of 10 seconds. The relocation overhead can be estimated by multiplying relocation frequency with the size of modulator state, the latter depending on modulator complexity. Please note that modulators typically carry out limited tasks, as demonstrated by their use in the multiple JECho applications developed in our research. As a result, modulator states tend to be small, with 100 bytes being a typical size. This means that modulator relocation costs are smaller than the overheads due to broker network topology exchanges. Moreover, there is a substantial set of pervasive applications that do not require consistent modulator states during migration, as exemplified by overlays that process and forward repeated sensor readings [108]. For such applications, modulator state relocation is unnecessary, thereby enabling opportunistic overlays to perform nearest broker discovery more frequently.

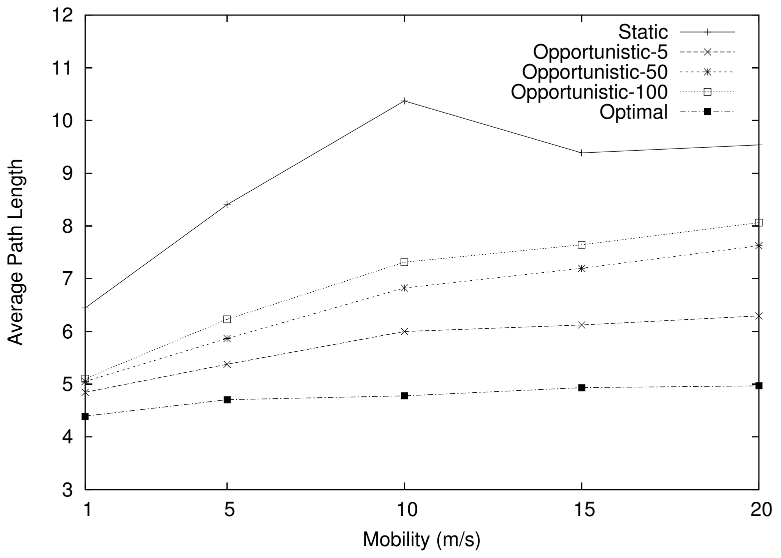


Figure 6.10: Average Path Length between Producers and Consumers versus Mobility

Average Path Length versus Mobility. The average path length at different mobility speed is shown in Figure 6.10. The figure shows that more frequent nearest broker discovery achieves shorter event delivery paths. In particular, when the nodes move at a very fast speed, increasing the discovery frequency can improve the performance significantly.

6.2 System Emulation

In order to evaluate the effects of load balancing, we have conducted a set of experiments on an ad-hoc wireless network emulator. The mobility emulator runs on a Linux cluster of 20 nodes with MobiEmu [111] running on each node. The cluster network is a gigabit Ethernet switch. MobiEmu is a software platform for testing and analyzing ad-hoc network protocols and applications. With control software running on each node, MobiEmu mimics dynamic connectivity among nodes by dynamically installing or removing packet filters for specific MAC addresses. Since we focus on load balancing in this set of experiments, we use the 'best-case' ad-hoc routing provided by MobiEmu software. These protocols always deliver packets via the shortest network paths (see [111] for more detail about MobiEmu).

The emulated mobile network consists of 20 mobile nodes, of which 5 nodes are event brokers, 5 are event producers, and 15 are event consumers (5 brokers reside at event receiver nodes). In our experiments, mobile nodes move in a space of 750m x 500m and use random

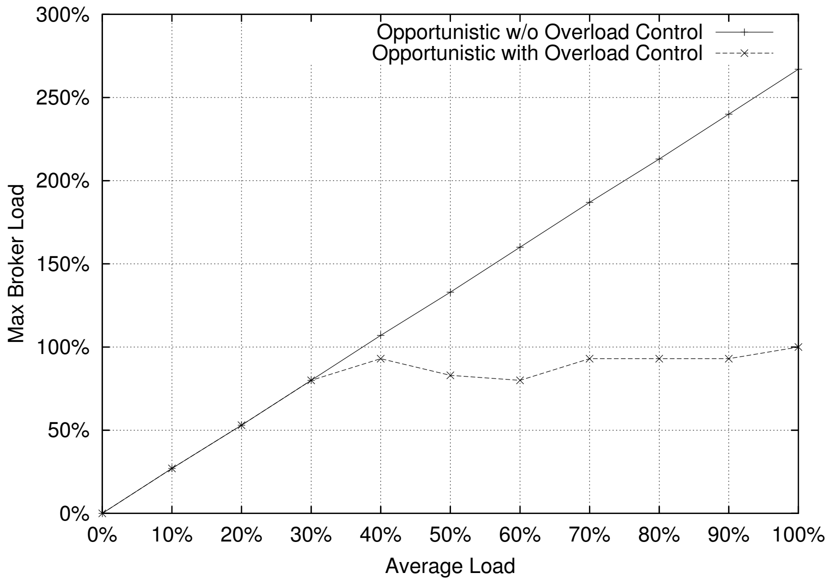


Figure 6.11: Maximum Broker Load versus Average System Load

waypoint mobility with a pause time of 30 seconds and speeds between 1m/s and 20m/s. Due to the relatively small network size, the broker update period is set to 5 seconds, and the nearest broker discovery period to 1 second. We vary the average load of the broker network and measure the maximum load during execution.

Results appear in Figure 6.11. As shown in the figure, when broker load is relatively light (i.e., less than 30%), there are no overloaded brokers, and both approaches behave the same, where the nearest broker to a mobile client is always chosen as the client's home broker. With increased system load, without overload control, some brokers become overloaded. The load balancing algorithm ameliorates this problem, because a client's home broker will not be moved to an overloaded node, even if that node is closer than the old one. The positive outcomes of load management reported in these measurements are moderate, of course, since in random waypoint mobility, nodes move independently. Load balancing is more important and will have more significant effects when nodes move in groups, as exemplified by conference participants moving from one presentation venue to another, for instance.

Figure 6.12 depicts path length versus system load. When system load is less than 70%, the opportunistic approach always chooses the nearest broker as home broker, and the delivery path can be more than 1 hop shorter than in the static approach. With

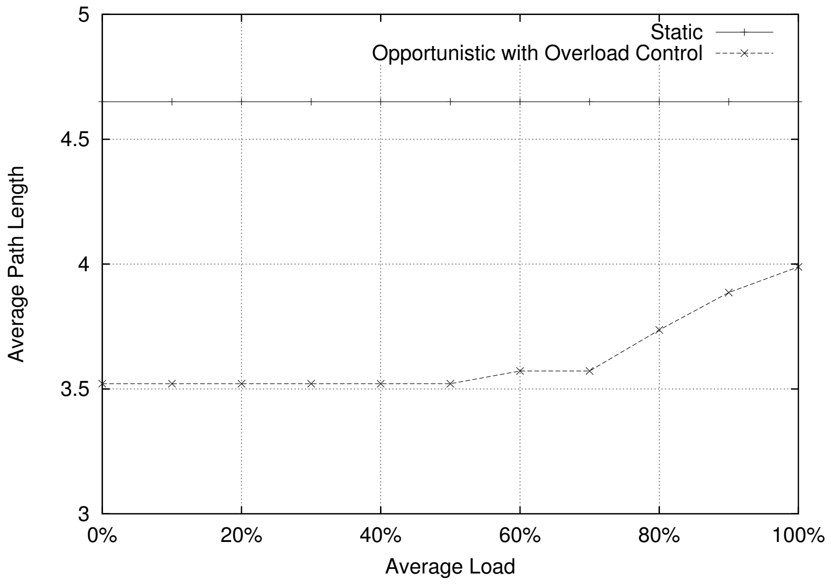


Figure 6.12: Path Length versus System Load

increased system load, load balancing selects brokers with lighter loads on relatively longer paths, resulting in increased path lengths. However, the opportunistic approach continues to outperform the static approach even when system load reaches 100%.

6.3 Flood Warning Application

The last set of experiments demonstrates the practical utility of our approach, by running a sample application on a small wireless testbed. The testbed consists of 3 laptops A, B, and C. A is a Mobile Pentium 1GHz with 512M memory, B is a 1.7GHz machine with 512M memory, and C is a 700MHz machine with 128M memory. Wireless connectivity is provided by Orinoco 802.11b cards. These cards are set to ad-hoc mode on channel 8. No WEP encryption is used. All three laptops use the UoB JAODV version 0.2, an AODV implementation in Java [49]. In order to simulate network connectivity changes, we dynamically set filters at the MAC layer. In a network consisting of 3 nodes A,B and C, there are four possible network topologies without network partitions: A–B–C, A–B–C–A , A–C–B and B–A–C.

The basic experiment configuration is shown in Figure 6.13. The experiments being performed run a flood watch application on the testbed, comparing the event delivery latency of the opportunistic with the static approach using the 4 different network topologies.

