UNIVERSITY OF CALIFORNIA

Los Angeles

Efficient Routing and Quality of Service Support for Ad Hoc Wireless Networks

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Computer Science

by

Tsu-Wei Chen

1998

The dissertation of Tsu-Wei Chen is approved.

Jack Carlyle

Leonard Kleinrock

Mani B. Srivastava

Mario Gerla, Committee Chair

University of California, Los Angeles

1998

To my wife, Hui-Yin

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I am particularly grateful to my advisor and committee chair, Mario Gerla. His support, patience, and guidance made this work possible. Thanks also to all other members in my committee, Jack Carlyle, Leonard Kleinrock, Mani Srivastava, for their suggestions and comments and the time in reviewing this dissertation.

I have benefited a great deal from the stimulating discussion with my former colleagues at Bell Laboratories: Michael Lyu, John Trotter, Cormac Sreenan, Paul Krzyzanowski and Mani Srivastava. They provided me valuable comments on my investigation on QoS in wireless networks. In particular, Michael is the model of ideal researcher whom I always hope to emulate. I also thank him for his continuous encouragement and support in the past two years.

I am also indebted to all the members in the network research lab for making my tenure at UCLA so enjoyable. I would like to recognize the following colleagues: Ilya Slain, Yuri Romanenko, Kyle Bae, Ching-Chuan Chiang, Ronn Ritke, Gary Pei, Eric Wu, Jack Tsai and Chunhung Lin. Also, I would like to thank two undergraduate students: Chia-Yi Wang and Mark Fernie. Especially, Chia-Yi, Mark and Kyle helped the project on multihop wireless testbed, Ilya and Yuri helped the project on the adaptive QoS, and Ronn Ritke have read and commented on the draft version of this dissertation.

Most importantly, I would like to thank my love wife, Hui-Yin. Without her sacrifices, love and moral support, I would definitely not be able to concentrate on my research during my most productive period.

Last but not least, I would like to thank to my parents for all their love and support, and to my brother for always sharing his valuable experiences in the world of engineering.

VITA

1968	Born, Taipei, Taiwan, R.O.C.
1986–1990	B.S. (Computer Science and Information Engineering), National Taiwan University, Taipei, Taiwan, R.O.C.
1990–1992	Computer Engineer, The Missile Base Depot, Chinese Army, Taiwan.
1992-1993	M.S. (Computer Science), UCLA.
1994–1996	Teaching Assistant, Computer Science Department, UCLA.
1996	Member of Technical Staff-I, Bell Laboratories, Lucent Technologies, Murray Hill, New Jersey.
1996–present	Research Assistant, Computer Science Department, UCLA.

PUBLICATIONS

Tsu-Wei Chen and Mario Gerla, “Global State Routing: A New Routing Scheme for Ad-hoc Wireless Networks”, in *ICC ’98*, Atlanta, 1998.

Ilya Slain, **Tsu-Wei Chen** and Mario Gerla, “Experiments on QoS Adaptation for Speech Delivery in Wireless Networks”, in UCLA CS Technical Report (#980010), 1998.

Tsu-Wei Chen, Jack Tzu-Chieh Tsai and Mario Gerla, “Performance of QoS routing in a multihop, wireless ATM networks”, in *ICUPC ’97*, San Diego, 1997.

T.-W. Chen, P. Krzyznowsky, M. R. Lyu, C. J. Sreenan and J. A. Trotter, “A VC-Based API for Renegotiable QoS in Wireless ATM Networks”, in *ICUPC ’97*, San Diego, 1997.

T.-W. Chen, P. Krzyznowsky, M. R. Lyu, C. J. Sreenan and J. A. Trotter, “Renegotiable Quality of Service - A New Scheme for Fault Tolerance in Wireless Networks”, in *Proceeding of the 27th FTCS*, Seattle, 1997.

Y.U. Cao, **T.-W. Chen**, M.D. Harris, A. B. Kahng, M. A. Lewis and A.D. Stechert, “A Remote Robotics Laboratory on the Internet”, in INET ’95, Honolulu, 1995.

ABSTRACT OF THE DISSERTATION

Efficient Routing and Quality of Service Support for Ad Hoc Wireless Networks

by

Tsu-Wei Chen

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 1998

Professor Mario Gerla, Chair

One feature that distinguishes the ad hoc wireless network from traditional wired networks and PCS (personal communication network) is that all hosts in an ad hoc wireless network are allowed to move freely without the need for static access points. This distinct feature, however, presents a great challenge to the design of the routing scheme and the support of multimedia services, since the link quality and the network topology may be fast changing as hosts roam around.

In this dissertation, we investigate the behaviors of existing routing algorithms. None of them satisfies the stringent requirements of ad hoc wireless networks. These requirements include: high accuracy, low overhead, scalability in a large network, the possibility of providing QoS routing, etc.. Therefore, we propose a new routing approach for ad hoc wireless network: Global State Routing (GSR). Similar to link state routing, GSR maintains a global view of network topology. But unlike link state routing, GSR uses the fisheye technique to keep the control message small, thus reducing the consumption of bandwidth by control overhead. As a result, GSR can be scaled for use in networks with large populations. GSR can also be extended with QoS parame-

ters to perform QoS routing for multimedia applications.

In order to validate the results in our investigation, all schemes are simulated and/or implemented in our testbed at UCLA. The implementation of our testbed also involves the creation of several new features in the operating system to provides QoS support in wireless networks, and the creation of new applications which fully utilize this QoS information. With the integration of these features across various layers, we can realize a truly mobile, multimedia, multihop wireless network.

Efficient Routing and Quality of Service Support for Ad Hoc Wireless Networks

Tsu-Wei Chen

April 28, 1998

CHAPTER 1

Introduction

Research of Quality of Service (QoS), as the name suggests, is to study the level of user satisfaction of the services provided by the communication system. In computer networks, the goal for the QoS support is to achieve a more deterministic communication behavior, so that information carried by the network can be better preserved, and the network resources can be better utilized. This concept is not new. In fact, it has been widely discussed in the context of high speed wired networks, including both datagram and real-time services. The ATM (Asynchronous Transfer Mode) network, for instance, has made a great emphasis on the support of QoS for traffic of different classes. Recently, the Internet, one of the most widely used data networks, also has recognized the importance of QoS in supporting multimedia services. Tremendous efforts have been made by the IETF (Internet Engineering Task Force) to enhance the current Internet.

However, these achievements of supporting QoS in traditional wired networks are not directly applicable to wireless environments, where transmission speed is relatively slow, interference is relatively high, and the network topology is very dynamic. These characteristics do not exist in wired networks, but they pose a major challenge in providing multimedia service to mobile users.

In this dissertation, we present our work toward the support of multimedia applications over wireless networks. Among many network functions in different layers,

two are especially targeted, namely, routing and renegotiable QoS support. Routing in wireless networks is one of the most fundamental problems for mobile computing. Its main purpose is to determine the path to connect two mobile stations. Renegotiable QoS permits us to readjust resource allocation so as to optimize quality, with considerations of dynamic topology and unreliable channel conditions in wireless networks. By means of efficient wireless routing at the network layer and renegotiable QoS in the application layer, the level of satisfaction of multimedia service users in wireless networks will be greatly improved.

1.1 Background: Wireless Mobile Networks

Based on the hop distance of packet transfers, wireless networks can be classified into two types: single-hop and multi-hop. The single-hop network generally requires preconfigured, fixed infrastructures. It is aiming at the provision of wireless access in a civilian area. The multihop network, on the other hand, does not rely on a fixed infrastructure, thus can provide a more flexible service, for example, in a rural area. These two different wireless networks are detailed below.

1.1.1 The Single-hop Wireless Network

In a single-hop wireless network, the whole service area is divided into several smaller service regions called cells. In each cell, at least one base station is allocated to provide network service to mobile hosts in the cell. The mobile host connects to the network by establishing a wireless connection to the base station. The connections among base stations are usually provided by high speed wired backbone. Fig. 1.1 shows an abstracted model of single-hop network.

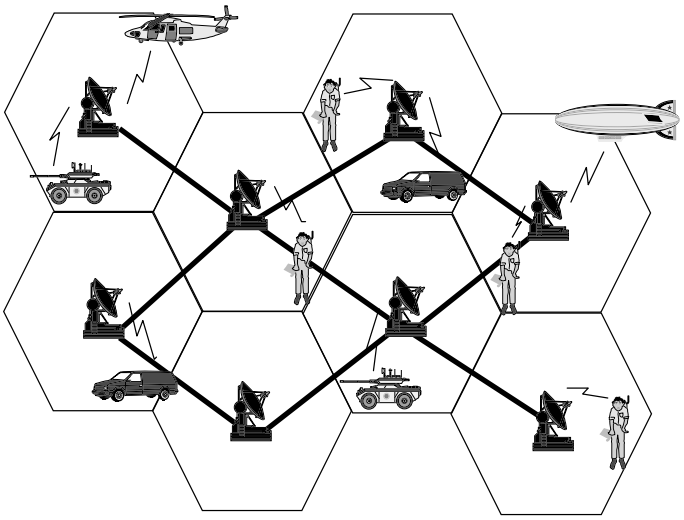

Figure 1.1: A Single Hop Network

Because of the wired backbone, wireless communications in single-hop network only exist between a mobile host and the associated base station. The end to end delivery of data packet relies mostly on the technology available for wired backbone. The major challenge in single hop network is the hand off problem caused by host mobility. The process of "hand off" happens when a host moves out of the transmission range of the current base station, and enters a cell served by another base station. In order to keep the connection alive and maintain a seamless packet delivery, extra procedures have to be performed. Proposed schemes in this are can be found in [RW94, PH95].

1.1.2 The Multi-hop Wireless Network

The drawback of a single hop network is that it requires a pre-established communication backbone, which is infeasible under certain circumstances. For example, battlefield, disaster (flood, fire, earthquake) recovery, search and rescue, or exploration of an

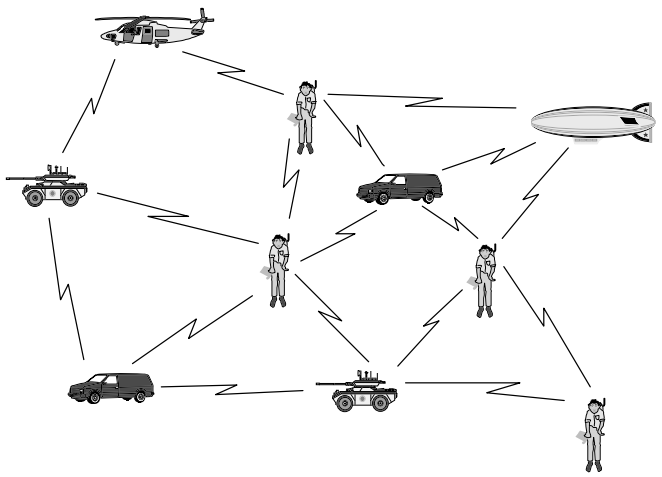

Figure 1.2: A Multihop Network

unpopulated area, etc. Applications of these types require an instant infrastructure to carry multimedia information. The multi-hop wireless mobile network, also called “ad hoc” network, serves this need because it relies merely on the wireless communication and allow host mobility. An abstracted example of multihop networks are shown in Fig. 1.2.

In a multihop network, all hosts move freely and do not require any fixed communication infrastructures. However, due to the limit of radio transmitted power, not all radios are within range of each other. Distinct from the “hand off” problem in single-hop network, the challenge in multihop networks is how to relay data packets from one host to another, and how to do it efficiently. Research in this area can be tracked back to the ’70’s, when DARPA started the Packet Radio Network (PRNET) project [JT87]. The PRNET focused on the support of datagram traffic. In ’90, the PRNET successors, WAMIS and GLOMO, focus on the issue of multimedia support.

1.2 Scope of This Research

Inspired by the WAMIS project, our work focuses on the QoS issue in multihop, ad hoc networks. Following is the research proposed in our work.

1.2.1 Efficient Wireless Routing with QoS constraints

An efficient routing protocol is the foundation of a multi-hop wireless network. For real time services, it is critical for a routing protocol to consider both reachability and connection quality. Therefore, the goals of our research in wireless routing are: first, the routing scheme has to be efficient so that the network information can be rapidly disseminated to all hosts without wasting too many bandwidth. Second, the routing scheme should provide link quality for the network management layer so that when a new call is issued, the network management layer decides whether to accept this call based on network conditions. Third, with QoS in mind, the routing algorithm will not only find out a path can reach the destination, but also a path which satisfies users QoS requirements.

1.2.2 Support of Renegotiable QoS

Multimedia applications place stringent requirements on networks for delivering multimedia content in real-time. Compared to the requirements of traditional data-only applications, these new requirements generally include high bandwidth availability, low packet loss rate, and a low variation in packet delivery time. Unfortunately, in a wireless environment, no guarantee on these requirements can be safely made in the fact of mobility. Therefore, in order to maintain same level of acceptable quality over such networks, we need to take a new look at QoS support. Namely, we proposed

the concept of renegotiable QoS, which requires interaction between application and network system. Based on this concept, we show that the information quality can be greatly improved even in an unreliable network.

1.2.3 Testbed Implementation

Implementation is another important goal for our research. With this goal in mind, our work not only has to be valid from the theoretical point of view. It also has to be robust enough to be implemented and used in a real environment. Our testbed is based on portable, laptop PCs, equipped with wireless devices operating in the unlicensed spread spectrum band. Our protocols are implemented in various sublayers. Several experimental applications are also implemented to validate and verify the system.

1.3 Contributions of This Research

This dissertation presents research toward the goal of a multimedia, wireless mobile networks. The document is organized according to the outline below:

- **Evaluation of Existing Wireless Routing Algorithms** In order to study routing in a mobile environment, we survey the existing routing algorithms, for both wired and wireless networks. Through a series of complex analysis and comprehensive simulations, we evaluate the pros and cons of different algorithms when applied to wireless networks. Chapter 2 surveys the existing schemes for routing in wireless networks.

- **Global State Routing** Based on the observation of existing routing schemes, we found that the impact of control overhead and host mobility are seldom considered in the previous works. Realizing this, we proposed the Global State Routing

(GSR), which is an improved version of link state routing. GSR is more efficient for wireless networks, and can be easily extended to deal with QoS routing. It is also more accurate than traditional distance vector routing in tracking shortest path in presence of mobility. The details of GSR is reported in chapter 3.

- **Fisheye Routing** The principle of fisheye routing is to maintain high accurate routing information for nodes close-by, but less accurate for nodes far away. This approach handles the exchange of network information in a more efficient way. Each node exchanges network information with its neighbor at two different update frequencies. The higher frequency is used for exchanging information for nodes within n hop distance, where only a relatively small number of nodes are covered. The lower frequency is used for exchanging information about nodes out side n hop distance, the majority of the nodes. Through a series of simulation and analyses, fisheye routing exhibits great reduction in control message size, and the routing accuracy is about the same as those without fisheye. More details on fisheye can also be found in chapter 3.

- **QoS Routing** QoS routing presents a great challenge in both wired and wireless network. Our work towards it includes two steps: 1) routing with QoS information, this can help the admission control function in preventing network overload; 2) find out a QoS satisfactory route, this can help load balancing in a low speed wireless network. We report the results of our QoS routing in chapter 4.

- **Renegotiable QoS Support** QoS routing can provide a path which meets QoS requirements at call setup time. However, it is not guaranteed that the QoS requirements will be met in the life of the connection. In a wireless network with host mobility, topology changes and interference in radio communication may invalidate the initial guarantees. Realizing this, we have designed a renegotiable

QoS scheme that provides a renegotiation function between user applications and network systems. The implementation of this work is reported in chapter 5.

- **Multimedia on demand system with QoS adaptation** With renegotiable QoS support, we present a programming model for multimedia application such that the information can be better preserved. Based on this model, an on-demand audio system has been implemented and the information carried over the unstable link is greatly improved. Chapter 6 has detailed description on this on demand, QoS adaptive, audio system.

CHAPTER 2

Routing in Wireless Networks - A Review

Routing is the function in the network layer which determines the path from a source to a destination for the traffic flow. In wireless networks, due to the host mobility, network topology may change from time to time. It is critical for the routing protocol to deliver data packets efficiently between source and destination. In the past, several approaches were proposed to provide wireless routing by adapting techniques developed in the wired networks. In this chapter, we first introduce the network model that we use through out this dissertation, then we survey the existing schemes proposed for end to end routing in wireless networks, including traditional distance vector, link state routing, as well as newer path-finding and on-demand routing, in the following sections.

2.1 Network Model

The ad-hoc network under consideration is a homogeneous network such that all mobile hosts have the same computation power, memory capacity and communication capability. With this assumption, the network can be modeled as an undirected graph $G = (V, E)$, where V is a set of $|V|$ nodes and E is a set of $|E|$ undirected links connecting nodes in V. Each node has a unique identifier and represents a mobile host with a wireless communication device which has transmission range R (free space

propagation model), and an infinite storage space. Nodes can move around and randomly change their directions of movement independently. An undirected link (i, j) connecting two nodes i and j is formed when the distance between i and j becomes less than or equal to R. Link (i, j) is removed from E when node i and j move apart, and out of their transmission ranges.