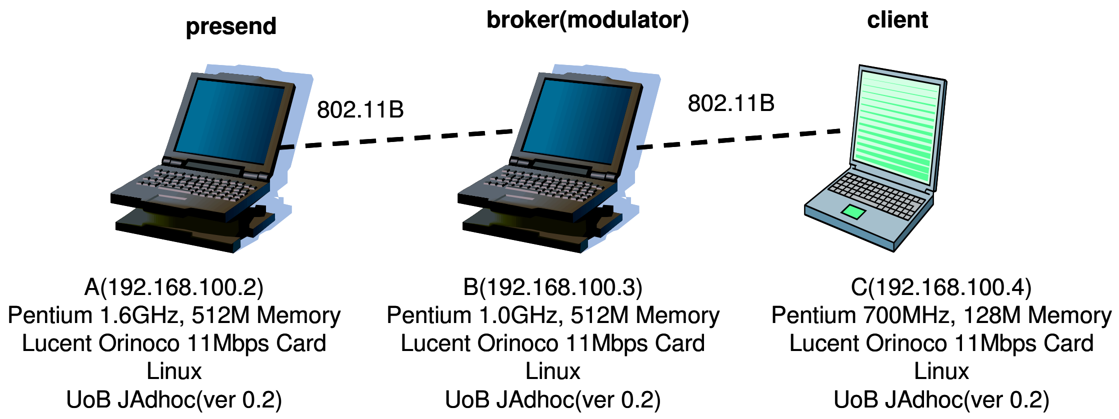


Figure 6.13: Basic Experiment Configuration

The application consists of two programs and works as follows. PreSend reads precipitation data from a file and places normalized precipitation data on an event channel. A client program subscribes to the channel and provides a modulator that calculates water depth from precipitation data and terrain topology data using a runoff model. The client is typically interested in flood information in some specific area, which can be defined by a two-dimensional bounding box. Flood data that is outside the bounding box will be filtered (i.e., removed) before data is sent to the channel. The output of full-size flood data is a 100 x 100 array of doubles.

We envision a scenario in which this application is used in a flood rescue action. In a flood disaster area, a rescue team is equipped with several mobile vehicles with relatively powerful computers (e.g., desktops) installed on each of them, and each team member or group has a handheld device (e.g., an IPAQ) or a lightweight laptop. All computers have communication hardware (e.g., 802.11b wireless cards) and software (e.g., the AODV protocol), so that they can communicate with each other via the ad-hoc wireless network. Brokers reside at computers on mobile vehicles and form a virtual network. They receive real weather data either from a weather center via a satellite connection or from sensors distributed across the area. They send collected data to an event channel. The client program running on each team member's mobile computer connects to a broker and receives the data of interest. Different people may be interested in receiving different data, which involves running different client programs and/or using different modulators.

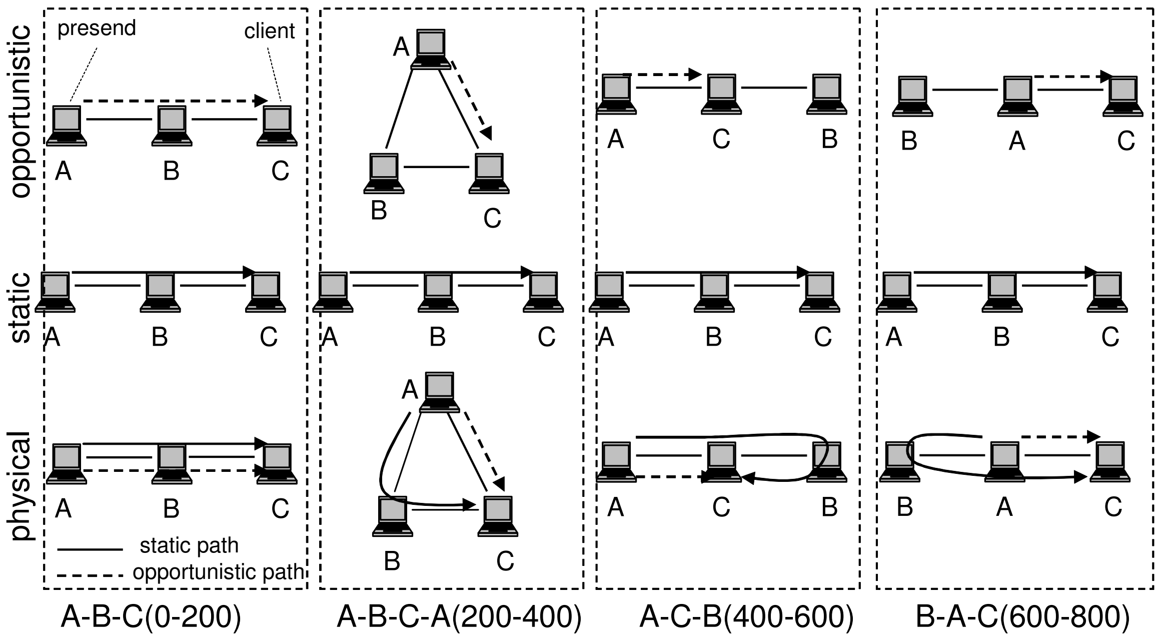


Figure 6.14: Changes in Experiment Configurations

In our experiment, PreSend is running on laptop A, and the client program runs on laptop C. Broker programs run on both A and B. The physical network topology changes every 200 time units. Changes in experiment configuration are shown in Figure 6.14. The initial physical network topology is A–B–C. B is C's home broker. Both approaches use the same event delivery path A→B→C. After 200 time units, the network topology changes to A–B–C–A, so that the opportunistic approach chooses A as C's home broker and relocates its modulator from B to A, hence resulting in the shorter delivery path A→C. During the time interval of 400 to 600, the network topology is A–C–B, where the opportunistic overlay still uses A→C as the delivery path, corresponding to the same physical network path A→C. The physical network path of A→B→C used by the static approach becomes A→C→B→C. In the final interval, the network topology changes to B–A–C, where the opportunistic approach still uses the same path A→C, and the static approach's path A→B→C now corresponds to the physical network path A→B→A→C, with overlap at A.

Figure 6.15 depicts latency comparisons with four topologies when the client is interested in all data. As shown in the figure, the opportunistic approach can deliver data up to 6 times faster than the static approach. The static approach has its worst performance in the

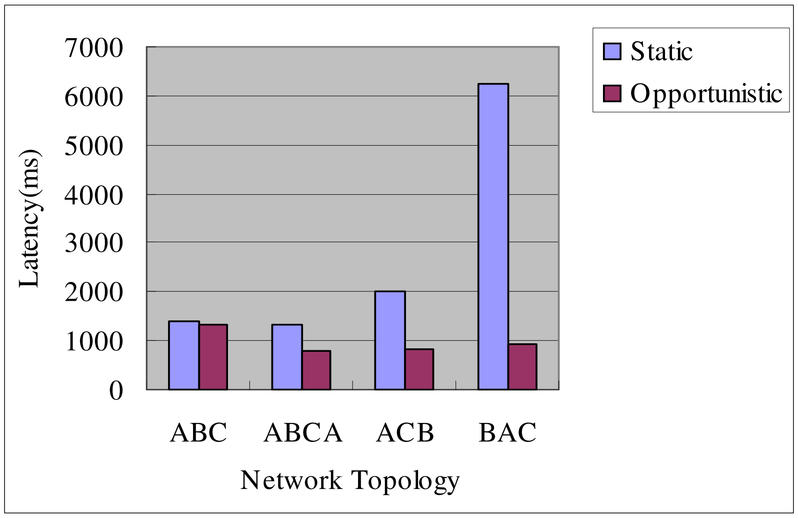


Figure 6.15: Average Latency of Flood Application (100% data)

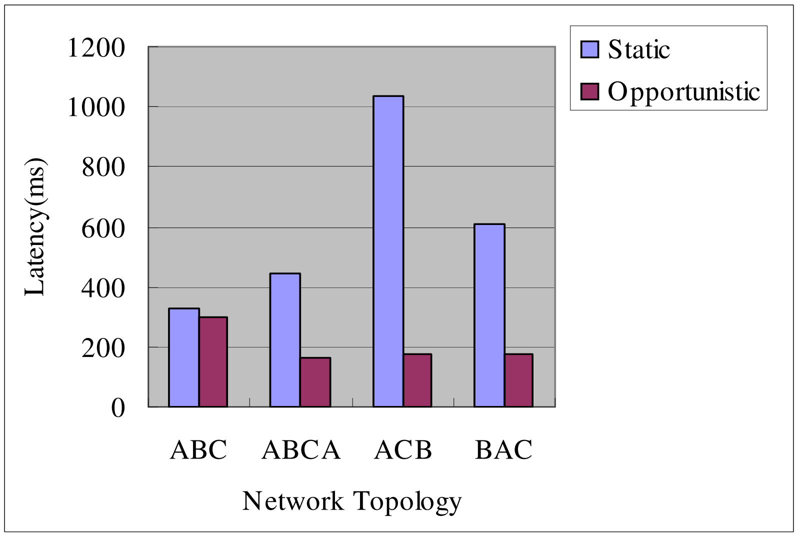


Figure 6.16: Average Latency of Flood Application (10% data)

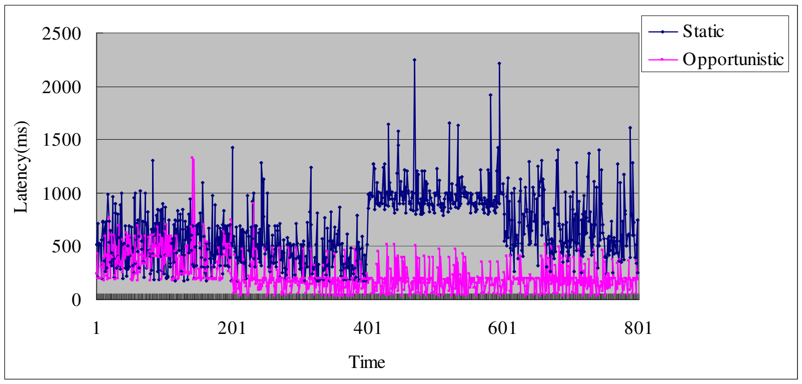


Figure 6.17: Timeline of Latency of Flood Application (10% data)

configuration of B–A–C (6000ms compared with to the opportunistic approach's 1000ms), where data delivery following the path A→B→A→C results not only in a longer path but also in additional network bandwidth usage at A, since at the physical layer, every event is actually delivered twice at A.

When a client is interested in only 10% of the area data, the opportunistic approach coupled with its use of modulators shows additional improvements. Results depicted in Figure 6.16 show that by relocating C's modulator from B to A, the opportunistic approach not only delivers data following a shorter path but also delivers less data, hence improving the application's performance significantly: the average latency is less than 200ms under all network configurations. Note that in this scenario, the static approach performs worse under network topology A–C–B than B–A–C because only 10% of the data reaches A for the second time in the latter configuration, while all data reaches C in the former one. The fact that machine C is less powerful than A may also contribute to the observed difference in performance. Figure 6.17 shows the timeline of event latency when clients are interested in only 10% of the data.

The key result of the testbed experiments presented here is that it is important to dynamically adjust the middleware overlays used in pervasive systems. The opportunistic overlay approach described and evaluated in our research is one method for runtime overlay management and by using it, significant performance improvements can be attained compared to non-adaptive approaches. However, the small scale of the testbed used in these experiments limits the generality of the results presented here. First, with larger numbers of machines, there will likely be interference and congestion effects. These will lead to increased costs and delays for managing overlays and therefore, reduce the absolute performance benefits of the opportunistic vs. static approaches being compared. Second, the testbed uses MAC address filters to simulate node mobility, but in actual adhoc networks with mobile nodes there will be additional communications compared to the filter-based communications in the testbed (since all nodes in the adhoc networked system still communicate with each other). This will result in additional congestion in actual systems.

CHAPTER VII

RELATED WORK

This chapter examines the related work from different areas including publish/subscribe systems, mobility support in publish/subscribe, peer-to-peer systems, application-level multicast, general overlay networks, and wireless network routing protocols.

7.1 Publish/Subscribe Systems

Traditional invocation-based middleware (CORBA, RPC, Java RMI) uses request/reply communication and is a useful abstraction for building distributed systems, but it does not work well for large-scale, Internet wide systems and unreliable and dynamic environments because of its tight-coupling (synchronous RPC paradigm) and one-to-one communication [114, 80]. Event-based middleware addresses the shortcomings of request/reply communication by supporting asynchronous, many-to-many, loose coupling communication, where producers publish information and consumers state their interests on information and the event system is responsible for delivering this information from producers to consumers. Event-based systems can be classified into two categories: topic-based publish/subscribe and content-based subscribe.

7.1.1 Topic-based Publish/Subscribe

In topic-based publish/subscribe, event producers publish events with respect to a topic and event subscribers specify their interest in a topic and receive all events published on that topic. Two such systems are TIBCO/Rendezvous [103] and the Information Bus [72]. Since a consumer's subscription is based on a topic, rather than on event content, they provide limited capabilities for event customization. In comparison to the above work, the opportunistic overlay framework presented in this dissertation has been built on top of a content-based publish/subscribe system [116, 114], which provides maximum expressiveness by supporting a general function based subscription.

7.1.2 Content-based Publish/Subscribe

Content-based publish/subscribe supports event subscription defined on event content, from simple predicate-based filter to general function. There are several research efforts concerned with the development of content-based publish/subscribe systems, including Gryphon [99], Siena [11], JEDI [22], Elvin [93], Rebeca [29], ECho [25] and JECho [116].

Gryphon [99] is based on an information flow model and implements publish/subscribe event delivery with JMS interface [100]. By defining efficient algorithms to match events against content-based subscriptions, Gryphon can be deployed to large scale. Gryphon also includes several extensions for event delivery, including guaranteed delivery [6], durable subscriptions [5] and relational subscription [53].

Similar to Gryphon, Siena [10, 11] delivers events through an overlay network of brokers. Subscription are conjunctions of predicates over the event attributes. Several broker network topologies are proposed in Siena. A routing algorithm based on the idea of reverse path forwarding of messages in broadcast algorithms is used. This algorithm pushes subscriptions close to producers and allows filtering to take place as early as possible. Siena is targeted at Internet-scale deployment.

JEDI [22] is a Java implementation of content-based publish/subscribe system. JEDI organizes event dispatchers as a tree structure and uses a hierarchical event routing, which propagates subscriptions and events upward. Events are propagated downward too when a matched subscription is met at a dispatcher.

Elvin [93, 95, 94] provides content-based routing of events with a predicate-based subscriptions language. Elvin's quenching mechanism [95] enables a publisher to stop delivering events when there is no one who is interested the generated events.

Rebeca [30, 68, 28, 31] focuses on advanced routing mechanisms and supporting the development of large scale e-commerce applications. Rebeca [30, 31] proposes a modular approach to build structured event-based systems and introduces a new notion of scope, which bundles a set of producers and consumers in order to utilize locality. The visibility of published events is restricted by the scopes and their composition.

ECho [25, 26, 7] event system supports flexible and high performance event-based communication in heterogeneous environments. ECho efficiently supports event transmission across heterogeneous machines and achieves performance similar to communication systems(e.g., MPI) typically used for high-performance applications [26, 7]. In addition, a derived event channel is used in Echo to enable the dynamic customization of event notification by the instantiation of general filter functions at the source [25].

The above systems have been originally built for static environments, where nodes do not move, and physical network topology remain fixed. None of them provides dynamic reconfiguration of application-level networks. Most of them assume a static event dissemination structure. For example, the overlay networks of event brokers in Gryphon and Siena are static and must be specified at deployment time. Having fixed event dissemination structures makes them difficult to adapt to changed network conditions, hence unsuitable for applications in mobile environments where physical network topology and node locations change continuously.

In addition, the above publish/subscribe systems perform event filtering with predicate-based subscriptions; they do not support the general event processing needed for the complex data conversions occurring in multimedia, business, or scientific applications. Opportunistic overlays are realized with the JECho pub/sub infrastructure [116, 115, 114]. JECho generalizes the capabilities of other event systems, by using consumer-provided functions, termed event modulators [115]. The intent is to address the severe resource limitations existing in many mobile and embedded systems, by permitting event consumers to deploy application-specific functions that manipulate event content into event sources and/or brokers, so as to precisely meet their current needs, and to avoid needless data transfers. Generic function-based subscription makes the opportunistic overlay system more feasible for developing applications in pervasive systems and mobile environments.

7.1.3 Peer-to-Peer Publish/Subscribe

By providing a decentralized, scalable and self-organizing approach, peer-to-peer technology is becoming attractive for building highly scalable systems that operate on an Internet-scale.

Several recent publish/subscribe systems [80, 82, 102, 105] have integrated content routing on top of peer-to-peer networks. Most of them use Distributed Hash Table (DHT). Though peer-to-peer networks can offer self-organization and can be closed tied to the topology of the underlying network, peer-to-peer publish/subscribe systems do not deal with reconfiguration at the content routing level. Instead, they completely rely on the peer-to-peer networks to handle the dynamics. Furthermore, none of the peer-to-peer publish/subscribe systems has addressed mobility issues introduced by mobile applications. Several application-level multicast systems have been built over peer-to-peer networks, including Scribe [92], Bayeux [117], muticast CAN [88]. They are equivalent to topic-based publish/-subscribe systems and do not support content-based routing.

7.1.4 Dynamic Reconfiguration in Publish/Subscribe

Though the majority of currently available public/subscribe systems do not deal with reconfiguration, there are a few work in the literature supporting dynamic reconfiguration somehow. Picco et al. [79] present an algorithm for topological reconfiguration in content-based publish/subscribe due to changes in underlying connectivity. Compared with opportunistic overlays, the reconfiguration in [79] is simpler, involving only a link removal or insertion. It has not given any detail on how to apply the proposed approach to handle changes in mobile environments. In addition, the approach assumes a tree-based topology between dispatchers, which makes it hard to achieve robustness since a single link failure partitions the tree. Baldoni et al. [106] propose a different approach for dynamic reconfiguration of the broker network. The reconfiguration in [106] aims at placing close to each other brokers that manage similar subscriptions while the reconfiguration in the opportunistic overlays is based on the nodes' physical locations and the underlying physical network topology. Similar to the approach in [79], the topology used in [106] must be maintained acyclic while the opportunistic overlay approach supports general broker overlay topologies. Hermes [80] supports limited reconfiguration by providing a repair mechanisms in case of brokers' faults.

7.2 Mobility Support in Publish/Subscribe Systems

There has been a moderate amount of research for supporting mobility in publish/subscribe systems.

7.2.1 Publish/Subscribe in Infrastructure-based Mobile Environments

Several research efforts have focused on extending existing event-based systems to support mobility in infrastructure-based mobile environments [21, 101, 27, 8, 17]. JEDI offers moveOut and moveIn operations that enable subscribers to disconnect and reconnect at a different network, requiring applications to explicitly call those operations [21]. Similarly, the mobility extension of Siena requires explicit request by applications [8]. Elvin supports disconnection and reconnection by using a central caching proxies [101]. [27] extends Rebeca to support mobile and location-dependent applications by rebinding a client to different brokers transparently and offering a client a fine-grained control over notification delivery in the form of location-dependent filters. [44] discusses general ideas about how to adapt a publish/subscribe system to mobile environments, but no design and implementation is described. As part of the thesis work, we present an approach to support the dynamic change of home brokers in infrastructure-based wireless networks, where only the last connection between broker and client is a wireless link [17].