2.2 Distributed Bellman-Ford

Many existing routing schemes for ad hoc wireless network are based on the distributed Bellman-Ford's (DBF) algorithm. These schemes are also referred to as distance vector (DV) schemes. In the distributed Bellman-Ford algorithm, every node i maintains a routing table which is a matrix containing distance and successor information for every destination j, where distance is the length of the shortest distance from i to j and successor is a node that is next to i on the shortest path to j. To keep the shortest path information up to date, each node periodically exchanges its routing table with neighbors. Based on the routing tables received with respect to its neighbors, node i learns the shortest distances to all destinations from its neighbors. Thus, for each destination j, node i selects a node k from its neighbors as the successor to this destination (or the next hop) such that the distance from i through k to j will be the minimum. This newly computed information will then be stored in node i's routing table and will be exchanged in the next routing update cycle. Fig. 2.1 shows an example of DV routing.

The advantages of DBF are its simplicity and computation efficiency due to its distributed characteristic. However, it is well known that DBF is slow to converge when topology change, and has the tendency to create routing loops, especially when the link conditions are not stable [BG87]. These problems are described below.

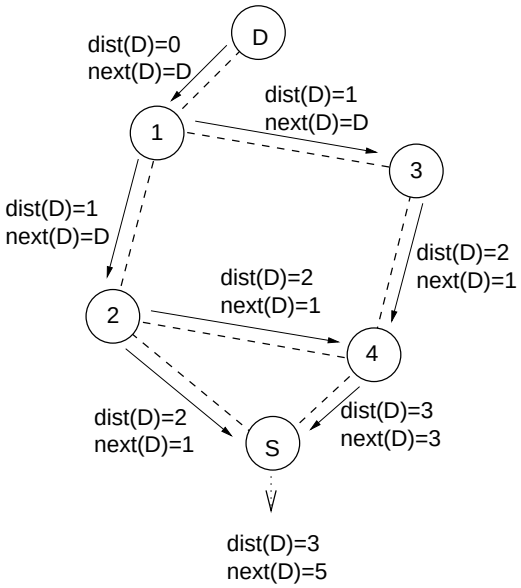

Figure 2.1: An example of Distance Vector Routing

2.2.1 Looping in Distributed Bellman Ford

The existence of routing loops is due to the fact that in DBF, nodes compute the shortest path and choose their successors in a fully distributed fashion, based on information that may not correctly reflect the actual network topology. An example is shown in Fig. 2.2.

The discussion of looping problem can also be found in [BG87]. Partial remedies like *split-horizon* and *poisoned-reverse* were developed and used in wired network protocol like RIP [Hed88]. However, *split-horizon* and *poisoned-reverse* are still insufficient for solving the loop problem in wireless networks [Gar89a].

2.3 Link State

Another algorithm that is also widely used in many existing routing protocols, such as OSPF [Moy94], is the link state (LS) routing. Although LS routing is seldom used for

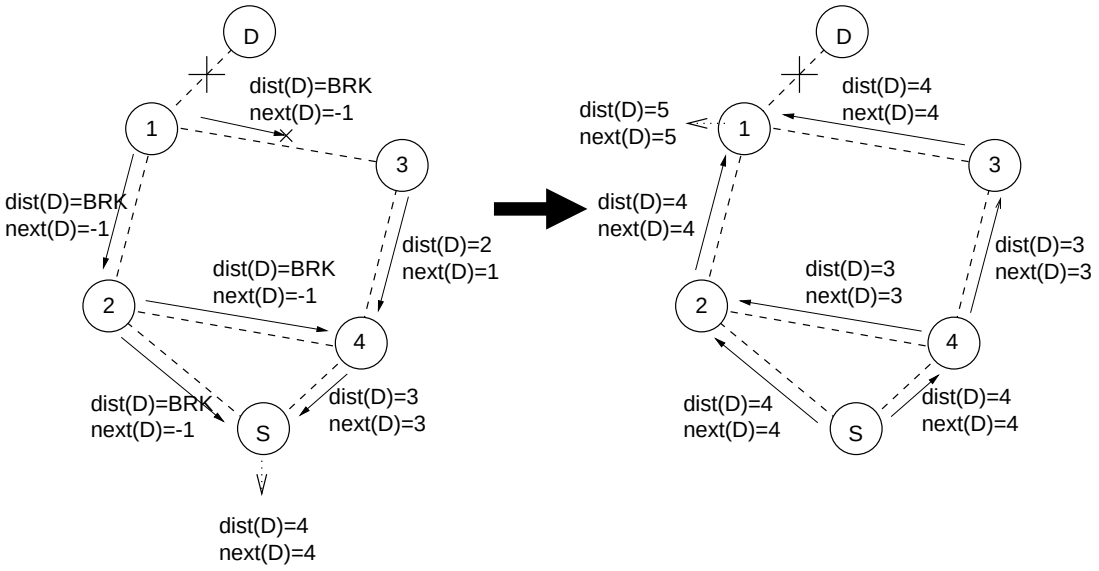

Figure 2.2: Routing loop in Distance Vector Routing

wireless networks, it has more potential for providing customized routing solution for various routing criteria. This is because paths are computed base on the global network topology, as opposed to the abstracted network view reported by neighboring nodes.

An example of link state routing is shown in Fig. 2.3. In LS routing, whenever a node detects a change in its local connectivity to, it floods a new link state packet containing its updated local connectivity. Other nodes are notified of this change when the link state packet arrives, so their view of network topology can then be modified accordingly. From this aspect, LS responds faster to topology changes than DBF, and is about the same as on-demand routing, which will be covered shortly. It computes routes in a centralized fashion, so it is easy to prevent routing loops. However, since LS also relies on flooding to disseminate information about the connectivity changes of nodes, the control overhead of flooding makes LS inferior to DBF or on-demand approach in a wireless mobile environment, despite the accuracy it provides. Another problem in LS is that a node may fail to discover the true topology change if the whole

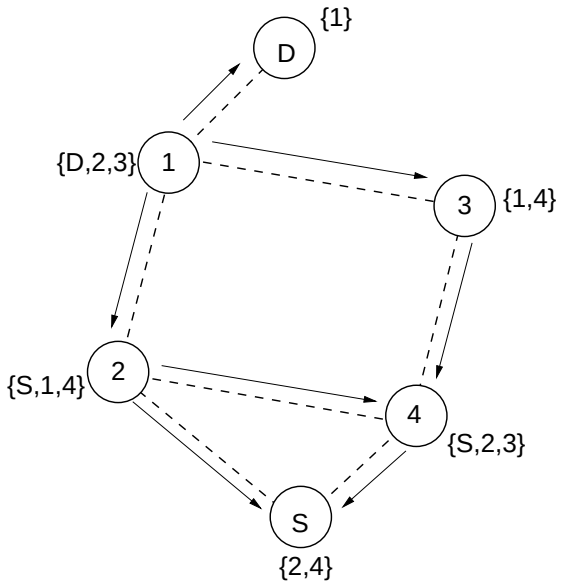

Figure 2.3: An example of Link State Routing

network has been divided into two parts and then recovered. Solution to this problem can be found in [Jaf86].

2.4 Path Finding

New approaches based on DBF to provide loop-free routing specially for wireless network have appeared recently, such as the Destination-Sequenced Distance Vector (DSDV) [PB94] and the Wireless Routing Protocol (WRP) [MG95]. Even though the looping problem is solved in these approaches, there is still a problem of routing inaccuracy in DBF which may degrade network performance. This routing inaccuracy is caused by the fact that in a network utilizing DBF, nodes don't have a global view of the network status; thus their routing decisions are made only locally optimized; it does not necessarily guarantee a globally optimized solution in a mobile environment. In addition, as DBF only maintains a single path to a destination, it lacks the ability to adapt to link failures and it requires more extension works to support multicasting.

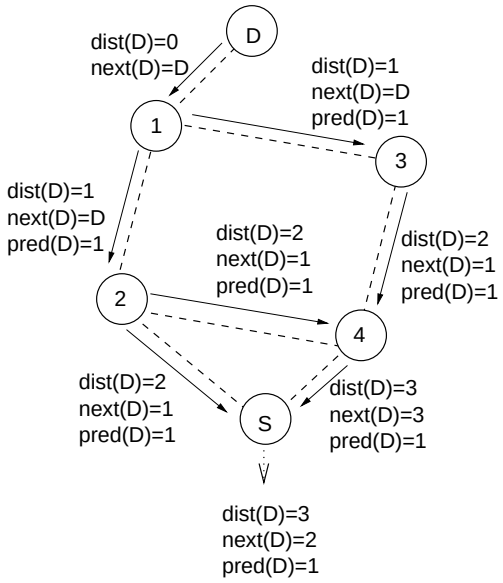

Figure 2.4: An example of Path Finding

An example of path finding is shown in Fig. 2.4. Here we see the computation for the shortest path from node S to node D. Based on the path predecessor information collectively received by node S, node S learns that in order to reach node D, it has to visit node 1 first, and to reach node 1, it has to visit node 2 first. Thus a whole path starting from node S, than node 2, than node 1, than node D, is computed.

2.5 On-demand Routing

Besides DBF, On-demand routing, also known as diffusion computation (DC) is another scheme used for routing in wireless network, such as the Lightweight Mobile Routing (LMR) protocol [CE95] and Temporally-Ordered Routing Algorithms (TORA) [PC97].

In the on-demand routing scheme, a node builds up a route by flooding a query to all nodes in the network. The query packet “picks up” the IDs of the intermediate

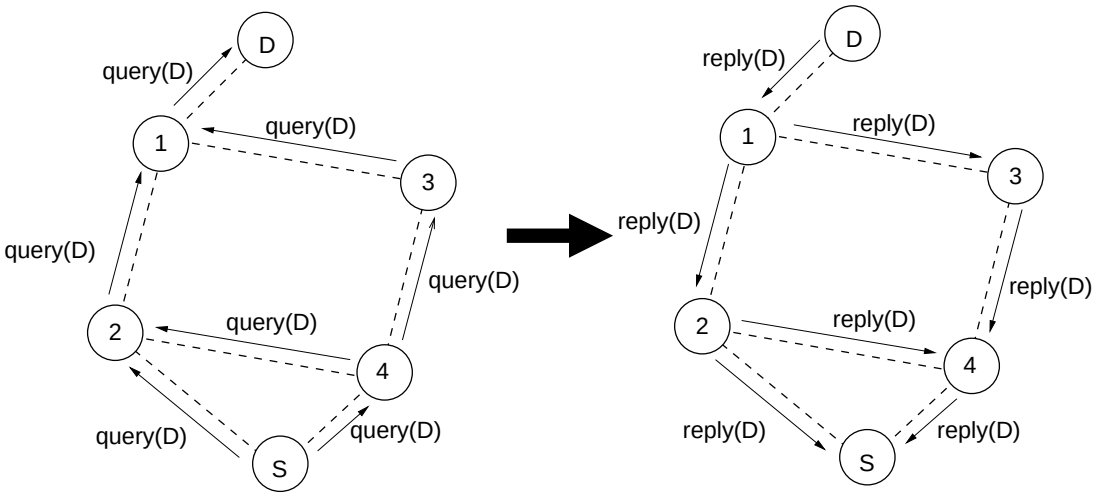

Figure 2.5: An example of On-demand Routing

nodes and stores them in a path field. On detecting the query, the destination or any other node who has already learned the path to destination answers the query by sending a "source routed" response packet back to the sender. Since multiple responses may be produced, multiple paths may be computed and maintained. After the paths are computed, any link failure will trigger another query/response so the routing can always be kept up to date. Though approaches based on DC reflect higher accuracy and faster response to network changes, they introduce excessive control overhead since they require frequent flooding, especially when mobility is high and traffic is dense and uniformly distributed. As a result, on-demand routing protocols are only suitable for wireless network with high bandwidth, small packet transmission delays and very sparse traffic.

2.6 Zone Routing

Zone routing [Has96] is another routing protocol designed for the ad hoc environment. It is a hybrid of on-demand routing with any existing routing protocol. In zone routing,

each nodes defines its own zone as the nodes within certain distance of itself. Two different routing schemes are required for zone routing. For routing inside the zone, any routing schemes, including DBF or LS, can be used. The goal for this intra-zone routing is to maintain a full information about the reachability of nodes within the region. For the inter-zone routing, it uses the on-demand routing to find the path. Combining these two routing schemes, zone routing operates like this: when there is a traffic needs to be routed, it checks whether the destination is within the zone. If it is, since the intra-zone routing scheme maintains the necessary information, it can be routed directly. When it comes to route a traffic to a destination outside a node's zone, zone routing searches for the path by multicasting request packets to the border nodes, using the shortest paths provided by the intra-zone protocol. If this destination is known by some border nodes, the response packets will then be sent back to the source. Otherwise, the border nodes keep requesting their border nodes, in the same fashion, for a route to the destination.

The advantage of zone routing is its scalability, as it reduces the need for a large storage for the routing table. But since it resembles the on-demand routing, it has the same problem of connection delay, and the termination of request packets.

2.7 Summary

Since DBF computes routes distributedly based on abstracted information, it tends to create routing loops and reacts slowly to a link failure. Link State routing suffers from the flooding overhead, so does on-demand routing. None of them satisfies the stringent requirement of supporting multimedia traffic in wireless networks. In order to support multimedia traffic, the routing algorithm needs to compute routes subject to different QoS constraints. This can be done best using global network knowledge, like

in link state. However, for efficiency, the ideal routing algorithm should not rely on flooding to disseminate the information, since this may cause excessive control packet overhead.

CHAPTER 3

Global State Routing

In this chapter, we present a new routing scheme for ad-hoc wireless networks. The goal is to provide an accurate routing solution while the control overhead is kept low. The network efficiency is achieved even when host mobility is high and network bandwidth is limited. Our proposed scheme is named "Global State Routing" (GSR). Similar to LS, GSR generates accurate routing decisions by taking advantage of the global network information. However, this information is disseminated in the network in a way that is more similar to DV. That is, the control packets are only exchanged locally. To reduce the control overhead GSR is composed of several techniques which will be described in the following sections. Along with a detailed description, the performance of GSR and other protocols are compared at the end of this chapter.

3.1 The Protocol Overview

To introduce the concept of GSR protocol, we use the same ad-hoc wireless network model defined in the previous chapter. Information that is maintained at each node i includes one list and three tables. They are: a neighbor list A_i, a topology table TT_i, a next hop table $NEXT_i$ and a distance table D_i. A_i is defined as a set of nodes that are adjacent to node i. That is, $A_i = \{x \mid (x \in V) \text{ and } (\text{link } (i, x) \text{ exists})\}$. One entry is allocated for each destination j in table TT_i which contains two parts: $TT_i.LS(j)$

and $TT_i.SEQ(j)$. $TT_i.LS(j)$ denotes the link state information reported by node j, and $TT_i.SEQ(j)$ denotes the timestamp indicating the time node j generates this link state information. Similar, for every destination j, *NEXT*$_i(j)$ denotes the next hop to forward packets destined to j on the shortest path, and $D_i(j)$ denotes the distances of the shortest path from i to j.

Additionally, a weight function, *weight*: $E \rightarrow Z_0^+$, is defined for each link. If min-hop shortest path is considered, it simply returns 1 if two nodes have direct connection. Otherwise, it returns ∞. This weight function can be replaced with other functions for routing with different metrics (e.g. bandwidth). Various metrics will be used for QoS routing, we report them in Chapter 5.

3.2 Algorithm

The detailed pseudo code of GSR protocol is listed in Section 3.5. As shown in *NodeInit(i)*, each node i initially starts with an empty neighbor list A_i, and an empty topology table TT_i. After its local variables are initialized, it learns about its neighbors by examining the sender field of each packet in its inbound queue, *PktQueue*. That is, assuming that all nodes can be heard by i are i's neighbors, node i adds these senders' IDs to its list, A_i.

Node i then invokes *PktProcess(i)* to process the received routing messages, which contain the topology information prepared and disseminated by its neighbors. This topology information is indeed the link states that used by link state routing. Procedure *PktProcess(i)* makes sure that only the most up to date link state information is recorded by comparing the embedded sequence number, $pkt.SEQ(j)$, with the ones stored in node i's local memory, for each destination j. If there is any entry in the in-

coming message that has a newer sequence number for destination j, $TT_i.LS(j)$ will be replaced by $pkt.LS(j)$, and $TT_i.SEQ(j)$ will be replaced by $pkt.SEQ(j)$.

After the routing messages are examined, node i rebuilds the routing table based on the newly computed topology table and then broadcasts the new information to its neighbors. This process is periodically repeated.

3.2.1 Information Dissemination

The key difference between GSR and traditional LS is the way routing information is disseminated. In LS, link state packets are generated and flooded into the network whenever a node detects topology changes. GSR does not flood the link state packets. Instead, nodes in GSR maintain a link state table based on the up-to-date information received from neighboring nodes, and periodically exchange it with their local neighbors only. Information is disseminated as the link state with larger sequence numbers replaces the one with smaller sequence numbers. In this respect, it is similar to DBF (or more precisely, the DSDV [PB94]) where the value of distances is replaced according to the time stamp of sequence number.