The above research on event-based middleware for wireless networks focuses on applications in which mobile nodes make use of the wireless network to connect to a fixed network infrastructure, such as the Internet. Instead, the opportunistic overlay handle mobility in both infrastructure-based mobile environments and mobile ad hoc networks. In addition, the mobility extensions in JEDI [21] and Siena [8] depend on the application's recognition of mobility when taking appropriate actions. Most importantly, none of the above extensions provides general adaptation support for dynamic reconfiguration and redeployment of the event dissemination structures. The opportunistic overlays presented in this paper differ from all of the systems mentioned above in that they are designed for both infrastructure-based mobile environments and wireless ad-hoc networks, support dynamic reconfiguration and redeployment of event dissemination structure, and offer behaviors transparent to applications.

7.2.2 Publish/Subscribe in Mobile Ad Hoc Networks

Steam [67] is an event-based middleware service designed for ad-hoc wireless networks. It targets application scenarios where nodes are more likely to interact when they are in close proximity to each other. Our approach aims to support general event delivery in mobile ad-hoc networks. Further, Steam uses an implicit event model without intermediate broker nodes, while our approach uses explicit broker networks. Both systems focus on the timely delivery of events, but Steam also allows applications to define delivery deadlines and assigns them to specific events.

Huang et al. [45] present a distributed protocol to construct optimized publish/subscribe trees in ad-hoc wireless networks [45]. Similar to our work, they focus on how to make a publish/subscribe system work in mobile ad-hoc networks. Their algorithm builds multicast trees directly on top of lower level radio broadcast primitives, while our work uses an overlay broker network approach and depend on the underlying network infrastructure to provide basic network connectivity. Another difference is that their approach assumes a relatively stable environment with occasional reconfigurations followed by periods of stability. Opportunistic overlays do not make that assumption, and they can actually handle high levels of mobility as shown by our experimental results. Finally, their research defines cost metrics based on the total amount of work performed by all tree nodes in a multicast tree while our approach focuses on end-to-end cost metrics like optimizing network distance with the constraint of event processing time.

Yoneki et al. [110] propose a content-based publish/subscribe system for MANETs, which integrates an extended ODMRP (On-Demand Multicast Routing Protocol [60]) and content-based subscriptions. Similar to [45], ODMRP-PUB/SUB delivers events by creating multicast groups. The difference is that ODMRP-PUB/SUB uses a mesh-based approach instead of the tree-based one used in [45]. Since a consumer's subscription is a general function applied to events in our system, the approach of combining a multicast protocol and subscription aggregation/match is not readily applicable in our case. Further, ODMRP-PUB/SUB focuses on the routing between brokers and does not address the issue of delivery from brokers to producers/consumers. The purpose of ODMRP- PUB/SUB is to optimize network throughput while our opportunistic method focuses on providing timely event delivery.

The use of overlay broker networks differs our approach from the above approaches. The use of broker overlays enable our system to effectively exploit reliable delivery mechanisms provided by underlying network protocols [56, 78, 77]. The advantages of the overlay approach are robustness and low overhead since network connectivity is totally handled by the underlying network protocols. In addition, traditional publish/subscribe systems are usually built on top of broker overlay networks. The opportunistic overlay approach keeps publish/subscribe system's model and semantics and makes it easy to extend existing publish/subscribe systems and applications to mobile environments. The main problem for overlay approach is low efficiency in terms of data delivery delay. Our approach address efficiency issue by dynamically reconfiguring overlay topology to react to changes in physical network topology. As demonstrated in our performance evaluation, the performance of our system can be improved significantly. The event delivery delay is much better than static overlay approach and is comparable with the optimal. Finally, since consumer's subscription is a general function defined in events in our system, the approach combining multicast protocols and subscription aggregation used in [45] and [110] is not readily applicable for our system.

7.3 Application-level Multicast

7.3.1 Multicast in Fixed Networks

Application-level multicast schemes built over overlay networks have been proposed for many years [33, 13, 48, 75, 19]. Traditional application level multicast is based on unstructured overlays and can be classified into two classes: tree-first approach or mesh-first approach. Tree-first approach directly builds an overlay tree topology for multicast out of the physical networks (e.g., Yoid [33] and ALMI [75]). Mesh-first approach first constructs a mesh on top of the physical networks and then generates a source-specific tree based on the mesh topology, such as Narada [19] and Scattercast [13]. That application-level multicast schemes need to dynamically manage overlay multicast tree or mesh, is similar to dynamic overlay broker maintenance used in opportunistic overlays. Specifically, [20, 33, 19] propose mechanisms for nodes in multicast overlays to gradually improve the overlay structures to optimize applications' performance (e.g., delay, bandwidth or both). These schemes basically have nodes periodically probe other nodes to evaluate the usefulness of with their neighbors in the overlay. However, multicast usually provides data routing while opportunistic overlays support content-based routing. That brokers need to process events distinguishes our system from multicast systems where nodes perform data routing and participate as relays. That content-based routing needs to manage consumer's subscription in addition to forwarding data, makes dynamic reconfiguration in content-based systems more complex than in multicast systems. In addition, tree-based multicast system have robustness problems.

The most related work in the context of application-level multicast appears in [3], which presents a similar model and approach to ours. The overlay multicast network infrastructure (OMNI) consists of a set of devices called Multicast Service Nodes (MSNs) and provides data distribution services to a set of end-hosts. An end-host subscribes with a single MSN to receive multicast data service. Here, MSNs are similar to brokers and end-hosts are similar to producers/consumers. Similarly, the overlay tree is iteratively modified to adapt with changing distribution of MSNs, clients, as well as network conditions. However, the multicast overlay network is modeled as a complete graph. As shown in [1], a full connected overlay network cannot scale up to a large scale. Similar models of overlay multicast have also been proposed in scattercast [13] and overlay multicast networks [97]. All of these systems has been designed for fixed networks and do not present solutions how to handle the highly dynamic changes in mobile environments.

7.3.2 Multicast in Mobile Ad Hoc Networks

Many multicast routing protocols have been proposed for mobile ad hoc networks [60, 34, 18, 4, 14, 50, 63, 40]. These protocols can classified into three categories: tree-based approaches (e.g., MAODV,LGT), mesh-based [60, 34, 18] and stateless multicast [50]. The tree-based approaches provide high data forwarding efficiency at the expense of low robustness and are not necessarily best suited for multicast in a mobile ad hoc networks where network topology changes frequently. The mesh-based approaches can provide multiple paths between any source and receiver pair. Both tree- and mesh-based approaches have an overhead of creating and maintaining the delivery tree/mesh with time. Stateless multicast minimizes the overhead by explicitly specifying the list of destination addresses in the packet header.

The work closely related to the opportunistic overlays is the mesh-based stateless multicast protocols, including AMRoute [63] and PAST-DM [40]. Both protocols build a virtual network on top of virtual mesh consisting of only member nodes on top of physical wireless networks and then use a subset of virtual links to generate multicast tree. These work has resulted in more stable protocol operation and low control overhead. However, AMRoute uses a static virtual mesh and has low efficiency due to the increasingly mismatching between static virtual mesh and the underlying physical network topology. PAST-DM aims to improve the efficiency and reduce the latency by progressively dynamically adapting the virtual mesh to changes in the physical network topology. Although opportunistic overlays use a similar dynamic virtual overlay construction technique as PAST-DM, the dynamic routing path in opportunistic approach involves not only path changes in event routing, but also subscription code relocation, which makes the existing dynamic delivery technique in PAST-DM is not readily applicable to our system. In addition, the processing of events will consume a broker's computational resources, which implies that brokers' computational capabilities need to be taken into account. Finally, in PAST-DM, all member nodes are considered to be equivalent peers and participate in overlay routing. In contrast, opportunistic overlays conceptually divide nodes into brokers which are organized into an overlay broker network, and clients(producers/consumers) which send/receive events via the broker network. This model is more suitable for content-based routing since overlay routing through 'thin' nodes with very limited resources will present burden on such nodes and may result in inefficiency of content delivery in mobile systems. Dynamic reconfiguration using by opportunistic overlays adapts both the overlay broker network and the connections between brokers and clients.

7.4 General Overlay Networks

There has been a large amount of work in the area of overlay network. We discuss those work which is closely related to opportunistic overlay presented in this thesis.

7.4.1 Overlay Network Construction

As shown in [61], the overlay topology has significant impact on the overlay routing performance. Several approaches to construct efficient overlay network topology have been proposed [61, 69, 90]. [90] presents a binning scheme whereby nodes partition themselves into bins such as nodes that fall within a given bin are relatively close to one another in terms of network latency. The strategy is then applied to topology-aware construction of overlay network and topology-aware server selection. A routing underly sitting between overlay networks and the underlying physical network is proposed in [69]. The underlay provides a set of basic primitives to help the overlay to probe the underlying physical network information. The proposed services include connectivity information about the Internet, routing path, disjoin paths, nearest neighbor and etc. GNP [70] describes an approach for a global network distance estimation. The above research work can benefit opportunistic overlays in several aspects, e.g., achieving accurate information about underlying network information such as connectivity and latency between any pair of nodes, building efficient overlay broker network, finding nearest brokers, and finding good routing paths.

7.4.2 Resilient Overlay Networks

Resilient Overlay Networks (RON) [1] at MIT optimize application-specific routing metrics, by monitoring the functioning and quality of network paths. Resilient overlay networks provides an application-layer overlay on top of the existing Internet routing substrate, which allows distributed Internet applications to quickly detect and recover from path outrages and periods of degraded performance. Both opportunistic overlays and resilient overlay networks optimize application-specific routing metrics by monitoring changes in the physical network and dynamic adjusting overly routing accordingly. However, resilient overlay networks have been designed for efficient fault detection and recovery while the opportunistic overlays focus on better end-to-end performance, and do not address the issues of failure in the underlying physical networks. Resilient overlay networks' routing mechanism detects, recovers and routes around of path outrages of the Internet by dynamically choosing routing paths while opportunistic overlays first reconfigure overlay broker network topology based on changes in the underlying wireless network topology, and then perform dynamic routing on the new overlay networking. In addition, a resilient overlay network is organized as a fully connected graph while the opportunistic overlay approach does not assume any specific overlay broker network topology. Finally, a RON's node performs data routing without involving any data processing, while opportunistic overlays support content-based routing and a broker in opportunistic overlays must perform data process in addition to data relay.

7.4.3 Service Service Networks

Service overlay networks are similar to content delivery overlay in that nodes in overlay service provide not only application-level routing, but also value-added service [52, 62, 51, 39]. QOS-aware service overlay routing (service composition) [62, 51, 39] dynamically constructs QoS-satisfied service paths based on changes in network condition and service nodes' computation capacity. This is similar to dynamic routing in opportunistic overlays. However, unlike a service function in overlay service, which is installed statically on fix service node, a modulator in opportunistic overlays is dynamically deployed when a consumer subscribes with the event system. Dynamical install of modulator implies dynamic routing involves not only event routing path change, but also subscription code relocation, which makes existing dynamic overlay path change technique in overlay service is not readily applicable in our system. At the same time, dynamic modulator location provides the new optimization opportunity such as broker computational load balancing by optimizing the placement of modulators.

7.4.4 Peer-to-Peer Systems

Research on peer-to-peer systems has result in many successful systems and applications. Most peer-to-peer systems are implemented as application-level overlay networks. Early peer-to-peer systems are unstructured. More recent work on peer-to-peer networks has resulted a new class of structured peer-to-peer networks. Most structured peer-to-peer systems have been built using Distributed Hash Table (DHT), including Chord [98], Pastry [91] and CAN [87], Tapestry [112]. Reconfiguration has been widely studied in these systems. Their self-reconfiguration capability is exploited for fault-tolerant routing and for adjusting routing paths with respect to metrics of the underlying network. These schemes however try to improve the selection of paths on an existing overlays. Our work in this thesis, by contrast, tries to improve both the structure of the overlay itself and the routing paths on the overlay. Furthermore, none of them has addressed the issues of dynamic reconfiguration in a highly dynamic environments, such as mobile ad hoc networks. The notion of dynamic routing and dynamic reconfiguration of dissemination structure required to handle changes in nodes' location and physical network topology in mobile environments, especially in mobile ad hoc networks, is probably more than what can be offered by current peer-to-peer networks.

7.5 Wireless Network Routing Protocols

Wireless networks can be classified in two types: infrastructure-based network and infrastructure less (ad hoc) networks. Infrastructure based network consists of a network with fixed and wired base stations. A mobile host communicates with a base station in the network within its communication range. The mobile node can move geographically while it is communicating. When it goes out of range of one base station, it connects with new base station and starts communicating through it. In contrast to infrastructure based networks, in ad hoc networks, all nodes are mobile and from an on-the fly communication network in an arbitrary manner. Each node in such network behaves as routers and takes part in discovery and maintenance of router to other nodes. Paths between each pair of mobiles nodes consist of one or multiple hops. Many routing protocols have been proposed for both infrastructure-based and ad hoc wireless networks. This section examines those network routing protocols and discusses the relationships between them and opportunistic overlay routing.

7.5.1 Mobile IP and its Location Management

The Mobile IP protocol was designed to support seamless and continuous Internet connectivity for mobile computing devices. Mobile IP [76, 54] enables a mobile node to stay connected when moving across network boundaries without changing its IP address. The fundamental problem addressed by Mobile IP is how to supply a user with a stable IP address as the user connects to different access points in an IP network. In Mobile IP, each mobile node uses two IP addresses: a static home address and a dynamic care-of address which indicates the mobile node current point of attachment. Whenever the mobile node is not attached to its home network (and is therefore attached to what is termed a foreign network), the home agent (the access point which the mobile originally attached to) gets all the packets destined for the mobile node and forwards them to the mobile node's current point of attachment. Route optimization for Mobile IP delivers packets directly to the mobile node's care-of address without any assistance from the home agent once a correspondent node knows the mobile node's current care-of address. Route optimization can dramatically improve performance for Mobile IP routing. Opportunistic overlays' dynamic home broker change is similar to Mobile IP's optimization of location management [57] in that both attempt to optimize event routing paths after a mobile host changes its location, by changing the 'triangle' delivery to direct delivery. Essentially, dynamic home broker change in opportunistic overlays is an analogue of optimized Mobile IP routing at application layer, though modulator handoff is somewhat more complex than link handoff, especially for stateful modulators. Since opportunistic overlays are realized on top of Mobile IP, its performance benefits and overheads depend on the underlying Mobile IP implementation.

7.5.2 Mobile Ad Hoc Network Routing Protocols

Routing protocols for mobile ad hoc networks can be divided into two categories: proactive (table-driven) and reactive (on-demand) based on when and how the routes are discovered.

In proactive routing protocols, each node maintains up-to-date routing information to other nodes in the network. This can be achieved by exchanging routing update message among node periodically or when the network topology changes. Well known proactive routing protocols include Dynamic Destination Sequenced Distance Vector (DSDV) [77], Global State routing (GSR) [15], Fisheye State Routing (FSR) [74] and Hierarchical State Routing [47]. In on demand routing protocols, the routing information is created as and when required. When a source wants to send a packet, it initiates the routing discovery mechanisms to find the path to the destination. Among on-demand protocols are Ad Hoc On-demand Distance Vector Routing (AODV) [78], Dynamic Source Routing Protocols (DSR) [56], Temporally Ordered Routing Algorithm (TORA) [73], Associativity Based Routing (ABR) [104] and Signal Stability Routing (SSR) [24].

The current implementation of opportunistic overlays adopts the routing algorithm used in global state routing (GSR) [15]. Both take the ideas of link state routing but avoid flooding of routing messages by propagating updated information to its neighbors only during each update period. However, opportunistic overlay routing has to deal with modulators, which means that a routing path change involves not only changing the routing table but also modulator relocation. Moreover, opportunistic overlay routing addresses the issue of nodes overloads, which GSR and most other routing protocols do not deal with. Though the current implementation of opportunistic overlays adopts a similar mechanism as the one used in global state routing, the opportunistic approach itself does not depend on any specific routing protocol and can be implemented using other routing protocols' algorithms. For example, fish eye routing (FSR) [74] is an improvement of GSR by reducing the size of update messages. Opportunistic overlays can use the idea behind FSR to improve overlay brokers routing. We leave this as future work.

7.6 Adaptations in Mobile Environments

In addition to the mobile routing protocols mentioned above, several network-level efforts address mobility. Berkeley's BARWAN project [58] provides seamless roaming across heterogeneous networks. Furthermore, BARWAN enables data forms to be changed to suit end system or wireless network limitations, by permitting application-level, type-specific data transformation and data compression[58]. Zhao, Castelluccia and Baker [113] develop a general-purpose mechanism at the network level, which supports multiple packet delivery methods and multiple network interfaces, and the system adaptively selects the most appropriate method and interface. Similarly, the CMU Monarch project aims to enable adaptive mobile host communications, to make the most efficient use of the best network connectivity available to the mobile host at any time [55]. [46] dynamically reconfigures a mobile system in response to changes in link-layer environment.

At the transport layer, MSocks provides transport layer mobility, and it allows a mobile host to change its point of attachment as well as control which network interfaces [64] are being used. Finally, there are several TCP enhancements for wireless networks [12]. Daedulus has developed a snoop protocol that provides significantly better TCP performance over lossy wireless networks [2].

Our work focuses on middleware-level and application level adaptations, and it can benefit from the above network research. Since such adaptations require information about network-level changes, it can benefit from the substantial ongoing work on real-time network monitoring, including the work performed in the Monarch project [55].

At application-level, the Odyssey projects extends the Unix System call interface to support flexible application-aware adaptations [71], as also done in our own work addressing interactive applications [86]. The system monitors resource levels, notifies applications of relevant changes, and enforces resource allocation decision. Each application independently decides how best to adapt when notified. This is similar to JECho's adaptations where an application can be notified of resource changes and responds to such changes according to its adaptation strategy defined in a modulator.

Some adaptations based on application semantics can be provided only at application level. For example, datatype-specific data transformation and data compression must depend on the application. HRL's Intelligent information dissemination services use bandwidth-aware filtering to adapt information streams to resource bandwidth availability [96]. Our own previous work with JECho has addressed these problems by supporting general application level adaptations via the dynamic deployment and partitioning of event modulators [116]. Opportunistic overlays, as well as our related research on coordinating network- with application-level adaptations [42] complement these efforts, by providing models and mechanisms for linking network- or broker-level changes in resource availability with suitable application- and middleware-level adaptations.

CHAPTER VIII

CONCLUSIONS AND FUTURE WORK

8.1 Conclusions

Middleware has become a key enabler for the development of distributed applications. Unfortunately, conventional middleware technologies do not yet offer sufficient functionality to make them suitable for mobile environments. This dissertation proposes a novel middleware approach and its dynamically reconfigurable support framework for building efficient mobile applications. Specifically, we address the inefficiency of content delivery introduced by node mobility and by dynamically changing system loads, in the context of publish/subscribe systems.