To prepare the update message, the most intuitive way is to simply use the topology table. But doing so may cause large size update message which consumes considerable amount of bandwidth. To avoid that, GSR uses two techniques to reduce the size of update message without sacrificing too much in routing accuracy. These two techniques are fresh update and fisheye.

3.2.1.1 Fresh Update

Fresh update allows a node to disseminate only the information that is useful to at least one of its neighbors. Consider the information for destination j maintained at node i,

if it is the oldest among all i's neighbors, it is useless for node i to disseminate this information. By eliminating the updating for obsolete network information, the size of routing message can be reduced.

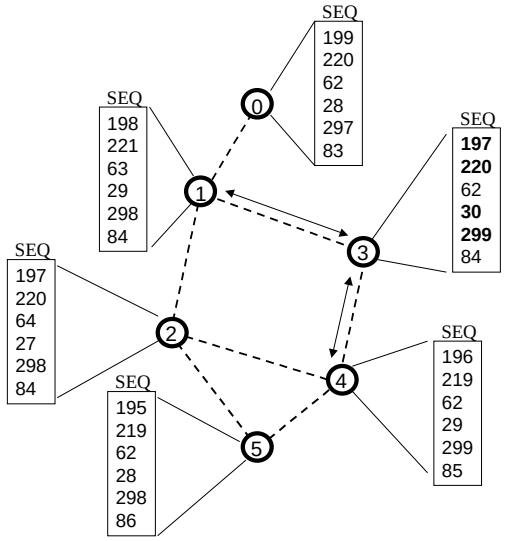

Figure 3.1: Fresh update: phase 1

Two phases of message update are required in GSR. In the first phase, Each node i broadcasts a sequence number vector (SNV) that contains all destinations maintained in its own local storage. This SNV indicates the "freshness" of each host information that node i is based on. In the second phase, node i compares its own SNV with those received in phase one, if node i has an entry which has the oldest (smallest) sequence number, node i removes this entry from its update message. After node i scan through the entries for all destinations, the resulting update table is ready to be disseminated. Fig. 3.1 and 3.2 show the two phases in fresh update. In Fig. 3.1, all nodes broadcast their own SNV to their intermediate neighbors. Consider node 3 in Fig. 3.2, after comparing all entries in its own SNV and SNVs from its neighbor nodes, 1 and 4, only the entries that printed in bold are the ones worth for dissemination.

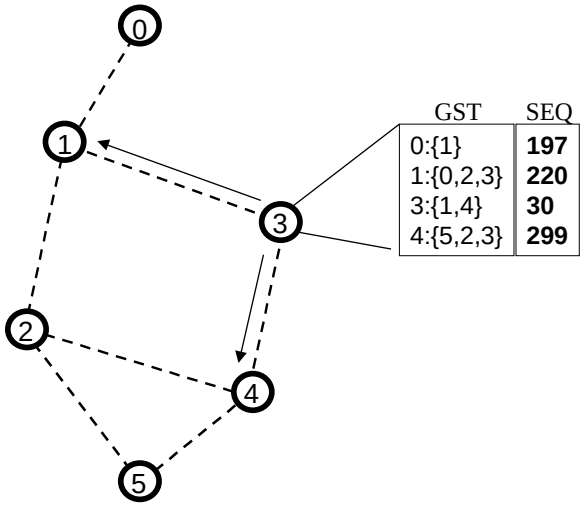

Figure 3.2: Fresh update: phase 2

3.2.2 Fisheye

Another approach used to reduce the size of update message is called “Fisheye”. In [KS71], Kleinrock and Stevens proposed the fisheye technique to reduce the size of information required to represent graphical data. The original idea of fisheye is to maintain high details for information within a range of a certain point of interest; and less detail while the distance to the point of interest increases. For routing, this fisheye approach can be interpreted as maintaining a highly accurate network information about the immediate neighborhood of a node, with progressively less detail as it moves away from the node.

Fig. 3.3 illustrates the application of fisheye in a mobile, wireless network. In this figure, we show the scope of fisheye for the center node. The small circles with number inside represent the mobile hosts in the network. The large circles plot the fisheye scope of the center node. The scope of fisheye is defined as the nodes that

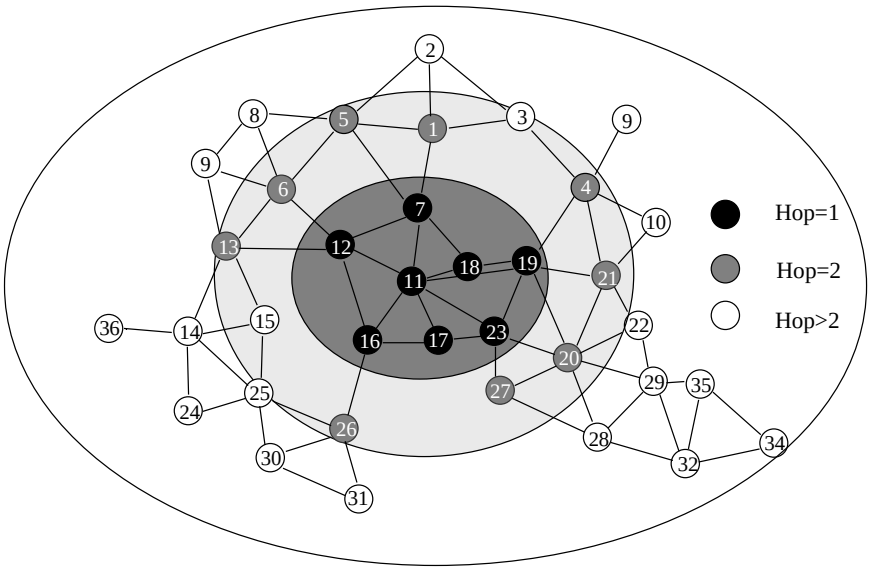

Figure 3.3: Scope of fisheye

can be reached within a certain number of hops. In our case, three scopes are shown and they represent the scope of 1-hop, 2-hop and 3-hop. Nodes that are located within scopes of different hop distances are plotted as black, grey and white, representing scope of 1-hop, 2-hop and 3-hop, respectively.

The reduction of message size is achieved by updating the network information for nearby nodes at a higher frequency for better accuracy. For remote nodes which are outside the fisheye scope, a lower frequency is used for smaller packet overhead in average. The detailed operation of this fisheye update is listed in Section 3.5. Procedure *RoutingUpdate(i)* scans through the update message and filters out entries that have hop distance larger than the fisheye scope. Fig. 3.4 depicts this operation. In this figure, entries in bold face indicate the actual messages to be disseminated into the network. The rest of the entries will still be sent out eventually, but at a much lower frequency. As a result, considerable amount of link state entries are suppressed so that the message size is reduced.

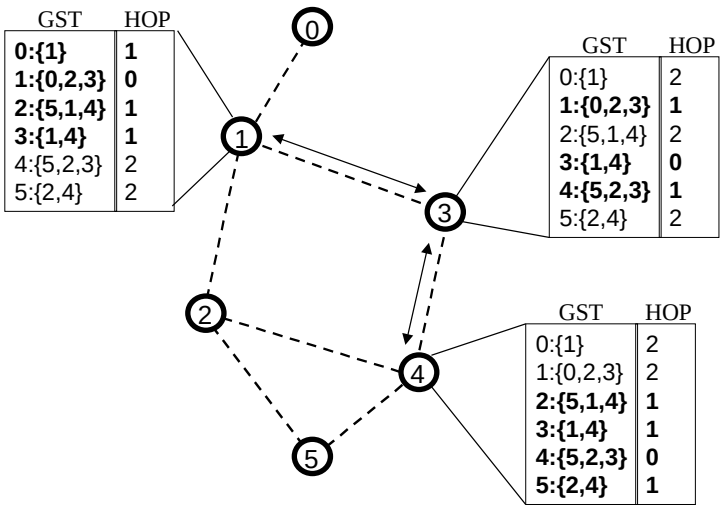

Figure 3.4: Message reduction using fisheye

3.2.3 Shortest Path Computation

FindSP(i) creates a shortest path tree rooted at i. In principle, any existing shortest path algorithm can be used to create the tree. In this paper, however, the procedure listed in Section 3.5 is based on the Dijkstra's algorithm [Sed83] with modifications so that the next hop table (*NEXT*$_i$) and the distance tables(D_i) are computed in parallel with the tree reconstruction.

At node i, *FindSP(i)* initiates with $P = \{i\}$, then it iterates until $P = V$. In each iteration, it searches for a node j such that node j minimizes the value of $(D_i(k) + weight(k, j))$, for all j and k, where $j \in V - P$, $k \in A_i$ and ***weight(k,j)*** $\neq \infty$. Once node j is found, P is augmented with j, $D(j)$ is assigned to $D(k) + weight(k, j)$ and *NEXT*$_i(j)$ is assigned to $next_i(k)$. That is, as the shortest path from i to j has to go through k, the successor for i to j is the same successor for i to k.

3.3 Complexity

In this section, we analyze the complexity of the GSR scheme and compare it with two other routing schemes: DBF and LS. The complexity is studied under five aspects:

1. Computation Complexity (CC): the number of computation steps for a node to perform a routing computation after an update message is received;
2. Memory Complexity (MC): the memory space required to store the routing information;
3. Data Complexity (DC): the aggregate size of control packets exchanged by a node in each time slot;
4. Packet Complexity (PC): the average number of routing packets exchanged by a node in each time slot;
5. Convergence Time (CT): the times requires to detect a link change.

Protocol	CC	MC	DC	PC	CT
GSR	$O(N^2)$	$O(N \cdot d)$	$O(d) + O(N-d)/I$	$O(1)$	$O(D \cdot I)$
LS	$O(N^2)$	$O(N^2)$	$O(\|N\|)/I$	$O(N)$	$O(D)$
DBF	$O(N)$	$O(N)$	$O(N)/I$	$O(1)$	$O(N \cdot I)$

Table 3.1: Complexity Comparison

Table 3.1 shows the results of our comparison. In the table, N denotes the number of nodes in network ($|V|$), D denotes the maximum hop distance, the diameter, in

the network, d and I denote the degree of node connectivity and the routing update interval, respectively.

GSR and LS have the same memory complexity and computation complexity as both of them maintain the network topology for the whole network and use Dijkstra's algorithm to compute shortest path routes. Dijkstra's algorithm requires typically $O(N^2)$ steps to compute the shortest paths from one source to all destinations, although it is possible to reduce it to $O(NlogN)$ [Sed83]. $O(N^2)$ memory space is required to store the network topology represented by a connection matrix. As for DBF, it has complexity of $O(N)$ for computing and memory, as it only keeps the distance information for each destination, and computes shortest paths in a distributed fashion.

The data complexity of GSR is $O(d) + O(N - d)/I$, which is smaller than LS's $O(|E|)/I$, and is close to DBF's $O(N)/I$, as d is usually a small number. In GSR, with the fisheye technique, each node broadcasts information about all $O(d)$ intermediate neighbors within its fisheye scope in each routing update. For the rest of of nodes, GSR updates them at a longer update interval, thus the amortized cost of updating the remote hosts is $O(N - d)/I$, where I is the long update interval used for remote hosts update. LS, on the other hand, requires each node not only broadcasts its link state packets to all hosts in the network, but also forward link state packets received from others for the purpose of flooding. Therefore, in a extremely high mobility environment where all nodes detect topology changes in almost every update interval, excessive amount of these small link state packets will be flooded into the network. As a result, a node may process as many as N link states flooded by itself and the rest. This issue, to be addressed shortly, can be verified through simulations.

Similar to the data complexity, as LS transmits one short packet for each link state update, its packet complexity can be as high as $O(N)$ when the mobility is high. On the

other hand, both GSR and DBF broadcast their routing messages in group, so fewer, but longer packets can be used to optimize the MAC throughput.

Lastly, the convergence time of GSR is also superior than that of DBF. This is because GSR and LS both compute route based on the global network information, a breakage on a link can be determined by a node as long as it receives the information of this link. DBF try to compute an alternate path to bypass the broken link. If the alternate path does not exist, DBF cannot detect it until the hop count for that node is iterated to the value of infinity, which can be as large as N.

3.4 Performance Analysis

Unlike in [MG95, PB94, CE95, PC97], where a wireless network is simulated by a static network with higher link failure rate, we used a truly mobile environment in our simulator to determine the connectivity among mobile hosts. The strategy of using static network with link failures is not sufficient to represent a true mobile environment where link failures are mostly caused by node mobility. Therefore, we study the performance of GSR as well as other schemes by simulating a truly mobile network with different protocols implemented.

3.4.1 Simulator

The simulation is programmed in C++ to simulate a virtual environment of 500 $\times$ 500 unit^2. An arbitrary number of nodes, representing the mobile hosts, can move independently within this virtual space. The maximum moving speed and the number of nodes are given at run time. We execute the simulation for the cases of node number ranging from 40 to 80 and moving speed ranging from 0 to 150 units per time slots.

The moving pattern of nodes in the simulation is based on a realistic trajectory model instead of the Brownian * model. Fig. 3.5 illustrates these two models. The difference between the two models is: in the realistic model, the new moving direction at time $\tau+1$ is computed based on the previous direction at time τ plus a steering angle, which is generated randomly by a normal distribution process with mean = 0, deviation = 10 degrees. That is, at each simulation time tick, a node changes its direction by a range of -10 to +10 degrees. In the Brownian model, the moving direction of a node at time $\tau+1$ is generated randomly from a uniform distribution process ranging from 0 to 360 degree, so it is independent of its previous direction at time τ. In both models, the moving speed is determined by a uniform process range from 0 to the maximum speed specified at run time. We in fact have evaluated the routing scheme for both models. Interestingly enough, its performance is usually better in the Brownian model than in the trajectory model. This is so because, in the Brownian model, a node's movement is independent of its previous status, hence the moving effects in a series of iterations are not correlated; thus, nodes are usually in the same region in the long run. In the realistic model, however, nodes are more likely to keep moving on their trajectories. Thus the topology change is more significant than what is observed in the realistic model. However, as we believe that the trajectory model operates closer to the real world than the Brownian model, it is the one used in the rest of this paper.

Additionally, the following assumptions are also used in the simulation:

1. no node failure during simulation;
2. node number is always constant in the run time of simulation;
3. time slotted system;

*We borrow this name from the Brownian motion of molecules.

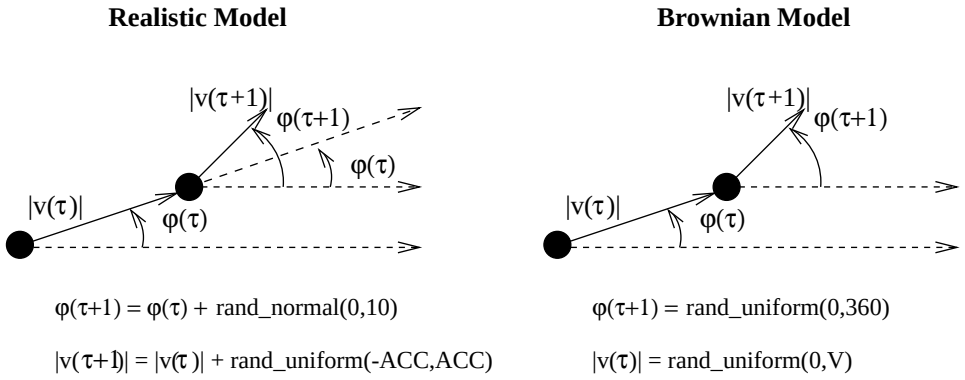

Figure 3.5: Two Mobility Models

4. radio transmission range is fixed at R, which is specified at the beginning of the simulation;

5. two nodes can hear each other only if they are within each other's transmission range. That is, the open space channel model is used.

Based on the mobility models and parameters, we developed simulators for three routing schemes: DBF, LS and our GSR. DBF and LS are based on the scheme described in [Tan96]. Both DBF and GSR can be executed with a routing update interval (I) specified at run time. By default, I is set to 3 (one update per three time slots), while in LS, nodes flood link state packets whenever they detect changes in their local connectivities.

3.4.2 Performance Measurements

Two metrics are used to evaluate the routing performances: routing inaccuracy and control overhead. Using them, we examine the impact to the performance for different mobility values, update intervals and radio transmission distances.

3.4.2.1 Routing Inaccuracy

Routing inaccuracy is checked by comparing the next hop table with the tables generated by an off-line entity. This off-line entity has the knowledge of exact network topology at every time slot and computes the optimal solution for each node. However, giving the same weight to error entries regardless of their distance to the source is not fair. As an incorrect next hop value for a node which is still far away, may be less critical than those for a node that is close by. Therefore, we used both weighted and unweighted in our comparison.

The unweighted inaccuracy for node i, A_i, is defined as:

$$A_i = \sum_{next_i(k) \neq next_M^i(k)} 1$$

This is simply a count for the number of different entries.

The weighted inaccuracy is defined as:

$$\hat{A}_i = \frac{1}{D} \sum_{next_i(k) \neq next_M^i(k)} (D - hop_i(k) + 1)$$

And the overall routing inaccuracy is computed by averaging A_i for unweighted case, or $\hat{A}_i$ for weighted case, for all $i \in N$:

$$Inaccuracy_i = \frac{1}{N} \sum_{i \in N} (A_i \text{ or } \hat{A}_i)$$

where N, $next_i()$, $hop_i()$, D are defined in section III, and $next_M^i()$ is the next hop table computed by the off-line algorithm.