In response to changes in physical network topology and to node mobility, opportunistic overlays dynamically change broker network topology, clients' assignments to brokers, and event delivery paths, with the goal of optimizing end-to-end delays in event delivery. Essentially, opportunistic overlays implement a middleware-level analogue of the dynamic routing protocols used in wireless communications [76, 77, 78, 56]. More importantly, by coordinating network- with middleware-level routing, opportunistic overlays can attain substantial performance improvements over non-adaptive event systems. Such improvements are due to their use of shorter network paths and the better balancing of loads across event brokers.

Opportunistic overlays and the adaptive methods they use are realized by a set of distributed protocols implemented in a Java-based publish/subscribe infrastructure. Opportunistic overlays are prototyped with the JECho pub/sub system [116]. The performance evaluations reported in this dissertation are performed both on actual hardware, to assess basic performance properties and penalties, and via emulation and simulation, to assess the effects of mobility and to better understand the scalability of our approach.

The opportunistic overlay approach presented in this dissertation can be summarized as follows:

- **Generality.** The opportunistic overlay approach adopts a general overlay network approach. It handles mobility transparently in both infrastructure-based mobile environments and mobile ad hoc networks in a consistent way. In addition, traditional publish/subscribe systems are usually built on top of broker overlay networks. Opportunistic overlays maintain the publish/subscribe's model and semantics, making it easy to extend existing publish/subscribe systems and applications to mobile environments.

  Applicability. The opportunistic overlay approach is practically applicable in both infrastructure-based and ad hoc mobile environments. Opportunistic overlays are realized by a set of distributed adaptation protocols executed by each broker/client without any central control. Experimental results show that the overheads of dynamic adaptation are moderate and controllable. Moreover, the protocols do not assume any specific broker network topology and do not rely on any specific properties of the underlying physical network so that they can operate on top of different broker network topologies and across heterogeneous systems using multiple underlying routing protocols. Our experiments include running a virtual workbench application and and a flood watch application in real wireless networks.

- **Efficiency.** Comprehensive performance evaluations, including simulation, emulation, and running representative applications on actual networks demonstrate that the opportunistic overlay approach can significantly improve event delivery delays compared to static approaches, in both infrastructure-based and ad hoc mobile environments.

- **Extensibility.** The opportunistic overlay software framework adopts a layered architecture and event-driven paradigm. The architecture is able to support flexible adaptation implementations, and the architecture itself is easy to reconfigure and extend. It is easy for different brokers to define different adaptations based on their capabilities and applications' requirements. Further, the framework provides flexible support for the reconfiguration and extension of the system with new adaptations.

8.2 Future Work

Future work should address some deficiencies of our current implementation, as well as generalize upon the basic concept of opportunistic overlays.

- **Application-specific failure recovery of broker overlays.** Our current implementation assumes a reliable network environment and therefore, does not consider dynamic disconnection, reconnection, and network partition. Future work will add application-specific failure recovery to broker overlays, an example being the mirroring and failure recovery described for business transaction systems in [35].

- **Optimization of performance metrics other than end-to-end latency.** There is a wide range of optimizations and adaptations to be explored for opportunistic overlays' event delivery methods. We will extend the opportunistic approach to optimize performance metrics other than end-to-end latency, including network bandwidth, power usage [85], and specific application needs [84]. We may also explore optimizing multi-dimensional performance metrics.

- **Dynamic optimization of modulator placement based on modulator semantics.** The location of a modulator has an impact on performance. For example, for a 'contract' modulator that typically reduces event size or filters events, placing it at a broker closer to a source broker can reduce network overheads. Toward this end, we will investigate the dynamic optimization of modulator placement based on modulator semantics, network condition and broker state.

- **More comprehensive simulation model.** Our simulations do not consider link layer details, such as MAC-layer protocol functionality, link errors, or multiple-access interference. They do not take into account physical layer characteristics (e.g., radio properties), either. As a result, we cannot model actual control overheads or the effects of link contention like increased message delay due to packet losses. Future work should address these limitations by constructing a more comprehensive simulator with a MAC layer model (e.g., IEEE 802.11 MAC).

- **Integration with wireless routing implementations.** A final topic of interest to us is the integration of the opportunistic overlay prototype with actual implementations of wireless network routing protocols as well as distributed resource discovery and resource management services.

REFERENCES

[1] Andersen, D. G., Balakrishnan, H., Kaashoek, M. F., and Morris, R., "Resilient overlay networks," in *Proceedings of Symposium on Operating Systems Principles (SOSP 2001)*, pp. 131–145, 2001.

[2] Balakrishnan, H., Seshan, S., Amir, E., and Katz, R. H., "Improving tcp/ip performance over wireless networks," in *Proceeding of 1st ACM Int's Conf. On Mobile Computing and Networking (Mobicom 1995)*, November 1995.

[3] Banerjee, S., Kommareddy, C., Kar, K., Bhattacharjee, S., and Khuller, S., "Construction of an efficient overlay multicast infrastructure for real-time applications.," in *Proceedings of INFOCOM 2003*, 2003.

[4] Belding-Royer, E. M. and Perkins, C. E., "Multicast operation of the ad-hoc on-demand distance vector routing protocol.," in *Proceedings of MOBICOM 1999*, pp. 207–218, 1999.

[5] Bhola, S., Zhao, Y., and Auerbach, J., "Scalably supporting durable subscriptions in a publish/subscribe system," in *Proceedings of the International Conference on Dependable Systems and Networks (DSN 2003)*, pp. 57–66, June 2003.

[6] Bhola, S., Strom, R., Bagchi, S., Zhao, Y., and Auerbach, J., "Exactly-once delivery in a content-based publish-subscribe system," in *Proceedings of the International Conference on Dependable Systems and Networks (DSN'02)*, IEEE Press, Jun 2002.

[7] Bustamante, F. E., Eisenhauer, G., Schwan, K., and Widener, P., "Efficient wire formats for high performance computing.," in *Proceedings of supercomputing 2000*, 2000.

[8] Caporuscio, M., Inverardi, P., and Pelliccione, P., "Formal analysis of clients mobility in the siena publish/subscribe middleware," tech. rep., Department of Computer Science, University of L'Aquila, 2002.

[9] Carzaniga, A., Rosenblum, D. S., and Wolf, A. L., "Design of a scalable event notification service: Interface and architecture," Tech. Rep. CU-CS-863-98, Department of Computer Science, Univ. of Colorado at Boulder, USA, 1998.

[10] Carzaniga, A., Rosenblum, D. S., and Wolf, A. L., "Achieving scalability and expressiveness in an internet-scale event notification service," in *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing (PODC 2000)*, (Portland, Oregon), pp. 219–227, July 2000.

[11] Carzaniga, A., Rosenblum, D. S., and Wolf, A. L., "Design and evaluation of a wide-area event notification service," *ACM Transactions on Computer Systems*, vol. 19, no. 3, pp. 332–383, 2001.

[12] Chan, M. C. and Ramjee, R., "Tcp/ip performance over 3g wireless links with rate and delay variation," in *Proceedings of The Eighth ACM International Conference on Mobile Computing and Networking (MobiCom 2002)*, (Atlanta, GA), September 2002.

[13] Chawathe, Y., "Scattercast: an adaptable broadcast distribution framework.," *Multimedia System*, vol. 9, no. 1, pp. 104–118, 2003.

[14] Chen, K. and Nahrstedt, K., "Effective location-guided tree construction algorithms for small group multicast in manet.," in *Proceedings of INFOCOM 2002*, 2002.

[15] Chen, T.-W. and Gerla, M., "Global state routing: A new routing scheme for ad-hoc wireless networks," in *Proceedings of IEEE ICC'98*, June 1998.

[16] Chen, Y., Schwan, K., and Rosen, D. W., "Java mirrors: Building blocks for remote interaction," in *Proceedings of the 2002 International Parallel and Distributed Processing Symposium (IPDPS 2002)*, April 2002.

[17] Chen, Y., Schwan, K., and Zhou, D., "Opportunistic channels: Mobility-aware event delivery," in *ACM/IFIP/USENIX International Middleware Conference (Middleware 2003)*, pp. 182–201, 2003.

[18] Chiang, C.-C., Gerla, M., and Zhang, L., "Forwarding group multicast protocol (FGMP) for multihop, mobile wireless networks," *Cluster Computing*, vol. 1, no. 2, pp. 187–196, 1998.

[19] Chu, Y.-H., Rao, S. G., and Zhang, H., "A case for end system multicast.," in *Proceedings of SIGMETRICS 2000*, pp. 1–12, 2000.

[20] Chu, Y., Rao, S. G., Seshan, S., and Zhang, H., "Enabling conferencing applications on the internet using an overlay multicast architecture.," in *Proceedings of SIGCOMM 2001*, pp. 55–67, 2001.

[21] Cugola, G., Nitto, E. D., and Picco, G. P., "Content-based dispatching in a mobile environment," in *Proceedings of WSDAAL 2000*, 2000.

[22] Cugola, G., Nitto, E. D., and Fuggetta, A., "The "jedi" event-based infrastructure and its application to the development of the opss wfms," in *IEEE Transactions on Software Engineering in 2001*, 2001.

[23] Cugola, G. and Jacobsen, H.-A., "Using Publish/Subscribe Middleware for Mobile Systems," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 6, no. 4, pp. 25–33, 2002.

[24] Dube, R., Rais, C., Wang, K., and Tripathi, S., "Signal stability based adaptive routing (ssa) for ad hoc mobile networks," pp. 36–45, 1997.

[25] Eisenhauer, G., Bustamante, F. E., and Schwan, K., "Event services in high performance systems.," *Cluster Computing*, vol. 4, no. 3, pp. 243–252, 2001.