For example, let us consider the node i in a network with its radii D equals to 8. If the next hop information for node j is different than what is computed by the off-line entity, and the hop distance indicated by the off-line table is 5, the effect of this error

is computed as $1 \times (D - 5 + 1)/D$. Substituting D with 8, we get 0.5 for this error. If unweighted inaccuracy is used, we get 1 for this error.

Fig. 3.6 to 3.8 show the unweighted inaccuracy of different routing schemes at different node densities, using. Fig. 3.9 to 3.11 show the weighted inaccuracy. As these figures show, LS performs best in all conditions, since it reacts fastest to the topology changes. GSR performs less accurately than LS in some cases, but by not much. This is because in a mobility environment, a change on a link far away from the source does not necessarily cause a change in the routing table at the source. Thus using a slower update frequency won't affect the routing accuracy by much. In fact, Fig. 3.6 to 3.11 all show that the accuracy of GSR is almost identical to LS, and is much better than DBF.

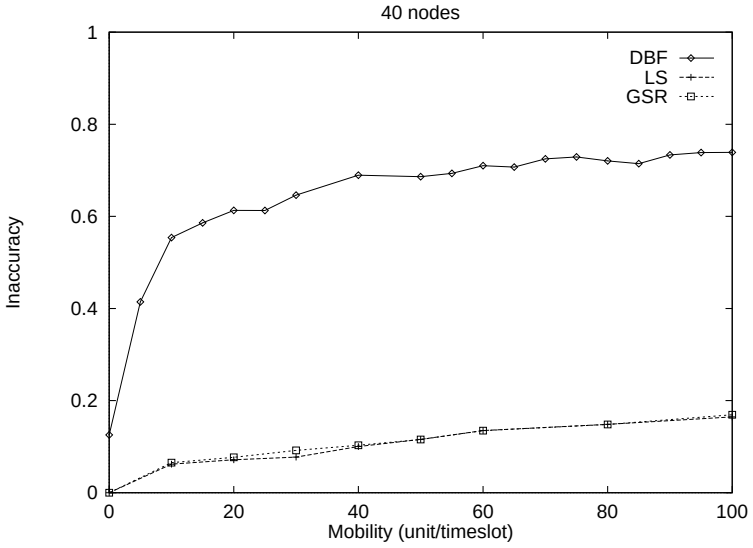

Figure 3.6: Inaccuracy: 40 nodes

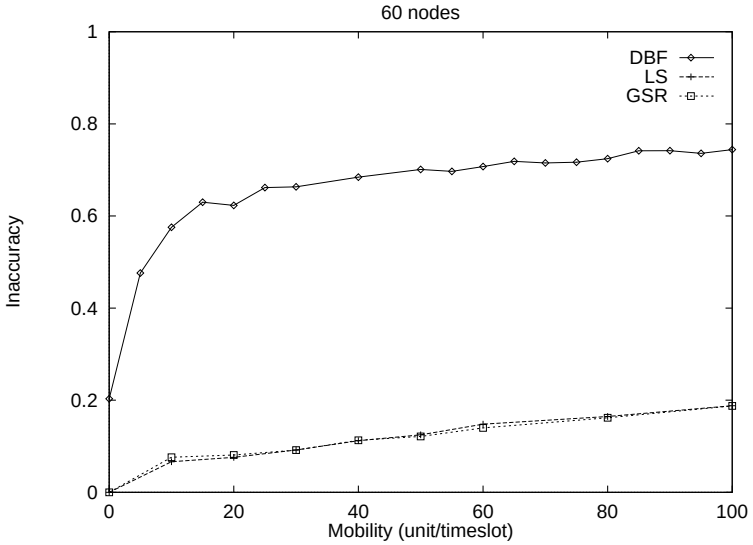

Figure 3.7: Inaccuracy: 60 nodes

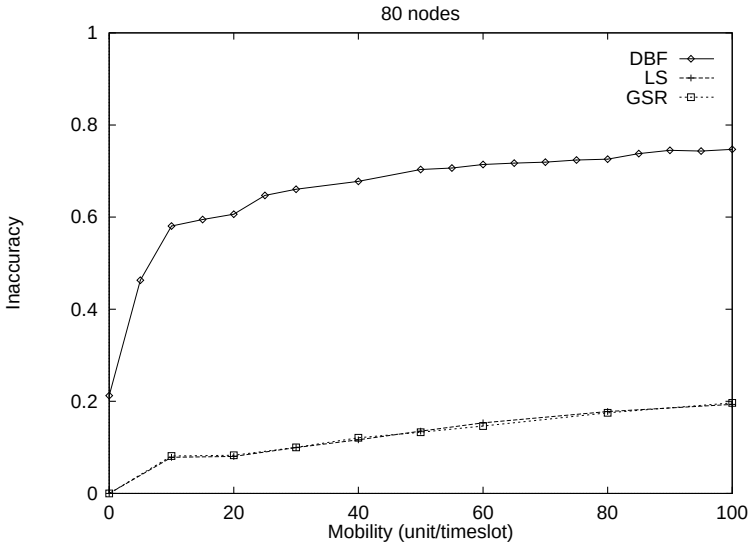

Figure 3.8: Inaccuracy: 80 nodes

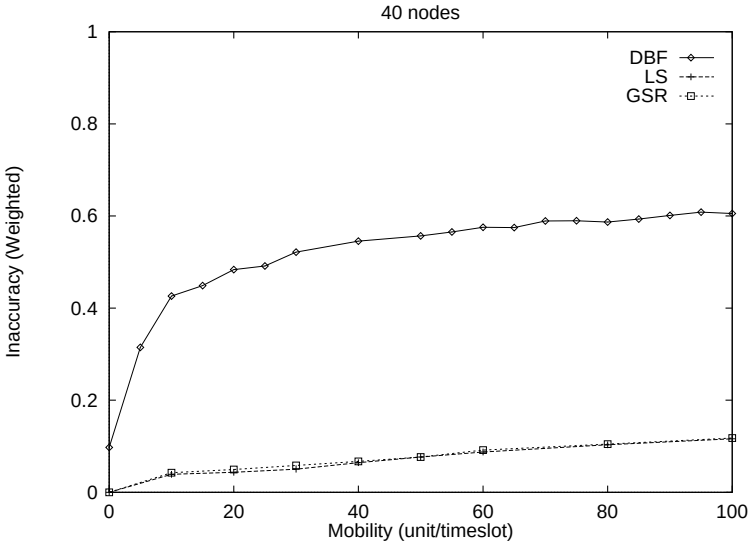

Figure 3.9: Weighted inaccuracy: 40 nodes

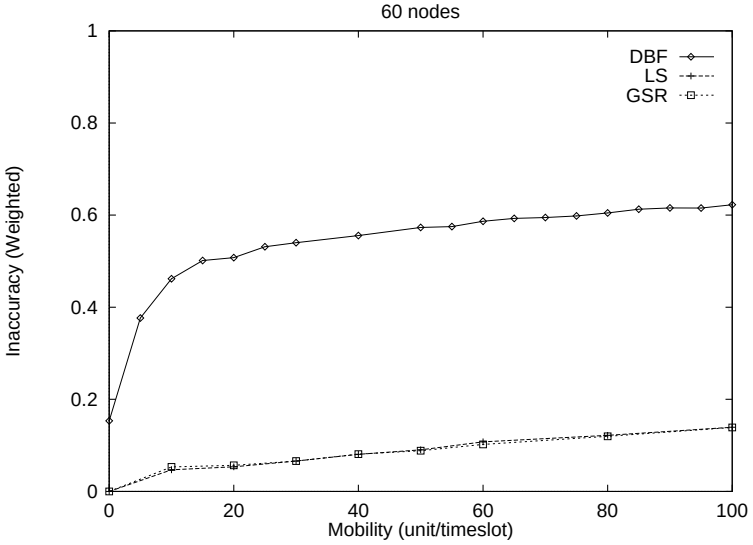

Figure 3.10: Weighted inaccuracy: 60 nodes

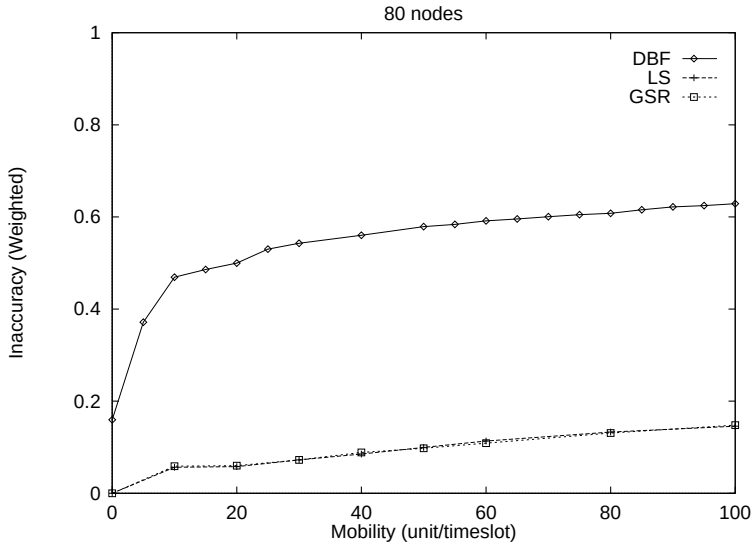

Figure 3.11: Weighted inaccuracy: 80 nodes

3.4.2.2 Control Overhead

The control overhead is evaluated by examining the average number of routing control packets exchanged on each link. The reason for using the number of control packets instead of the total size of control packets is due to the characteristics of the regular radio devices and MAC layer protocol. It is known that a radio device spends more time to switch from receiving mode. This typically exceeds the time used for sending a small packet. If spread spectrum is used, the acquisition time will become even more significant. Based on these considerations, we believe that the number of packets transmitted is as important as the cumulative number of control data bytes for the overhead evaluation.

In LS scheme, we account for each link state packet that is generated by a node either because it detects the topology changes, or it receives one from its neighbors and forwards it for the purpose of flooding, as each of them requires a transition for the radio device from receiving mode to transmitting mode. For DBF type algorithm,

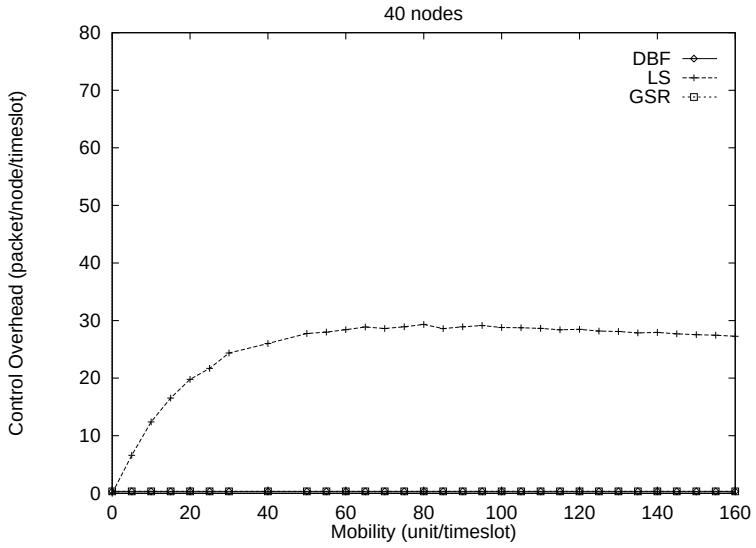

Figure 3.12: Packet number: 40 nodes

a routing table update is counted as one packet. This is under the assumption that the routing table can be transmitted in a constant number of MAC layer frames.

Fig. 3.12-3.14 show the average number of control messages on y axis for different algorithms, versus the maximum moving speed of moving nodes on x axis. The unit on y is the average number of messages transmitted on each link in each time slot, and the unit on x axis is unit distance per time slot.

As expected, both DBF and GSR algorithms have a flat distribution of packet overhead, which means the overhead of both cases remain constant regardless of the mobility, because nodes in both cases exchange routing information periodically with only their adjacent neighbors. On the other hand, with LS schemes, the overhead is much worst than DBF and GSR which means more packets are generated. The figures also show that as degree of mobility becomes higher, the overhead for LS increases. This validates the argument that due to the flooding mechanism, LS is not suitable for high mobility environments.

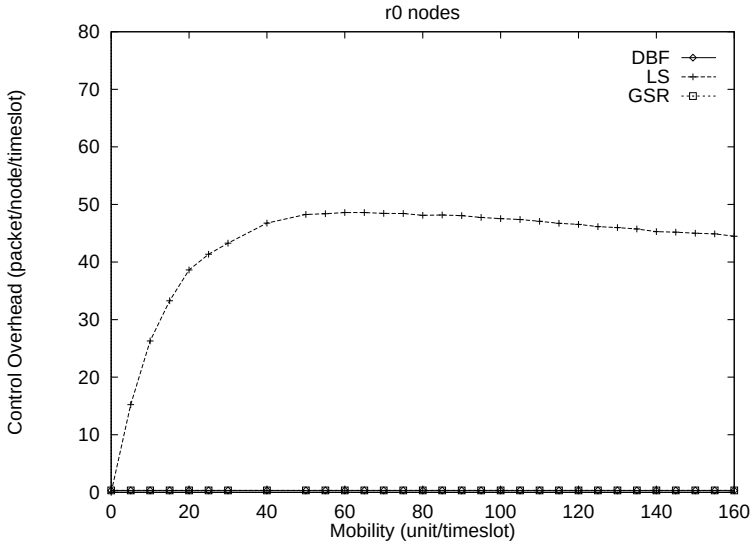

Figure 3.13: Packet number: 60 nodes

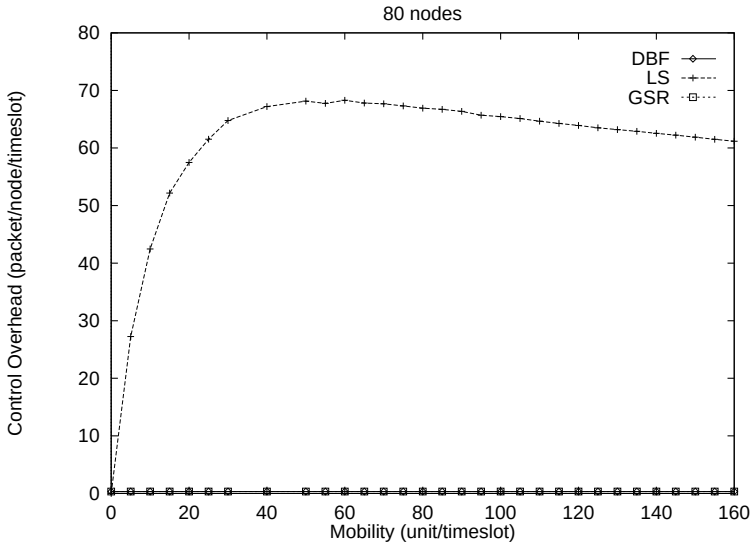

Figure 3.14: Packet number: 80 nodes

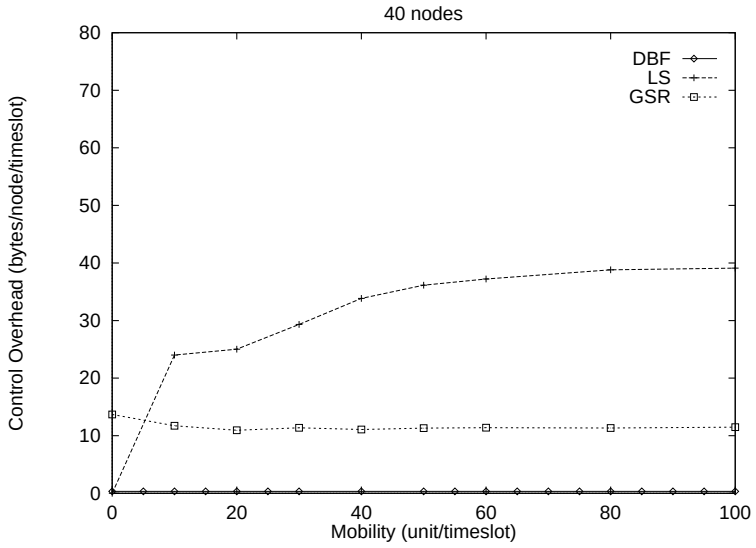

Figure 3.15: Message size: 40 nodes

Fig. 3.15 to 3.17 show the average message size exchanged in each time slot. Obviously, GSR out performs LS with the help of fresh update and fisheye. And similar to DBF, the increase in mobility doesn't cause any impact on the message size, since both DBF and GSR update their routing information at a fixed update frequency, unlike LS where routing updates are driven by topology changes.

3.4.3 Simulation Results

As the purpose of GSR is to provide a routing scheme that is efficient in a wireless network with node mobility. We evaluate the impact to its performance due to changes in mobility, update interval and radio transmission range.

3.4.3.1 Mobility Impact

As Fig. 3.15 shows, the control overhead for LS increases rapidly as nodes move at higher speeds. In a real network, this represents that an unmanageable flood of packets

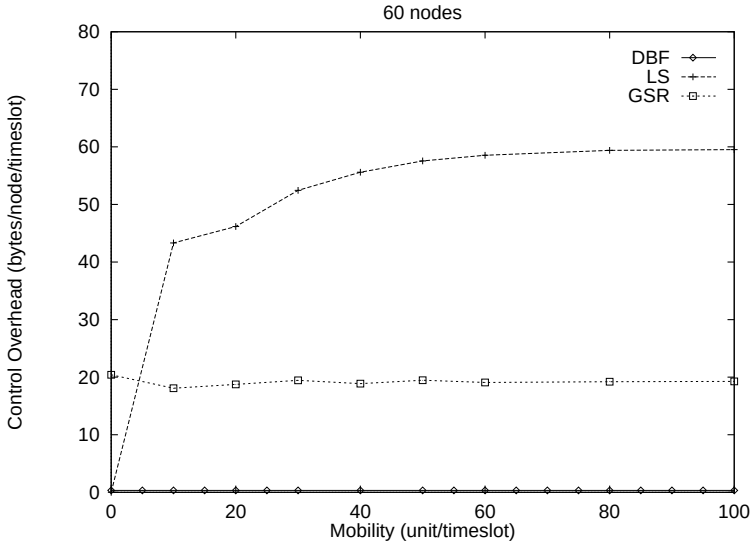

Figure 3.16: Message size: 60 nodes

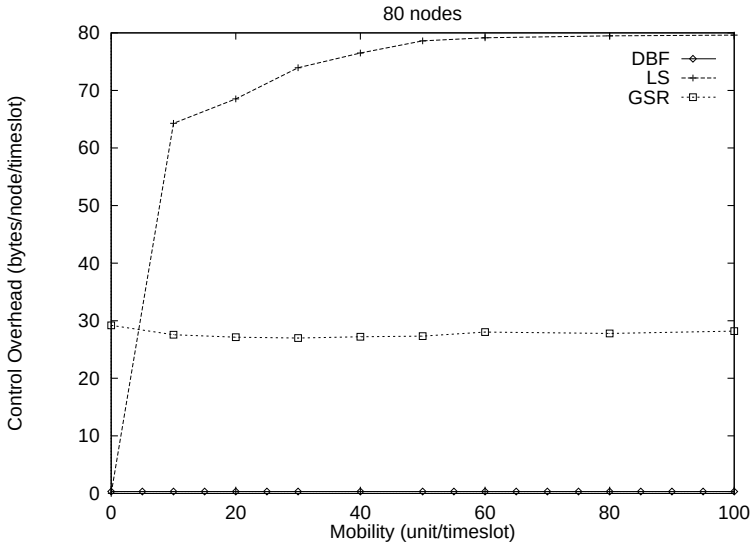

Figure 3.17: Message size: 80 nodes

overwhelms the radio channel and dominates the packet queue in each node. On the other hand, mobility has no effect on control overhead for DBF and GSR. This is reasonable because in LS, routing updates are event driven: a node sends a link state packet into a network whenever changes in its neighborhood are detected. And a large amount of this link state packet will then be generated due to the flooding mechanism. Apparently, as node mobility increases, it is more likely for a node to detect topology changes and then tremendous amounts of control packets are triggered.

The impact of mobility to routing inaccuracy is, however, distinct from the impact to control overhead. Overall, higher mobility causes higher inaccuracy for all three schemes. LS always performs the best in every mobility value, as shown in Fig. 3.6. LS sustains inaccuracy equal to or lower than 15% even at a node speed of 160 units per time slot, while DBF provides poorly acceptable routing solutions. Our GSR performs between DBF and LS: in the low speed range, GSR performs closely to LS; while in the high speed range, it becomes worse but is still better than DBF.

3.4.3.2 Update Interval

The update interval plays an important role for the routing overhead and inaccuracy. As we show earlier, LS provides a highly accurate routing solution because update packets are sent out immediately whenever a node detects a topology change. There is no delay for such updates. Thus, the update interval in LS is in fact the system minimum time resolution. For GSR, Fig. 3.18 shows that the inaccuracy is degraded or improved by adjusting the routing interval up or down. It is obvious that the same improvement holds for DBF, except that the improvement is not very significant as the accuracy is already poor even in the low mobility conditions (Fig. 3.19). As expected, we note that more improvements can be observed when mobility is high.

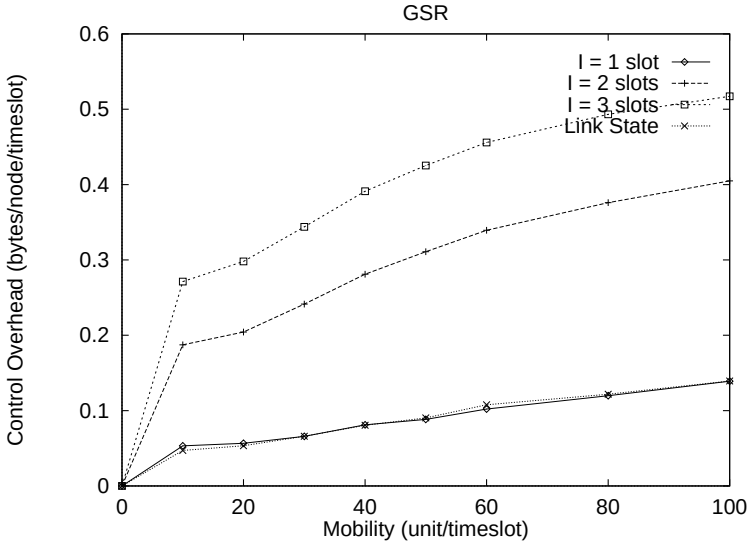

Figure 3.18: Inaccuracy at different update intervals: GSR

For GSR, the improvement of control overhead when using longer update interval can be clearly observed in Fig. 3.20.

3.4.3.3 Radio Transmission Range

The range of radio transmission determines the degree of node connectivity. As shown in Fig. 3.21, the larger the transmission range the larger the connectivity degree. Thus the larger the control packet size for GSR and LS, as we pointed out in in Table 3.1.

Fig. 3.22 shows the decrease in the routing error rate as the transmission range increases. It is because that a larger transmission range implies that more nodes can be reached in one hop without requiring a routing decision. However, as indicated in [GT95], the spatial reuse is less efficient when the transmission range is large.

It is interesting to note that in Fig. 3.22, the worst case doesn't happen when $R = 80$, which is the smallest range in our simulation. Instead, it happens at about $R = 150$, regardless of what the mobility is. This is because that the average hop distance

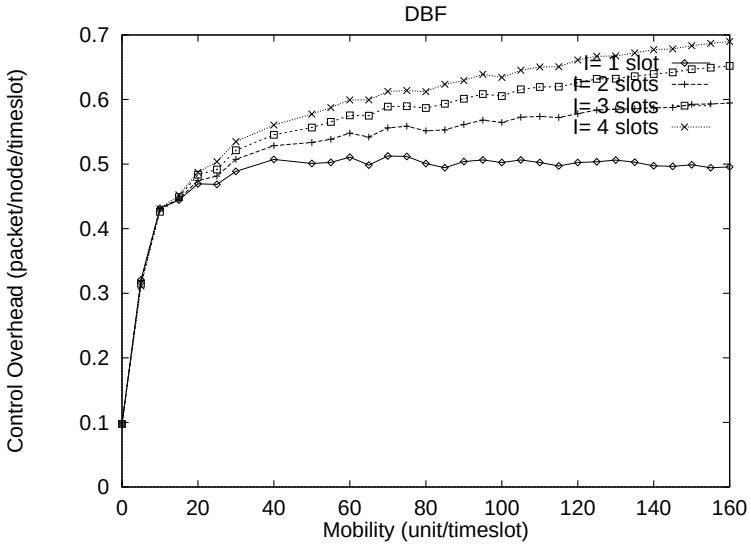

Figure 3.19: Inaccuracy at different update intervals: DBF

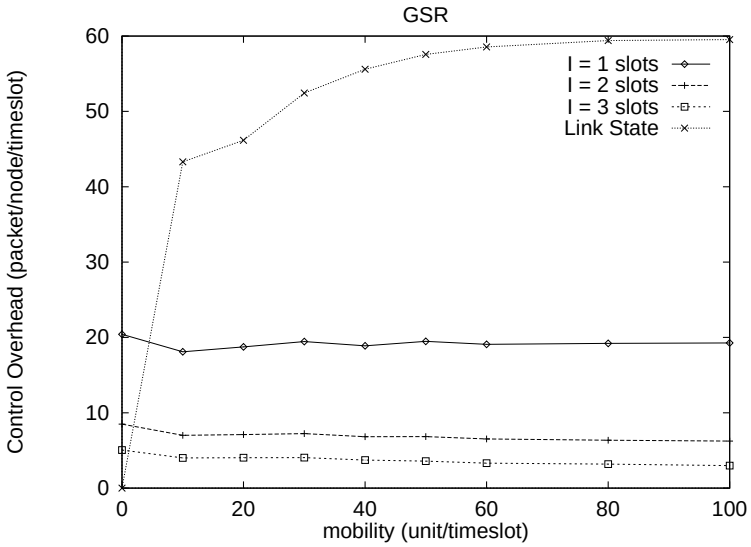

Figure 3.20: Overhead at different update intervals: GSR

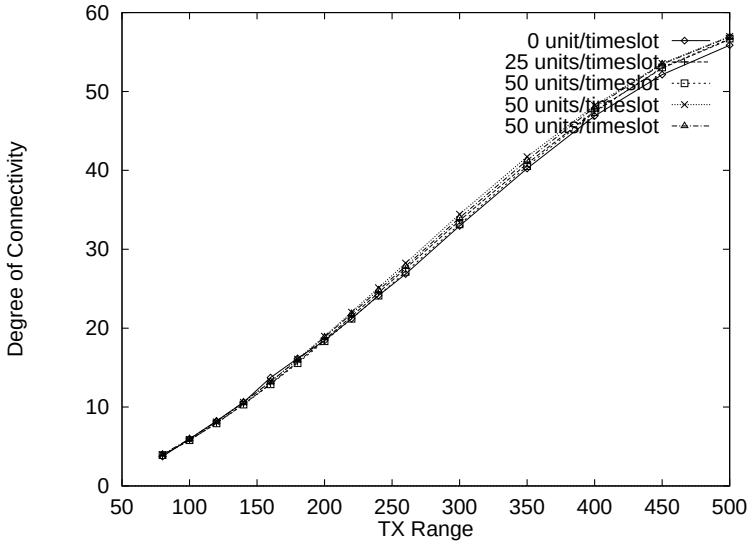

Figure 3.21: Connectivity vs. TX. range

increases as transmission range decrease. That is, more hops are required to reach a destination. Therefore, routing errors at far away destinations are more likely to be eliminated by the inaccuracy weighting function.

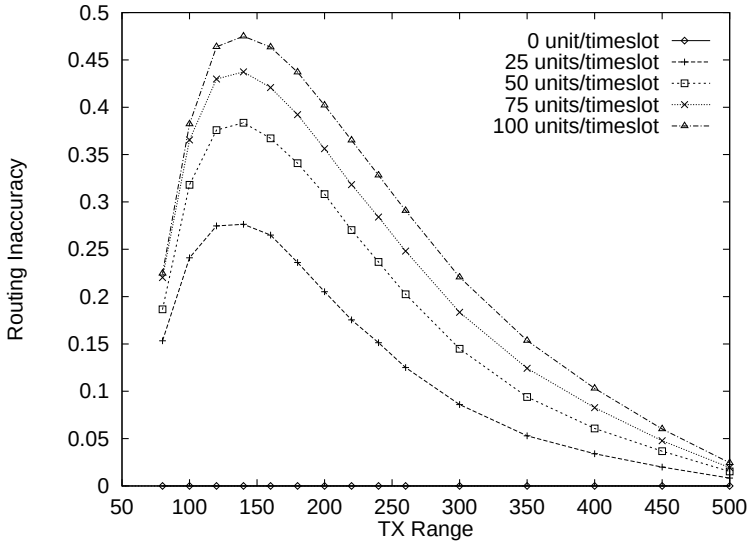

Figure 3.22: GSR: Inaccuracy vs. TX. range (I=3)

3.5 List of Detailed Algorithms

proc *NodeInit*(i) $\equiv$

foreach $j \in V$ **do**

$A_i(j) \leftarrow \phi$;

$D_i(j) \leftarrow \infty$;

$NEXT_i(j) \leftarrow -1$;

$SEQ_i(j) \leftarrow -1$;

od

$A_i \leftarrow A_i \cup \{x \mid link(i,x)$ exists$\}$;

$TT_i.LS(i) \leftarrow A_i$;

$D_i(i) \leftarrow 0$;

$NEXT_i(i) \leftarrow i$;

$t_i \leftarrow 0$;

$SEQ_i(i) \leftarrow t_i$;

.

```
proc Node(i) ≡
  NodeInit(i);
  while TRUE do
      if PktQueue ≠ φ   !! packet received
        foreach pkt ∈ PktQueue do
          A_i ← A_i ∪ {pkt.source}
          if pkt.type = SeqNumVectType
            foreach x ∈ N do
              SeqVect_i(x) ← Min(SeqVect_i(x), Pkt.seq(i))
            od
          fi
          if pkt.type = UpdateMessageType
            PktProcess(i, pkt)
          fi
          if pkt.type = DataPktType
            PktForward(Pkt)
          fi
        od;
      fi
      FindSP(i);
      if (clock() mod UpdateInterval) = 0
        RoutingUpdate(i);
      fi
      CheckNeighbors(i);
      TT_i.LS(i) ← A_i;
  od
.
```

proc *RoutingUpdate*(i) $\equiv$

$t_i \leftarrow t_i + 1$;

$TT_i.SEQ(i) \leftarrow t_i$;

$TT_i.LS(i) \leftarrow \phi$;

foreach $x \in A_i$ **do**

$TT_i.LS(i) \leftarrow TT_i.LS(i) \cup \{x\}$;

od

message.*SenderId* $\leftarrow i$;

message.*TT* $\leftarrow \{i, TT_i\}$;

foreach $x \in N$ **do**

if (*clock*() mod *GlobalUpdate*) $\neq 0$

if $(SeqVect_i(x) < TT_i.SEQ(x)) \ \wedge \ (D_i(x) < FishEyeScope)$

message.*TT* $\leftarrow$ *message*.*TT* $\cup \{i, TT_i.LS(x)\}$;

fi else

message.*TT* $\leftarrow$ *message*.*TT* $\cup \{i, TT_i.LS(x)\}$;

fi

od

broadcast$(j,$ *message*$)$ to all $j \in A_i$;

.

proc *FindSP*(i) $\equiv$

Dijkstra's shortest-path algorithm

$P \leftarrow \{i\}$;

$D_i(i) \leftarrow 0$;

foreach $x \in \{j \mid (j \in V) \wedge (j \neq i)\}$ **do**

if $x \in TT_i.LS(i)$

then $D_i(x) \leftarrow$ *weight*(i, x);

NEXT$_i(k) \leftarrow k$;

else $D_i(x) \leftarrow \infty$; *NEXT*$_i(k) \leftarrow -1$;

fi

od

while $P \neq V$ **do**

foreach $k \in V - P, l \in P$ **do**

Find (l, k) such that

weight$(l, k) = \min\{D_i(l) +$ *weight*$(l, k)\}$;

od

$P \leftarrow P \cup \{k\}$;

$D_i(k) \leftarrow D_i(l) +$ *weight*(l, k);

NEXT$_i(k) \leftarrow$ *NEXT*$_i(l)$;

od

.

proc *PktProcess*(i, pkt) $\equiv$

source $\leftarrow$ *pkt*.*source*;

$TT_i.LS(j) \leftarrow TT_i.LS(j) \cup \{$*source*$\}$;

foreach $j \in V$ **do**

if $(j \neq i) \wedge (pkt.SEQ(j) > TT_i.SEQ(j))$

then begin

$TT_i.SEQ(j) \leftarrow pkt.SEQ(i)$;

$TT_i.LS(j) \leftarrow pkt.LS(i)$;

end

fi

od

.

proc *CheckNeighbors*(i) $\equiv$

foreach $j \in A_i$ **do**

if $weight(i, j) = \infty$

$A_i = A_i - \{j\}$;

fi

od

.

CHAPTER 4

Routing with QoS Reports

Up to now, most of the routing protocols that have been proposed for ad hoc wireless networks optimized the solution for only one metric: hop distance. For datagram traffic, single metric routing based on hop distance may be sufficient. However, when multimedia traffic is concerned, these single metric routing schemes may cause network congestion on some particular links thus the QoS is degraded.

Based on this consideration, we figure that in order to provide QoS support, it is necessary to effectively control the total traffic that can flow into the network system. And the key to a successful admission control is QoS routing. The goals for QoS routing are two-fold. First, the QoS routing schemes can help admission control. That is, routing protocol not only provides route to destination, but also computes the QoS that is supportable on a route during the process of route computation. The network control mechanism decides whether to accept a new connection request by examining whether the route given by the route finding scheme still has sufficient QoS to adapt this new connection. Secondly, QoS routing schemes that consider multiple constraints provide better load balance by allocating traffic on different paths, subject to the QoS requirement of different traffic.