[26] Eisenhauer, G., Bustamente, F., and Schwan, K., "Event services for high performance computing," in *Proceedings of the Ninth IEEE International Symposium on High Performance Distributed Computing (HPDC'2000)*, August 2000.

[27] Fiege, L., Gärtner, F. C., Kasten, O., and Zeidler, A., "Supporting mobility in content-based publish/subscribe middleware," in *ACM/IFIP/USENIX International Middleware Conference (Middleware 2003)*, pp. 103–122, 2003.

[28] Fiege, L., Mühl, G., and Buchmann, A., "An architectural framework for electronic commerce applications," in *Proceedings of Informatik 2001: Annual Conference of the German Computer Society*, 2001.

[29] Fiege, L., Mühl, G., and Gärtner, F. C., "A modular approach to build structured event-based systems," in *Proceedings of the 2002 ACM Symposium on Applied Computing (SAC'02)*, pp. 385–392, 2002.

[30] Fiege, L., Mühl, G., and Gärtner, F. C., "A modular approach to build structured event-based systems," in *Proceedings of the 2002 ACM Symposium on Applied Computing (SAC'02)*, (Madrid, Spain), pp. 385–392, ACM Press, 2002.

[31] Fiege, L., Mühl, G., and Gärtner, F. C., "Modular event-based systems," *The Knowledge Engineering Review*, vol. 17, no. 4, pp. 359–388, 2003.

[32] "Environmental hydrologists workbench." http://access.ncsa.uiuc.edu/Stories/SC98-live/demoabstractsfull.html.

[33] Francis, P., "Yoid: Extending the internet multicast architecture." http://www.icir.org/yoid/docs/index.html, April 2000.

[34] Garcia-Luna-Aceves, J. J. and Madruga, E. L., "A multicast routing protocol for ad-hoc networks.," in *Proceedings of INFOCOM 1999*, pp. 784–792, 1999.

[35] Gavrilovska, A., Schwan, K., and Oleson, V., "A practical approach for 'zero' downtime in operational information syste ms," in *Proceedings of the 22nd International Conference on Distributed Compu ting Systems (ICDCS-22)*, July 2002.

[36] Gavrilovska, A., Schwan, K., and Oleson, V., "A practical approach for 'zero' downtime in operational information systems," in *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS-22)*, July 2002.

[37] "Mobiwan: A ns-2.1b6 simulation platform for mobile ipv6 in wide area networks." http://www.inrialpes.fr/planete/mobiwan/Documents/mobiwan-report-0501.pdf.

[38] Gruber, R., Krishnamurthy, B., and Panagos, E., "The architecture of the READY event notification service," in *Proceedings of the 19th IEEE International Conference on Distributed Computing Systems Middleware Workshop*, (Austin, Texas, USA), May 1999.

[39] Gu, X., Nahrstedt, K., Chang, R., and Ward, C., "Qos-assured service composition in managed service overlay networks," in *Proceedings of IEEE International Conference on Distributed Computing Systems (ICDCS2003)*, May 2003.

[40] Gui, C. and Mohapatra, P., "Efficient overlay multicast for mobile ad hoc networks," in *Proceedings of IEEE Wireless Communications and Networking Conference*, March 2003.

[41] Hauspie, M., Simplot, D., and Carle, J., "Partition detection in mobile ad-hoc networks," in *Proceedings of 2nd IFIP Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET 2003*, 2003.

[42] He, Q. and Schwan, K., "Iq-rudp: Coordinating application adaptation with network transport," in *Proceedings of High Performance Distributed Computing-9 (HPDC-9)*, July 2002.

[43] Huang, Y. and Garcia-Molina, H., "Publish/subscribe in a mobile environment," in *2nd ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE'01)*, (Santa Barbara, California, USA), 2001.

[44] Huang, Y. and Garcia-Molina, H., "Publish/subscribe in a mobile environment," in *Proceedings of the 2nd ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE'01)*, pp. 27–34, 2001.

[45] Huang, Y. and Garcia-Molina, H., "Publish/subscribe tree construction in wireless ad-hoc networks," in *Proceedings of the 4th International Conference on Mobile Data Management(MDM 2003)*, pp. 122–140, 2003.

[46] Inouye, J., Bikley, J., and Walpole, J., "Dynamic network reconfiguration support for mobile computers," in *Proceedings of The Third ACM International Conference on Mobile Computing and Networking (MobiCom 1997)*, September 1997.

[47] Iwata, A., Chiang, C., Pei, G., Gerla, M., and Chen, T., "Scalable routing strategies for ad-hoc wireless networks," August 1999.

[48] Jannotti, J., Gifford, D. K., Johnson, K. L., Kaashoek, M. F., and Jr., J. O., "Overcast: Reliable multicasting with an overlay network.," in *Proceedings of OSDI 2000*, pp. 197–212, 2000.

[49] "Uob-jadhoc aodv implementation, rfc 3561." http://www.aodv.org/, 2004.

[50] Ji, L. and Corson, M. S., "Differential destination multicast - a manet multicast routing protocol for small groups.," in *Proceedings of INFOCOM 2001*, pp. 1192–1202, 2001.

[51] Jin, J. and Nahrstedt, K., "Qos service routing for supporting multimedia applications," Tech. Rep. UIUCDCS-R-2002-2303/UILU-ENG-2002-1746, University of Illinois at Urbana-Champaign, 2002.

[52] Jin, J. and Nahrstedt, K., "Large-scale service overlay networking with distance-based clustering," in *ACM/IFIP/USENIX International Middleware Conference (Middleware 2003)*, 2003.

[53] Jin, Y. and Strom, R. E., "Relational subscription middleware for internet-scale publish-subscribe.," in *Proceedings of the 2nd International Workshop on Distributed Event-based Systems(DEBS'03)*, June 2003.

[54] Johnson, D. and Perkins, C., *Mobility Support in IPv6.* Internet Draft, IETF, draft-ietf-mobileip-ipv6-12.txt(work in progress), September 2000.

[55] Johnson, D. B. and Maltz, D. A., "Protocols for adaptive wireless and mobile networking," *IEEE Personal Communications*, vol. 3, no. 1, pp. 34–42, 1995.

[56] Johnson, D. B. and Maltz, D. A., "Dynamic source routing in ad hoc wireless networks," *Mobile Computing*, vol. 353, 1996.

[57] Johnson, D. B. and Perkins, C., *Route Optimization in Mobile IP.* In Internet Draft(work in progress), 1998.

[58] Katz, R. and Brewer, E., "The case for wireless overlay networks," in *Proceedings of SPIE Multimedia and Networking Conference 1996*, January 1996.

[59] Kephart, J. O. and Chess, D. M., "The vision of autonomic computing," *IEEE Computer Magazine*, January 2003.

[60] Lee, S.-J., Su, W., and Gerla, M., "On-demand multicast routing protocol in multihop wireless mobile networks.," *MONET*, vol. 7, no. 6, pp. 441–453, 2002.

[61] Li, Z. and Mohapatra, P., "The impact of topology on overlay routing," in *Proceedings of the 23rd Conference of the IEEE Communications Society (INFOCOM 2004)*, March 2004.

[62] Li, Z. and Mohapatra, P., "Qron: Qos-aware routing in overlay networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, January 2004.

[63] Liu, M., Talpade, R. R., and McAuley, A., "AMRoute: Adhoc Multicast Routing Protocol," Tech. Rep. 99, The Institute for Systems Research, University of Maryland, 1999.

[64] Maltz, D. A. and Bhagwat, P., "Msocks: An architecture for transport layer mobility," in *IEEE INFOCOM'98*, March 1998.

[65] Mao, Y., deBruijn, W., Knutsson, B., Lu, H., and Smith, J. M., "Dharma: Distributed home agent for robust mobile access," in *Proceedings of the 19th ACM Symposium on Operating Systems Principles(SOSP 2003), WIP Session*, October 2003.

[66] Medina, A., Lakhina, A., Matta, I., and Byers, J., "Brite: An approach to universal topology generation," in *Proceedings of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems- MASCOTS '01*, (Cincinnati, Ohio), August 2001.

[67] Meier, R. and Cahill, V., "Steam: Event-based middleware for wireless ad hoc networks," in *In Proceedings of the 1st International Workshop on Distributed Event-Based Systems (DEBS'02)*, 2002.

[68] Mühl, G., Fiege, L., Gärtner, F. C., and Buchmann, A. P., "Evaluating advanced routing algorithms for content-based publish/subscribe systems," in *Proceedings of The Tenth IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2002)* (Boukerche, A., Das, S. K., and Majumdar, S., eds.), (Fort Worth, TX, USA), pp. 167–176, IEEE Press, October 2002.

[69] Nakao, A., Peterson, L., and Bavier, A., "A routing underlay for overlay networks," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 11–18, ACM Press, 2003.

[70] Ng, T. S. E. and Zhang, H., "Predicting internet network distance with coordinates-based approaches," in *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies(INFOCOM 2001)*, April 2002.

[71] Noble, B. D., Satyanarayanan, M., Narayannan, D., Tilton, J., Flinn, J., and Walker, K., "Agile application-aware adaptation for mobility," in *Proceedings of the 16th ACM Symposium on Operating Systems Principles (SOSP-16)*, October 1997.

[72] Oki, B. M., Pflügl, M., Siegel, A., and Skeen, D., "The information bus - an architecture for extensible distributed systems," in *Proceedings of the Fourteenth ACM Symposium on Operating System Principles(SOSP 1993)*, pp. 58–68, 1993.

[73] Park, V. D. and Corson, M. S., "A highly adaptive distributed routing algorithm for mobile wireless networks.," in *Proceedings of INFOCOM 1997*, pp. 1405–1413, 1997.

[74] Pei, G., Gerla, M., and Chen, T.-W., "Fisheye state routing in mobile ad hoc networks," in *Proceedings of ICDCS Workshop on Wireless Networks and Mobile Computing*, pp. D71–D78, 2000.

[75] Pendarakis, D., Shi, S., Verma, D., and Waldvogel, M., "ALMI: An application level multicast infrastructure," in *Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems (USITS '01)*, (San Francisco, CA, USA), pp. 49–60, 2001.

[76] Perkins, C., *IP Mobility Support.* IETF, Request for Comments 2002, Oct., 1996.

[77] Perkins, C. and Bhagwat, P., "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," in *Proceedings of ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, pp. 234–244, 1994.

[78] Perkins, C. E., Belding-Royer, E. M., and Das, S. R., "Ad hoc On-Demand Distance Vector (AODV) Routing," July 2003. RFC 3561.

[79] Picco, G. P., Cugola, G., and Murphy, A. L., "Efficient content-based event dispatching in the presence of topological reconfiguration," in *Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCS 03)*, pp. 234–243, 2003.

[80] Pietzuch, P. R., *Hermes: A Scalable Event-Based Middleware.* PhD thesis, University of Cambridge, February 2004.

[81] Pietzuch, P. R. and Bacon, J., "Peer-to-peer overlay broker networks in an event-based middleware," in *Proceedings of the 2nd International Workshop on Distributed Event-Based Systems (DEBS'03)*, June 2003.

[82] Pietzuch, P. R. and Bacon, J., "Peer-to-peer overlay broker networks in an event-based middleware," in *Proceedings of the 2nd International Workshop on Distributed Event-Based Systems (DEBS'03)*, June 2003.

[83] *Planetlab.* http://www.planet-lab.org.

[84] Poellabauer, C. and Schwan, K., "Kernel support for the event-based cooperation of distributed resource ma nagers," in *Proceedings of the 8th IEEE Real-Time and Embedded Technology and App lications Symposium (RTAS 2002)*, September 2002.

[85] Poellabauer, C. and Schwan, K., "Power-aware video decoding using real-time event handlers," in *Proceedings of the 5th International Workshop on Wireless Mobile Mult imedia (WoWMoM)*, September 2002.

[86] Poellabauer, C., Schwan, K., and West, R., "Coordinated cpu and event scheduling for distributed multimedia applications," in *Proceedings of the 9th ACM Multimedia Conference*, October 2001.

[87] Ratnasamy, S., Francis, P., Handley, M., Karp, R. M., and Shenker, S., "A scalable content-addressable network.," in *Proceedings of SIGCOMM 2001*, pp. 161–172, 2001.

[88] Ratnasamy, S., Handley, M., Karp, R., and Shenker, S., "Application-level multicast using content-addressable networks," in *Proceedings of the Third International COST264 Workshop (NGC 2001)* (Crowcroft, J. and Hofmann, M., eds.), no. LNCS 2233, (London, UK), pp. 14–29, Springer-Verlag, Nov. 2001.

[89] Ratnasamy, S., Handley, M., Karp, R. M., and Shenker, S., "Topologically-aware overlay construction and server selection," in *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies(INFOCOM 2002)*, June 2002.

[90] Ratnasamy, S., Handley, M., Karp, R. M., and Shenker, S., "Topologically-aware overlay construction and server selection.," in *Proceedings of INFOCOM 2002*, 2002.

[91] Rowstron, A. and Druschel, P., "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," in *Proceedings of the 4th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2003)*, December 2003.

[92] Rowstron, A., Kermarrec, A.-M., Castro, M., and Druschel, P., "Scribe: The design of a large-scale event notification infrastructure," in *Proceedings of the Third International Workshop on Networked Group Communications (NGC2001)*, (London, UK), November 2001.

[93] Segall, B. and Arnold, D., "Elvin has left the building: A publish/subscribe notification service with quenching," in *Proceedings of A UUG97*, September 1997.

[94] Segall, B., Arnold, D., Boot, J., Henderson, M., and Phelps, T., "Content based routing with elvin4," in *Proceedings of AUUG2K*, (Canberra, Australia), June 2000.

[95] Segall, W. and Arnold, D., "Elvin has left the building: A publish/subscribe notification service with quenching," in *Proceedings of the 1997 Australian UNIX Users Group, Brisbane, Australia, September 1997.*, 1997. http://elvin.dstc.edu.au/doc/papers/auug97/AUUG97.html.

[96] Shek, E. C., Dao, S. K., Zhang, Y., and etc, "Intelligent information dissemination services in hybrid satellite-wireless networks," *ACM Mobile Networks and Applications (MONET) Journal*, vol. 5, December 2000.

[97] Shi, S. and Turner, J. S., "Routing in overlay multicast networks.," in *Proceedings of INFOCOM 2002*, 2002.

[98] Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., and Balakrishnan, H., "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proceedings of ACM SIGCOMM 2001*, August 2001.

[99] Strom, R., Banavar, G., and et.al, "Gryphon: An information flow based approach to message brokering," tech. rep., IBM TJ Watson Research Center, 1998.

[100] Sun Microsystems, *JMS:Java Message Service(JMS).* http://java.sun.com/jms.

[101] Sutton, P., Arkins, R., and Segall, B., "Supporting disconnectedness - transparent information delivery for mobile and invisible computing," in *CCGrid 2001 IEEE International Symposium on Cluster Computing and the Grid*, May 2001.

[102] Terpstra, W. W., Behnel, S., Fiege, L., Zeidler, A., and Buchmann, A. P., "A peer-to-peer approach to content-based publish/subscribe," in *Proceedings of the 2nd International Workshop on Distributed Event-Based Systems (DEBS'03)*, June 2003.

[103] TIBCO Software INC., *TIB/Rendezvous.* http://www.tibco.com/software/enterprise_backbone/rendezvous.jsp.

[104] Toh, C.-K., "Associativity-based routing for ad hoc mobile networks," *Wireless Personal Communications*, vol. 4, pp. 103–139, March 1997.

[105] Triantafillou, P. and Aekaterinidis, I., "Content-based publish-subscribe over structured p2p networks," in *Proceedings of the 3rd International Workshop on Distributed Event-Based Systems (DEBS'04)*, (Edinburgh, Scotland, UK), May 2004.

[106] Virgillito, A., Beraldi, R., and Baldoni, R., "On event routing in content-based publish/subscribe through dynamic networks," in *Proceedings of the Ninth IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS 2003)*, pp. 322–328, IEEE, 2003.

[107] Wheeler, M. F. and etc., "A data intense challenge: The instrumented oilfield of the future," tech. rep.

[108] Wolenetz, M., Kumar, R., Shin, J., and Ramachandran, U., "Middleware guidelines for future sensor networks," in *Proceedings of 1st workshop broadband advanced sensor networks (BASENETS 2004)*, October 2004.

[109] Wolski, R., Spring, N. T., and Hayes, J., "The network weather service: a distributed resource performance forecasting service for metacomputing," *Future Generation Computer Systems*, vol. 15, no. 5–6, pp. 757–768, 1999.

[110] Yoneki, E. and Bacon, J., "An adaptive approach to content-based subscription in mobile ad hoc networks," in *Proceedings of the The First International Workshop on Mobile Peer-to-Peer Computing (MP2P'04)*, pp. 92–97, Mar 2004.

[111] Zhang, Y. and Li, W., "An integrated environment for testing mobile ad-hoc networks," in *Proceedings of the Third ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'02)*, June 2002.

[112] Zhao, B. Y., Huang, L., Stribling, J., Rhea, S. C., Joseph, A. D., and Kubiatowicz, J. D., "Tapestry: A resilient global-scale overlay for service deployment," *IEEE Journal on Selected Areas in Communications*, vol. 22, pp. 41–53, Jan. 2004.

[113] Zhao, X., Castelluccia, C., and Baker, M., "Flexible network support for mobility," in *Proceedings of The Fourth ACM International Conference on Mobile Computing and Networking (MobiCom 1998)*, October 1998.

[114] Zhou, D., *JECho — An Efficient, Customizable, Adaptable Distributed Event System.* PhD thesis, Georgia Institute of Technology, 2002.

[115] Zhou, D., Pande, S., and Schwan, K., "Method partitioning - runtime customization of pervasive programs without design-time application knowledge.," in *Proceedings of ICDCS 2003*, pp. 610–619, 2003.

[116] Zhou, D., Schwan, K., Eisenhauer, G., and Chen, Y., "Supporting distributed high performance application with java event channels," in *Proceedings of the 2001 International Parallel and Distributed Processing Symposium (IPDPS 2001)*, April 2001.

[117] Zhuang, S., Zhao, B., Joseph, A., Katz, R., and Kubiatowicz, J., "Bayeux: An architecture for scalable and fault-tolerant wide area data dissemination," in *Proceedings of the 11th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'01)*, June 2001.

VITA

Yuan Chen (陈 源) was born in Xuanwei, Yunnan, China. He received a B.S. degree in Computer Science from the University of Science and Technology of China (USTC) in 1994. He then attended the Institute of Computing Technology, Chinese Academy of Sciences in Beijing, earning a M.S. degree in Computer Science in 1997. That fall, he began his graduate studies in the College of Computing at the Georgia Institute of Technology. In April 2005, under the supervision of Prof. Karsten Schwan, Yuan completed his Ph.D. dissertation entitled “Opportunistic Overlays: Efficient Content Delivery in Mobile Environments”.