In this chapter, we present our approaches to achieve the goal of QoS routing. These approaches include QoS extensions in DBF type routing, and a heuristic multi-metric QoS routing that based on GSR.

4.1 Motivation

Here, we briefly discuss our motivation and the effectiveness of the proposed QoS routing approach in attacking the multihop design problems outlined in section 4.2.

Connectivity and QoS maintenance: Routing and QoS information is computed and disseminated using a distributed algorithms (DBF+QoS), which is robust to node failures and to topology changes. Convergence to the optimal solution at steady state is guaranteed. Loops are prevented during transients. The times to converge is somewhat slower than in conventional distance vector or link state algorithms. However, our experiments show that performance is adequate even at sustained speeds, up to 50 unit/timeslot. The link and processing overhead is about twice that of conventional distance vector algorithms since both distance and bandwidth vector must be now transmitted and processed.

VC connection setup/maintenance: The knowledge of available bandwidth to destination enables the source to exercise CAC (Call Admission Control), thus avoiding network overload. Furthermore, the source can adjust the voice/video code parameters (e.g., quantization, frame rate etc.) to meet the available bandwidth constraints. Once a VC connection has been set up, if the primary path fails (because of mobility or channel degradation), alternate routing provides instant backup (assuming that the alternate path is disjoint from the min hop path).

Congestion prevention: At call set up time, bandwidth information permits to enforce effective CAC and to renegotiate rates, if necessary. Furthermore, during the life of a connection, when the primary path fails because of motion, the alternate routing and selective discarding features help alleviate congestion by effectively rerouting traffic and reducing the rate at the same time.

Multicast support: Most of the existing multicast routing schemes (e.g., the MBone DVMRP, PIM, etc) as well as multicast signaling and reservation schemes (e.g., RSVP) rely on an underlying routing protocol. In the case of multimedia traffic, the routing protocol must support QoS (more specifically, bandwidth) guarantee. Thus, the proposed QoS routing algorithm is ideally suited to extend to the wireless domain all the various multicast schemes available in the wired network. In particularly, for multicasting within the wireless network, a novel scheme, inspired to CBT (Core Based Trees), was been developed, which relies on QoS routing and exploits the cluster structure of the network [CG].

Wired net interconnection: When a VC is being set up from the wired net (say, ATM) to a destination in the wireless net, the gateway (e.g., base station) can renegotiate QoS (on the basis of the bandwidth advertised by the wireless network) in order to correct the possible wired/wireless mismatch. For example, the rate of the source can be reduced, by reducing quantization levels for example. If the connection is a multicast connection originating from the wired network, it may be undesirable to reduce the rate for the entire multicast group. Instead, rate adaptation via selective discarding of high resolution substreams (generated by multilayered encoding and carried by separate VCs) can be performed at the gateway. Selective discarding can also be used on an already established connection, to adjust to dynamic bandwidth fluctuations in the wireless network.

Soft handoff: In a multihop wireless network connected to the wired network via gateways, the handoff procedure from one gateway to the next (as the user moves) is quite different from that used in single hop, cellular networks. In the multihop case, the switch-over is determined not by received signal strength, but by hop distance and available bandwidth to the next gateway. In fact, handoff in the cluster TDMA is

simpler than in the cellular case because the dynamic clustering algorithm automatically change in signal strength, by reassigning the user to a new cluster. Furthermore, the routing algorithm maintains the ”old” connection while the user is negotiating the ”new” connection through the next gateway. Once the user decides to switch over to a new gateway, it signals the source (in the wired network) to open a new VC through the gateway. For a short period of time, both connections (via the old and the new gateway) will coexist, until the old one is timed out. The ability to communicate bandwidth information to the wired source is important in the case of transition between heterogeneous wireless environments (e.g., from indoor high speed wireless LAN to outdoor low speed wireless MAN). By communicating the bandwidth expected in the wireless MAN, the user warns the source to reduce the rate, thus assuring soft degradation during the indoor to outdoor transition.

4.2 VC Management

In order to support multimedia service, it is necessary for a network to provide connection oriented scheme so that the resource can be computed and reserved. To this extend, most researches adapt the “soft state”, or called “Virtual Circuit” method to achieve this goal. In WAMIS [GT95], Cluster/TDMA and VC with Fast Reservation is proposed for transporting real time traffic. In the following discussion, we assume such scheme is used in our ad hoc wireless model.

The Cluster/TDMA provides a easy solution for bandwidth reservation. This is because gateway which connects neighboring clusters alternate between different spreading codes from slot to slot. For example, the gateway may receive or transmit on code C1 in slots $\{1,3,5,7,9\}$ and receive/transmit on code C2 in slots $\{2,4,6,8,10\}$. Note that in our routing implementation the gateway can only forward packets across clus-

ters (i.e., it cannot be used to relay packets internally in a cluster). It then follows that for any path crossing the gateway there cannot be common free slots between incoming (into the gateway) and outgoing (out of the gateway) link.

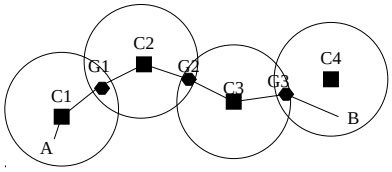

Figure 4.1: Path crossing several clusters

This property greatly simplifies the computation of available bandwidth on a given path as well as the computation of shortest paths with bandwidth constraints. In fact, consider a generic path carrying several cluster, as shown in Fig. 4.1. Within each cluster, there is still the possibility of free slot overlap between incoming and outgoing links (through the cluster head). This overlap, however can be easily accounted for locally, allowing us to compute the available bandwidth within each cluster, eg. $BW(G2, G3)$ in Fig. 4.1 Across clusters, there is no free slot overlap. Thus the end to end bandwidth is simply given by the minimum of the intra-cluster bandwidth. In our example:

$$BW(A,B) = min\ (BW(A,G1),\ BW(G1,G2),\ BW(G2,G3),\ BW(G3,B) \tag{4.1}$$

An important corollary of EQ. 4.1 is that we can now apply polynomial time bandwidth constrained shortest path algorithms such as described in [Ger86]. That is, given a target bandwidth B, we can compute the shortest path between two arbitrary nodes with bandwidth $\geq$ B (assuming such path exists) in $O(HlgH)$ time, where H is the number of nodes. Moreover, as shown in [Ger86], we can compute all paths between a given node pair, routed by increasing hop distance and bandwidth.

4.3 QoS Extension for Distributed Bellman-Ford

As well known, distance vector routing schemes, such as distributed Bellman-Ford, are subject to looping and counting-to-infinite problems [PB94]. In a mobile wireless network, while topology changes are caused by the host motion rather than the link exception, the looping induced by topology changes can severely affect performance. Thus, it is imperative to use a loop free routing scheme. Several distance vector loop free schemes have been reported in the literature [Gar89b][PB94]. We built a QoS routing protocol based on DSDV (Destination Sequenced Distance Vector)[PB94], because it provides loop-freedom while making minimum modification to the B-F routing structure.

Before we describe our QoS extension, let us introduce the DSDV in brief. In DSDV, each routing table entry carries hop distance and next hop for all available destinations (as in B-F). In addition, each entry is tagged with a sequence number which originates from the destination station. The routing information is advertised by broadcasting periodically and incrementally. Upon receiving the routing information, routes with more recent sequence numbers are preferred as the basis for making forwarding decisions. Of the paths with the same sequence number, those with the shortest hop distance will be used. That information (i.e., next hop and hop distance) is entered in the routing table, along with the associated sequence number tag.

When the link to the next hop has failed, any route through that next hop is immediately assigned an ∞ hop distance and its sequence number is updated. When a node receives a broadcast with an ∞ metric, and it has a more recent sequence number to that destination, it triggers a route update broadcast to disseminate the important news about that destination. In [PB94], the DSDV protocol is shown to guarantee loop-free paths to each destination at all instants.

The QoS extension to this DBF protocol is intuitively. In addition to the hop distances to all destinations, the routing table is now augmented with the information for available bandwidth observed on each node. The available bandwidth indicates how many extra bandwidth can be provided by the path leads to the destination. During the routing computation, DBF computes the shortest path based on the information provided by neighboring nodes. Similarly, the available bandwidth is also computed. With this information, each node decides whether to accept for connection request based on the available bandwidth to the destination. By doing so, the network QoS can be better preserved as the congestion is less likely to happen.

4.4 QoS Extension for Global State Routing

Since GSR maintains a global view of network status, the extension of QoS in GSR simply requires augmentation of QoS parameters for each link state entry. Unlike in DBF, where routes are computed in a distributed fashion, and detailed network information is lost during the information dissemination, GSR computes a more optimized route based on the global information. However, this computation for a QoS route that satisfies multiple addictive metrics (e.g. delay, distance, etc.) is an NP-complete problem [WC95]. To overcome this, we use a heuristic algorithm, which is similar to [Iwa96], to compute a sub-optimal route that also satisfies two metrics: end-to-end delay and bandwidth.

4.5 Simulation Results

The performance of the proposed QoS routing schemes was evaluated in a representative network example similar to the one used in previous chapter. We first study the

improvement of QoS routing based on DBF, then we show that result based on GSR. The improvement is measured by evaluating the real time traffics carried by the VC schemes. We assume a congested network so that each source of the VC is generating packet at a constant data rate. We monitor total packet throughput versus offered traffic load as stations move at various speeds. In order to compare, the same traffic patterns are also applied to a network performing standard min-hop routing.

Additionally, real time traffic is modeled by a continuous source which generates data packets at constant rate. Because of real time constraints, packets will be dropped if their "age" in the network exceeds the TTL (time to live) value. In our experiments, we study traffics of two types, TTL=5 and TTL=9, representing high delay sensitive traffic and low delay sensitive traffic, respectively.

4.5.1 DBF

As described before, based on the information provided by QoS routing (i.e., number of slots available on the shortest path to destination), a node can determine whether to accept a new call. During VC rerouting following a change in topology, congestion may arise if there is no sufficient bandwidth to support the initial QoS. In this case, queues tend to grow large. Based on packet age, expired packets are dropped.

The results are reported in Fig. 4.2. As expected, DBF with QoS extension achieves higher reception rate than standard DBF (in terms of packets received). With DBF+QoS routing, the packet received ratio is about 5% more than standard DBF for traffic with TTL=5. Also as expected, less significant improvement is observed for the less QoS-sensitive traffic (TTL=9).

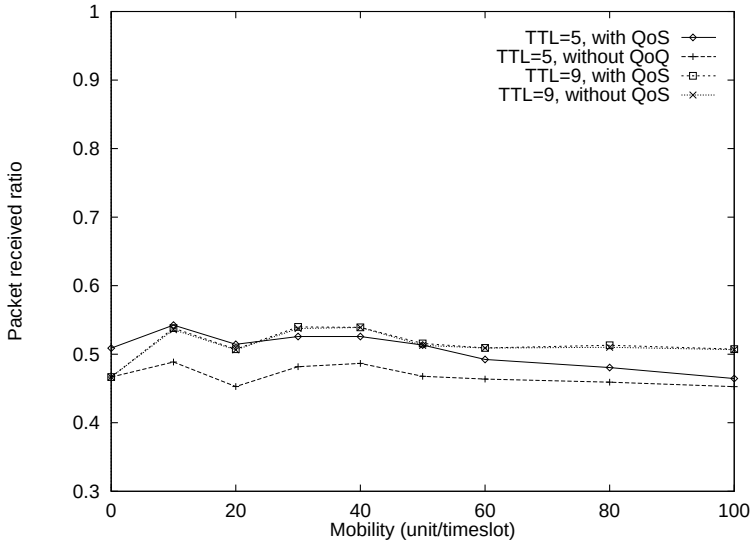

Figure 4.2: Packet Received Ratio for DBF+QoS

4.5.2 GSR

Fig. 4.3 shows the results when QoS is applied to GSR. As shown in the figure, GSR+QoS does exhibit more substantial improvement in terms of packet received ratio. And since GSR provides more accurate routing information, the overall reception rate is also higher than DBF+QoS. More over, standard GSR without QoS achieves even higher throughput than DBF+QoS. This is also due to the help of routing with accurate network information.

If we examine the behavior as a function of mobile station speed, we notice that GSR+QoS performs better in low speed range. When the speed increases to 50 unit/timeslot, the improvements are less significant. This is because at high speeds (i.e., very high topology change rate), GSR is penalized by the effect of fisheye, which causes slower convergence for part of nodes (i.e. remote nodes) than it is in link state routing.

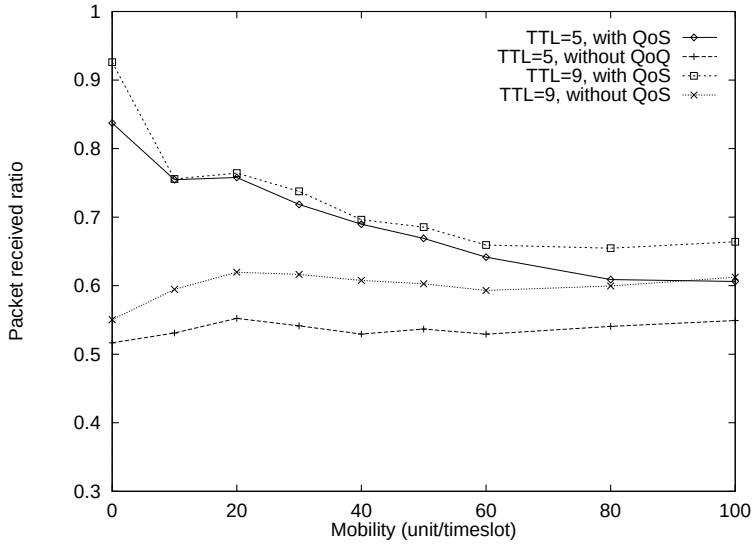

Figure 4.3: Packet Received Ratio for GSR+QoS

4.6 Summary

In this chapter we have briefly presented the results of applying QoS information to different routing schemes for ad hoc wireless networks, which includes an "augmentation" of existing loop free DBF routing algorithm, DSDV, and our proposed GSR. Our proposed GSR routing scheme with QoS extension can be more effectively used in the support of real time traffic in multihop wireless network, and find applications in two important scenarios: (a) stand alone, multimedia, multihop networks, and; (b) wireless, multihop extensions of an ATM network or an Internet. In either case, the QoS feature of the routing algorithm is instrumental in establishing and maintaining virtual circuits. In the ATM interconnection case, the QoS routing information can be used to assist in the handoff of the mobile host between different ATM base station.

The simulation results show that both DBF and GSR have improvements on network performance with this augmentation of QoS information. But more substantial improvements are observed in GSR, because of a more precise network information.

Also shown in these results is the tendency to low throughput in all schemes when mobility increases. This indicates that in a very high mobility environment, all sophisticated routing will fail; and flooding will probably be the most effective scheme for this situation.

CHAPTER 5

QoS Renegotiation

Even with efficient QoS routing in the network layer, it is still not sufficient to provide QoS guarantee at a constant level in ad hoc wireless networks. Hence, there is a need for renegotiating existing QoS on an established connection, since the characteristics of a wireless link may well change during the lifetime of a connection due to mobile hosts' movements or external interference. In this chapter we study the QoS issues from system aspect. We present a QoS scheme with renegotiation capability, define an API (application programming interface) for the access to this scheme and describe our implementation for this QoS API on the SWAN system, a wireless ATM network, and summarize its performance using measurements obtained from a series of experiments based on different fault scenarios.

5.1 Introduction

Wireless networking is inherently unreliable. Various forms of interference on the wireless link result in changing bandwidth availability and low effective bandwidths due to high error rates. These problems are exacerbated as users move around. Faults of this kind require a fresh look at how such networks can be used to support applications which demand some degree of predictability. We adopt the approach of ATM, in which QoS is used to form a service contract between applications and the network.

We build on that work by recognizing that an unreliable wireless network demands a more dynamic approach to resource usage. Many applications can deal with varying bandwidth availability once provided with sufficient knowledge of the resource climate. Typical examples include audio and video applications which can alter their rate or encoding to match the available bandwidth or deal with different error rates. Our contribution is a QoS scheme which builds on this notion of adaptation by providing explicit renegotiation. This is similar in spirit to the feedback mechanisms for non real-time traffic in ATM, but differs in that we aim to provide feedback right up to the application level, not just to the sending host [SM96]. Thus we incorporate renegotiation as a key part of our QoS API.

Four elements compose our approach. First, we engage the support for multiple VCs over a wireless channel, and the usage of a set of per-VC QoS parameters to influence bandwidth allocation. Second, we define a group of interface routines for opening, accessing and closing VCs, as well as being able to assign QoS parameters. These parameters include a description of the traffic type as constant bit-rate (CBR), variable bit-rate (VBR) or available bit-rate (ABR). CBR and VBR applications have real-time requirements, e.g. 64 Kbps speech and compressed video. ABR is used for more traditional applications which can accept much more variability in service. Consistent with our emphasis on adaptation, we accept values for preferred and minimum bandwidths for a VC. Third, we design a centralized QoS manager to coordinate the access to a wireless channel. Using parameters supplied via the API, the manager performs admission control, monitors performance on the channel and initiates renegotiation when necessary. Fourth, for each VC, the application provides a callback routine which is used by the QoS manager to provide feedback as part of renegotiation.

The QoS scheme described above is successfully implemented in a wireless ATM

network: SWAN [Agr96], developed in the Lucent Bell Laboratories.

5.2 The SWAN Environment

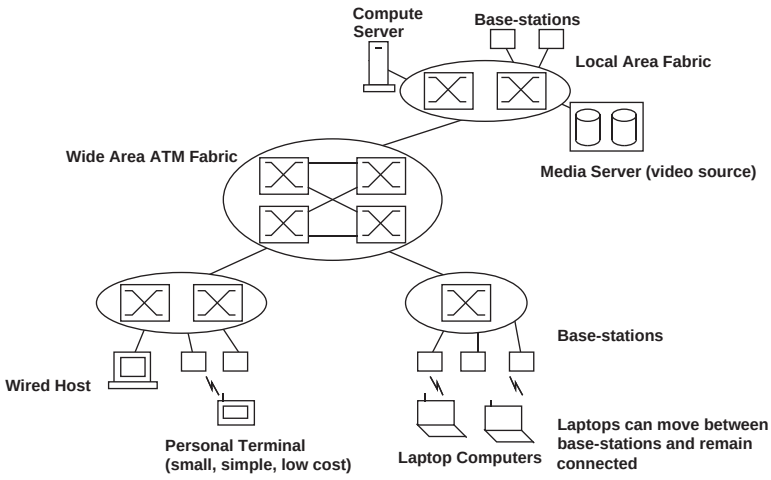

Figure 5.1: SWAN system architecture

The SWAN (Seamless Wireless ATM Network) system, shown in Fig. 5.2, is a testbed for wireless networked computing. SWAN consists of mobile units which are usually laptops, and base stations which are connected to a backbone network [Goo90]. Both the base stations and laptops are equipped with a radio interface known as the FAWN (Flexible Adapter for Wireless Networking) [TC94] card that allows them to communicate with each other wirelessly. Each base station has a range of 100 feet inside a building, providing access to a local area network for mobiles in their vicinity. As well as communicating with the base stations the mobiles can communicate with each other, allowing them to create ad-hoc networks that continually change as the mobiles move around.

The FAWN card provides a very programmable platform on which to develop interface software, which is important in a testbed. FAWN uses a 2.4 GHz ISM band

radio modem whose raw bit rate is 624 Kbps which is divided between incoming and outgoing connections. The modem has a raw error rate of 1×10^{-5} for a signal strength of -77 dB which translates to a packet loss of one in 1500 for our 64 byte packets. The FAWN adapter has four 64 byte packet buffers implemented in hardware to store buffer complete packets and therefore improve performance. The FAWN card is controlled by an ARM610 processor which takes the packets from the buffers, processes them and makes them available to a host computer via a PCMCIA interface.

A simplified diagram of SWAN's channel access scheme is shown in Fig. 5.2. A TDD (Time Division Duplex) scheme is used to share the bandwidth between the base station and the mobile host. The traffic of each direction alternatively transmits a data burst of 10 ATM cells at a time and then switches to receiving mode for data from the other direction. Due to the overhead introduced by the TDD scheme and the ATM cell structure used in SWAN, the available bandwidth is 240 Kbps in each direction.

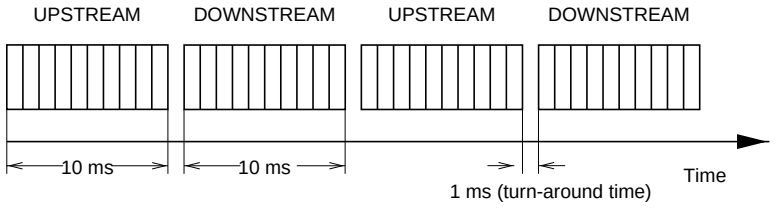

Figure 5.2: TDD scheme of SWAN

Several other wireless communication devices are available for the local area network market (e.g. WaveLAN, RangeLAN), and most of them are based on or akin to the IEEE 802.11 or CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) schemes. Though a CSMA/CA scheme simplifies the hardware implementation and provides reasonable efficiency in supporting datagram traffic, its random access characteristic cannot provide multimedia applications with a predictable bandwidth

available on the link. The SWAN system, as described previously, was designed with ATM traffic in mind. Its TDD scheme assures a constant bandwidth can be granted, at least when the channel condition is stable. Therefore, SWAN provides a good platform for realizing our proposed QoS scheme.

Unfortunately, despite the direct support of ATM cells and TDD MAC scheme, challenges still exist in SWAN when considering the support of multimedia traffic in a wireless, mobile environment. These challenges include (1) low radio bandwidth; (2) increasing likelihood of erroneous packets due to lower SNR; (3) lack of a mechanism to support traffic of different classes and (4) lack of an interface to provide link QoS information to upper layer applications. We will address our approach to these problems when describing our QoS implementation in the next section.

5.3 System Implementation

To achieve QoS renegotiation in a unreliable wireless environment, We design a mechanism that exists in the operating system level (the interface at which the applications request resources) for an application to request a grade of service for a network connection and for the application to be informed about changes on that connection. In this section, we describe our approach in detail from two aspects: (1) the policy, in which we determine how an application can specify its QoS requirements and how it can be notified of failures so it may adapt to a new environment; (2) the mechanism, where we address the realization of the policy. Fig. 5.3 shows the layout of SWAN system.

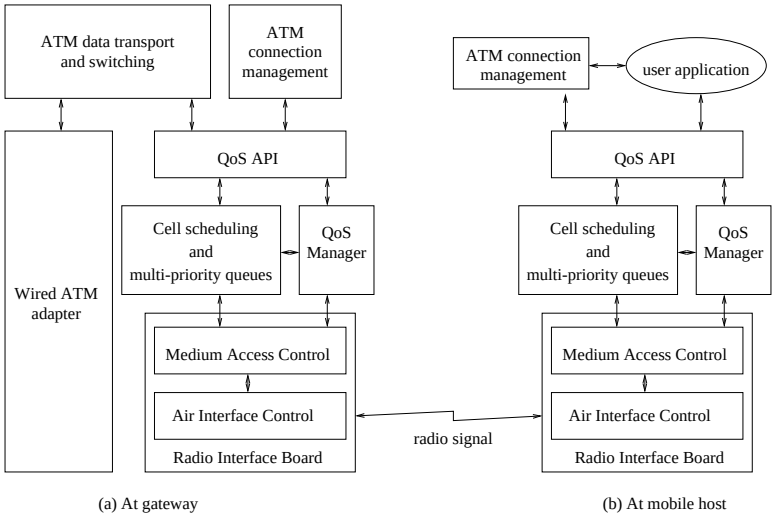

Figure 5.3: Layout of SWAN system software

5.3.1 Policy

Our goal is to use existing interfaces and facilities provided by widely accepted operating systems, instead of creating an ad-hoc system or proposing a new, proprietary interface. Therefore we chose Linux, a UNIX-like system, to develop our work. Also, since SWAN is designed to provide ATM connectivity, we consider the QoS negotiation on a per-VC basis. In our approach, the VCs are instantiated as UNIX devices, such that one may use the `open()` system call to obtain a VC and the `close()` system call to release the VC. During the connection, data is sent and received via `write()` and `read()` system calls.

As soon as a circuit is activated (opened), it is given a default service grade of unspecified bit rate (UBR) service. This allows applications that do not have QoS demands to receive the best service effort from the system. If an application does wish to specify its bandwidth need, it does so with one or more `ioctl()` system calls (I/O control).

By performing these `ioctl()` operations, an application may select ABR, CBR, or UBR services, and specify the associated QoS parameters. Currently, two bandwidth parameters (minimum and preferred) have been considered. Supporting these two bandwidth parameters allows an application to specify a range of acceptable bandwidth so that it doesn't get informed each time when the supported bandwidth changes.

The way an application should be notified of QoS failure is also considered. Using existing UNIX facilities, the signal mechanism allows the operating system to send a "QoS failure" message to the application. The application uses the `signal()` system call to setup an exception handler to process this QoS failure event. A similar policy exists for the reverse operation, where an application receives a signal when the service failure is removed and returns to its original performance.

5.3.2 Mechanism

In this section we describe in detail the realization for the above policy. We first introduce the usage and functionality of this API, by which the applications specify the QoS parameters associated with the VC. Then we draw the core of the implementation.

5.3.2.1 The API

The first aspect of creating the desired interface is to provide a device driver for the VCs and the associated API to manage them. The VCs are implemented as devices within the UNIX file systems to which the standard system calls can be applied. The entire API of our implementation is shown in Fig. 5.4. Since it is implemented using standard UNIX I/O operations, a user program can manipulate its connection just as an ordinary character device.

The QoS requests are made through the `ioctl()` system call with the application

specifying parameters for the type of service, minimum or preferred bandwidth, etc. The parameters we have implemented for the QoS negotiation are listed in Table 5.1. The default values shown in the table indicate that an UBR service is assumed to reserve the system minimum bandwidth, which is zero, if the application does not make any QoS request.

```
int open(char *vc_dev_name, int mode);
     /* acquire a VC, return -1 if requested VC is in use */
int close(int vc_des);
     /* release a VC */
int read(int vc_des, char *buff, int n);
     /* read n bytes from a VC */
int write(int vc_des, char *buff, int n);
     /* write n bytes to a VC */
int ioctl(int vc_des, int qos_request, long arg);
     /* request, negotiate a QoS attribute of a VC */
int signal(int QoS_SIGNAL, void *qos_handler(int));
     /* set up a handler for QoS changes */
```

Figure 5.4: List of the API

QOS_REQUEST	ARGUMENT	DEFAULT
VC_SERVICE	ABR,CBR,UBR	UBR
VC_MIN_BW	n (Kbps)	0 (Kbps)
VC_PREF_BW	m (Kbps)	0 (Kbps)

Table 5.1: ioctl(): parameters

With this API, the application can be easily programmed using a traditional client/server model: (1) the client and the server first request a VC using `open()`. (2) If the VC can be opened successfully, the required QoS can be provided by using `ioctl()` with parameter values based on the traffic characteristic. (3) Data is transmitted using

`read()` and `write()`. (4) When the program terminates, VCs should be released by using `close()`. The system routine `signal()` listed in Fig. 5.4 is not really a QoS operation; instead it is used by the application to setup its own QoS interrupt handler which can renegotiate new QoS agreements when the original service requirements cannot be met.

5.3.2.2 Multiple Queues Scheme

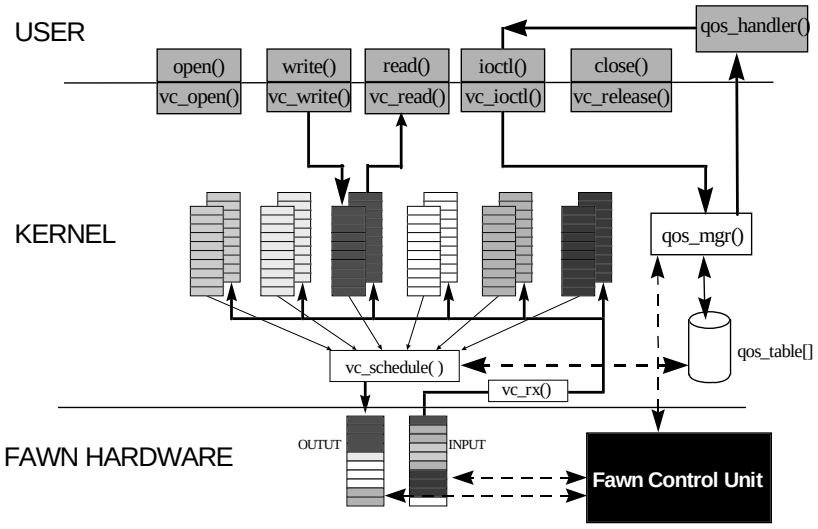

Figure 5.5: The architecture for multiple queues scheme

Fig. 5.5 illustrates the block diagram that shows the relationship between user applications and FAWN hardware, as well as the interaction among each module. On the top, applications in the user level communicate with the QoS mechanism through a set of interface routines (the API). A group of priority queues are dynamically allocated in the kernel space. Each queue corresponds to an individual VC. Once a VC is opened and its QoS is negotiated through `ioctl()`, which interacts with the QoS manager

(`qos_mgr()`) for service and bandwidth specification, the QoS manager translates the requested service type and bandwidth in terms of time slots for carrying data cells in each data burst. This information will be kept in a QoS table (`qos_table()`) that will later be referred by the VC scheduler (`vc_schedule()`). The QoS manager is also responsible for monitoring the overall link quality through the FAWN hardware, and providing feedback directly to application when the requested QoS can not be satisfied or when a better service is available. This feedback is implemented through the UNIX signal, as described in Section 4.2.1.

The VC scheduler reads packets from those activated queues and sends them to the FAWN hardware for transmission. It serves these multiple queues in a "round-robin" fashion which allows a control of QoS granularity such that one circuit will not dominate the data path with a large chunk of data.

5.4 System Functions

Currently, two major functions provided in this system are bandwidth reservation and QoS renegotiation. They are the keys to providing multimedia traffic support in wireless networks.

5.4.1 Bandwidth Reservation

In SWAN, the radio channel in use is shared between a base station and a mobile host in a TDD fashion. Thus the allocated bandwidth can be represented in terms of the number of time slots devoted to a connection. For example, a connection granted with one slot in each data burst is served at the bit rate of 24 Kbps (240 Kbps/ 10 slots) in FAWN's TDD scheme.

During bandwidth reservation, the QoS manager is responsible for converting the bandwidth requirements into the necessary number of time slots for transmitting the data to meet the bandwidth guarantees. If the time slots cannot be allocated, the bandwidth request will be rejected by the QoS manager. UBR service is provided by placing data in slots that are unreserved or unused by CBR/ABR circuits. In addition to providing service to queues of different QoS requirements, a starvation prevention scheme is also utilized to prevent starvation on UBR service. In this scheme, at least one data slot is reserved and shared among all UBR queues in a "round-robin" fashion so no UBR connection will be starved even if some of them are heavily loaded. This avoids any dominate usage of one application over the bandwidth of the wireless link.

5.4.2 QoS Renegotiation

In a wired network, QoS is usually guaranteed for the life time of each connection. In a wireless network with host mobility, however, such a guarantee is not realistic due to distance, noise or channel fading, etc. On the other hand, many multimedia applications have used algorithms that can adapt to bandwidths that users specify. For instance, several video transmission schemes (nv, vic, etc.) can adjust their resolution and frame rates to fit the bandwidth parameters that they are given; several audio applications may adjust their sampling rate and level of quantization based on the channel bandwidth. Such properties have not been utilized for a self-adjusting multimedia application because of the lack of QoS feedback from the traditional packet network like the Internet, or even some ATM networks.

In our work, the signaling mechanism we propose informs the applications of the changes in QoS. The applications can benefit from this mechanism by simply setting up interrupt handling routines so that when they are notified of a change in QoS, they

can adjust their data transmission algorithm based on the current QoS information. This signaling mechanism was implemented in the QoS manager which has a direct access to the FAWN hardware to learn about the current status of radio link. For example, when the QoS manager detects the decrease in radio bandwidth, it first reduces the service to the UBR traffic; If such a reduction is not sufficient to guarantee the requested bandwidth for all ABR and CBR traffics, it then reduces the ABR/CBR service rate to its minimum requirement. Finally if the bandwidth is still not sufficient, the QoS manager will prorate the assigned bandwidth on all CBR and ABR connections and signal the corresponding handlers created by the applications to notify the change of QoS. Upon receiving the signal from the QoS manager, the handler in each application can decide whether to accept the newly assigned QoS, to terminate the connection, or to renegotiate a new QoS through the provided API.

5.5 Fault Tolerant Experiments and Analysis

We conducted two experimental studies to verify the implementation of our QoS scheme, and assess the effectiveness of this scheme as a fault tolerant mechanism in the presence network failures. These experiments are described as follows.

5.5.1 QoS Renegotiation Experiment

The first experiment studies the effect of signal to noise ratio on a wireless link in the SWAN system. As signal to noise ratio decreases the number of erroneous packets received increases, which maps to a decrease in available bit rate over the link. We plot the performance of the system as the bit rate decreases (in other words as the error rate increases) by measuring the traffic through the system for a system based on UDP

datagram transmission (non QoS system) as well as our QoS based system.

In the experiment we assume that there are three data streams, A, B and C sharing the wireless link from a mobile to a base station that is connected to the network. Stream A is an ABR stream like an *ftp* file transfer which can use as much data rate as possible up to some maximum. In our experiment this maximum was 48 Kbps. Streams B and C are CBR streams, like those used in an uncompressed video transmission. Stream B needs 72 Kbps and stream C 96 Kbps. Stream C can operate at the lower bit rate of 48 Kbps if it is informed of the change. It can achieve this by reducing the number of frames per second that it sends.

5.5.1.1 Variation of Throughput for the Non-QoS Case

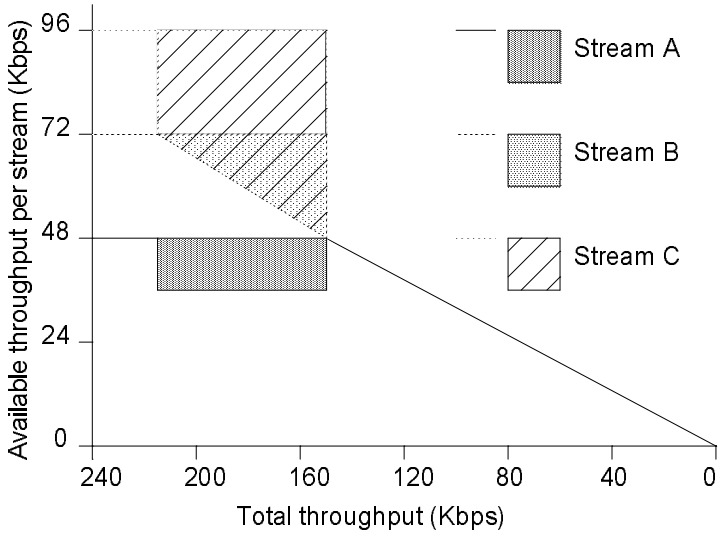

Figure 5.6: Realizable throughput for a system without QoS

The graph in Fig. 5.6 shows the variation of actual throughput versus the available data throughput for each of the streams A, B and C. The system's available data

throughput varies along the *x* axis from 240 Kbps down to zero. The shaded regions in the graph indicate a range of possible bitrates that data streams achieve, and the actual data rate tends to oscillate between the maximum and minimum values in each region. The three streams, A, B and C are presented to the communication channel. In the first region from 240 to 216 Kbps of throughput, streams A, B and C (whose total requirement is 216 Kbps) are accommodated.

In the second region, from 216 through about 150 Kbps, the total bandwidth requirements of all the channels cannot be satisfied, and they begin to interfere with each other. The scheduler attempts to give one third of the total data rate to each of the channels. Streams B and C can consume the third that they are given. However, stream A under-utilizes the available data rate because it only needs 48 Kbps, while a third of the bandwidth in this region varies from 72 to 96 Kbps. This means that there is spare throughput that streams B and C attempt to use. Both streams have the potential of getting their full bandwidth at some instances of time, thus the minimum bandwidth in this region is set by a third of the available data rate and the maximum bandwidth by the maximum data rate that can be sent by each stream. Even though stream A's data rate requirements do not exceed one third of the available data rate, it is interfered with by streams B and C which are using some of that capacity by putting large packets in the single queue which delay stream A's packets.

The final range is from 144 through 0 Kbps. Here all three streams can consume a third of the available bandwidth and compete for the bandwidth evenly.

5.5.1.2 QoS Version

In our implementation of the QoS scheme, all the queues are sharing the available bandwidth of 240 Kbps (with no errors). As the number of errors increases over the

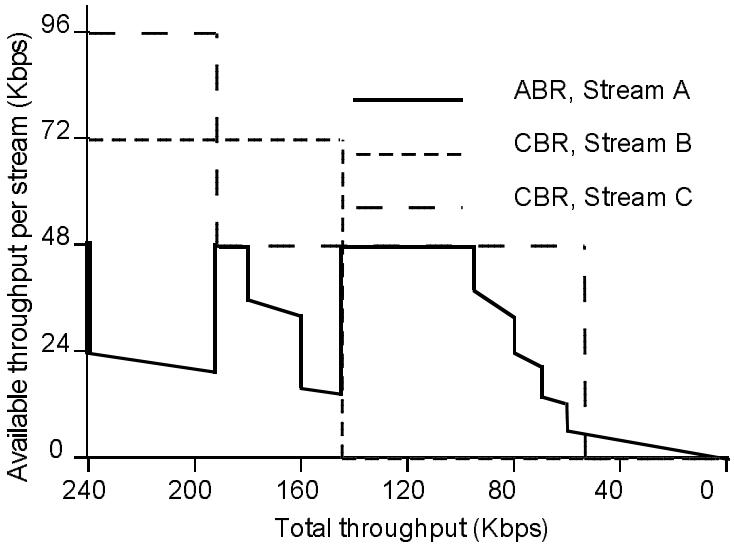

Figure 5.7: Realizable throughput for a system with QoS

link the effective data rate for each of the queues decreases proportionately. Because the QoS scheme allocates bandwidth on a slot basis the granularity of the available data rate is one tenth of the total available bandwidth.

The graph in Fig. 5.7 shows how each of the streams A, B and C respond to variations in the available bit rate. When there are no errors on the link stream A can operate at 48 Kbps (consuming two of the ten available timeslots), stream B operates at 72 Kbps and stream C at 96 Kbps. As the error rate increases slightly both streams B and C need extra timeslots to continue to be provided with their required bit rate. One slot comes from slack in the system (only 9 of the 10 were in use initially) and the other comes from stream A, which is an ABR stream and is downgraded to 1 timeslot. As the data rate reduces further, eventually stream C is unable to have its requested 96 Kbps. At this point the QoS manager sends it a signal telling it to renegotiate its required bit rate, and since it can operate at 48 Kbps it does so. At about 50 Kbps stream

C needs 3 timeslots to provide the 48 Kbps. This allows stream A to use another two timeslots for its ABR traffic. As the error rate increases the two CBR streams consume more timeslots, and correspondingly the bandwidth available for stream A reduces. At an error rate of about 95 Kbps stream B needs another timeslot to satisfy its data rate needs. However, since stream A always needs at least 1 timeslot and the CBR traffic of stream B cannot support a lower bit rate, it is renegotiated to zero. This makes more bandwidth available for stream A, but as the error rate further increases it gives that bandwidth to the CBR stream C which eventually stops when the error rate rises to about 185 Kbps, when the throughput falls to 55 Kbps.

5.5.2 Network Link Failure Experiment

The objective of the next experiment is to examine whether our QoS VC architecture and implementation scheme truly provides the required fault tolerant mechanisms in delivering the service it guarantees, and to compare the results with a network without QoS assurance. In this experiment, we consider link faults due to the failure of radio interface, particularly when the mobile stations move out of the radio range. That is, the quality on a failed link will degrade to a level where no data can be transmitted, and therefore traffic needs to be rerouted to another link in order to maintain session continuity.

5.5.2.1 Network Topology and Traffic Flow Assumptions

We consider a multi-hop wireless topology based on SWAN environment as shown in Fig. 5.8. There are four nodes in this network topology: Node A, Node B, Node C, and Node D. Five SWAN radio links are set up for communication between these nodes, marked as Link 1 through Link 5 in Fig. 5.8. Without losing generality, we made the

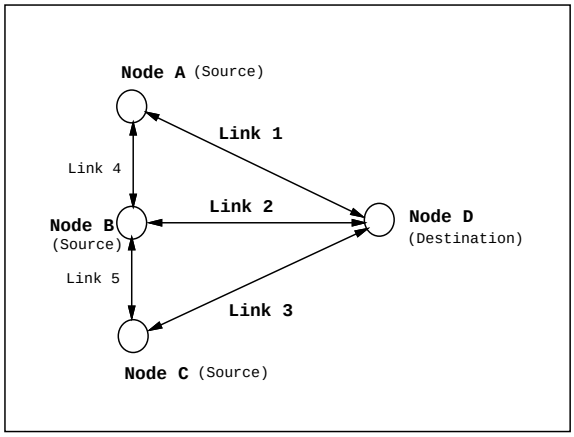

Figure 5.8: Topology for a network link failure experiment

following assumptions during the our experiment:

1. Nodes A, B, and C are source nodes while node D is a destination node.

2. Initially, Node A sends a CBR traffic (CBR video 1, abbreviated as *v1*) to Node D via Link 1, Node B sends an ABR traffic (ABR datagram, abbreviated as *d1*) to Node D via Link 2, and Node C sends a CBR traffic (CBR video 2, abbreviated as *v2*) to Node D via Link 3.

3. Both *v1* and *v2* are uncompressed video sessions transmitted at frame rate of 0.5 frame/sec and 1 frame/sec, which yield the constant bit rate of 73 Kbps and 145 Kbps, respectively. The traffic *d1* is designed to represent the ordinary datagram traffic thus we assume it may consume all the bandwidth that is left available.

4. Links 4 and 5 are robust rerouting links in the presence of link failures. When Link 1 fails, *v1* from Node A will be routed to Node B via Link 4, then delivered to Node D via Link 2. Similarly, Node C will redirect *v2* to Node D via Link 5 and Link 2 in the presence of Link 3 failure.

5. When Link 2 fails, the ABR traffic from Node B will be redirected to Node D through Node A (not Node C).

Note these nodes could communicate with other network components (not shown here) via wired links or other wireless links.

5.5.2.2 Impact on bandwidth utilization

To examine the impact on bandwidth utilization on a particular SWAN link, we conduct an experiment to measure transmission efficiency when different traffic sources have to be rerouted to share bandwidth of another link in the presence of link faults. In this experiment, the event of fault on Link 1 is at time 15th sec., and later the fault is recovered at time 85th sec. The event of fault on Link 3 starts at time 35th sec., and its recovery happens at time 130th sec. During the link down time, the associated traffic is rerouted to Link 2, based on the decision made by the network routing function.

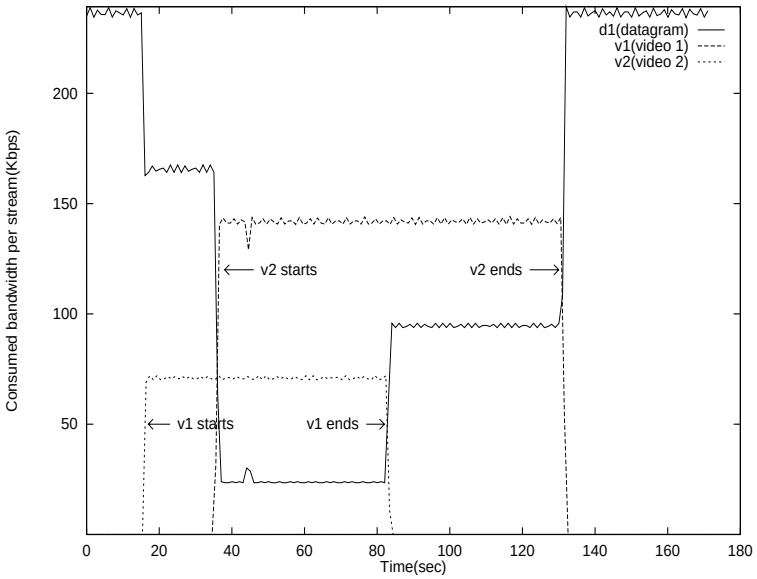

Figure 5.9: Received bandwidth in Link 2 using VC-QoS

Fig. 5.9 shows the effect of link failure to the bandwidth usage on Link 2 with respect to various traffic sessions. At the beginning, traffic *d1* is able to use up all the bandwidth until Link 1 fails. When Link 1 fails, traffic *v1* is rerouted to Link 2 by the routing mechanism. The QoS scheme on Link 2 will then allocate bandwidth used by *d1* to *v1*, since ABR has lower priority than CBR. Similarly, when Link 3 fails, *v2* is granted the required bandwidth after rerouting and *d1* can only use the bandwidth that is left after *v1* and *v2*. In Fig. 5.9 we also see that *d1* regains bandwidth after the recovery of Link 1 and Link 3.

As a comparison we repeat the same experiment using a non-QoS scheme (UDP/IP). The result is shown in Fig. 5.10. In this figure, we observe that due to the lack of a QoS mechanism, the amount of bandwidth that a connection can utilize is related to how aggressive the traffic source is. As we have described, *v1* generates data at 73 Kbps, which is much less aggressive than *d1*. Therefore between time 15th and the 85th sec., the quality of *v1* suffers tremendous fluctuations by having to compete with *d1*. Since *v2* is more aggressive (about 145 Kbps) than *v1*, thus between the 35th sec. and the 85th sec., the observed bandwidth shows that all these three sessions get about 1/3 of the bandwidth (Although *v1* is in fact slightly less than the other two). When *v1* stops at time 85th second, *v2* and *d1* both get half of the bandwidth. Also note the fluctuation between time 15th and 35th sec. is more significant than that between time 85th and 130th sec. This is because *v1* transmits video frames slower (0.5 frame/sec) and tends to fall behind the competition with *d1*. However *v2* transmits video at a faster pace (1 frame/sec), so it can share the bandwidth with *d1* more competitively. Moreover, due to the overhead of UDP/IP headers, the maximum bandwidth observed by the receiver here (220 Kbps) is less than previously (240 Kbps) when using QoS VC scheme.

As clearly observed, in this UDP experiment where no QoS is guaranteed, and

packets are rerouted correctly after link faults, none of the video sessions get the bandwidth they require. Consequently, they become unattractive in their real-time applications. Furthermore, video session *v1* suffers great fluctuations during its link fault, making the bandwidth it grabs from another link annoying and intolerable to its receivers.

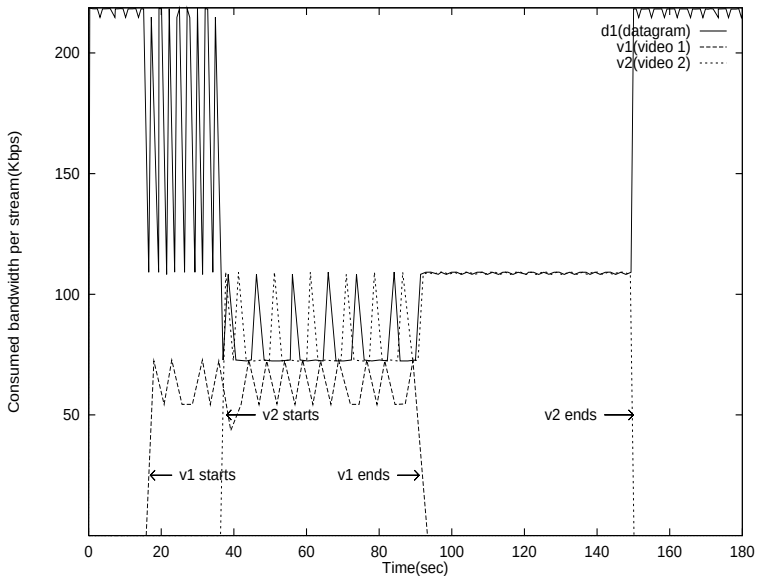

Figure 5.10: Received bandwidth in Link 2 using UDP

5.5.2.3 Impact on Delay Jitter

We also examine the impact to the delay jitter due to a link fault. This time, however, we consider the fault on Link 2 that carries ABR traffic. According to assumption 5 in Section 5.5.2.1, the routing function in network layer reroutes the ABR traffic from Link 2 to Link 1 after the link fault occurs. The distributions of inter-frame delay of the traffic *v1* carried by Link 1 before and after the traffic rerouting is shown in Fig. 5.11 and Fig. 5.12.

Fig. 5.13 and Fig. 5.14 present results performed under the UDP protocol for the purpose of comparison. The results are as expected: under the QoS VC scheme, the delay jitter is well controlled even in the presence of the extra traffic due to link fault. On the other hand, using the UDP protocol experiences a completely different result. *V1* has a decent delay distribution before Link 2 fails (Fig. 5.13); but once Link 2 fails and *d1* are rerouted to Link 1, the delay jitters exhibit uncontrolled and unexpected delays(Fig. 5.14). Table 5.2 summarizes the results obtained in Fig. 5.11 through Fig. 5.14. We see the UDP (with no QoS) experiences severe delay jitter problems (23.6% overhead) in a heavy traffic situation. Our QoS VC mechanism, on the other hand, is very stable and efficient. Though in Table 5.2 it seems the QoS VC scheme introduces more overhead (2.6%, computed as the extra delay in relative percentage to the UDP with no background traffic) than the UDP does when no other load is added, overhead introduced by the QoS VC scheme is still less significant then the overhead caused by the TCP/UDP/IP headers.

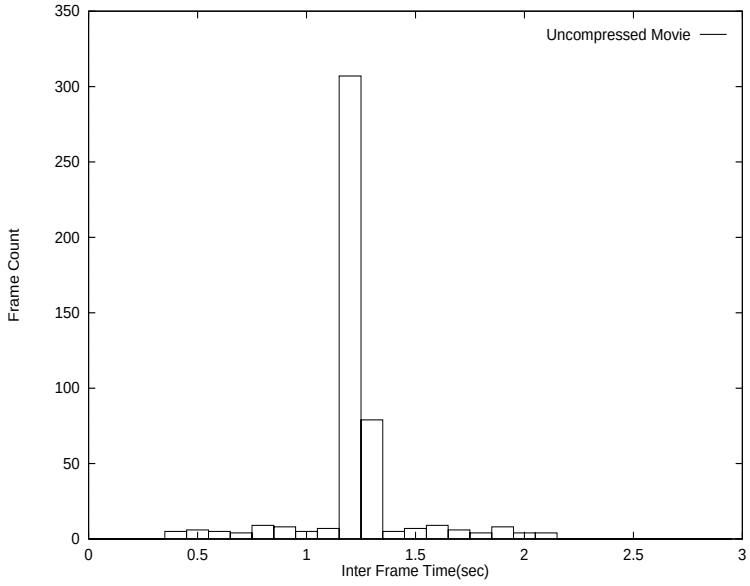

Figure 5.11: Before Link 2 fails, with QoS support

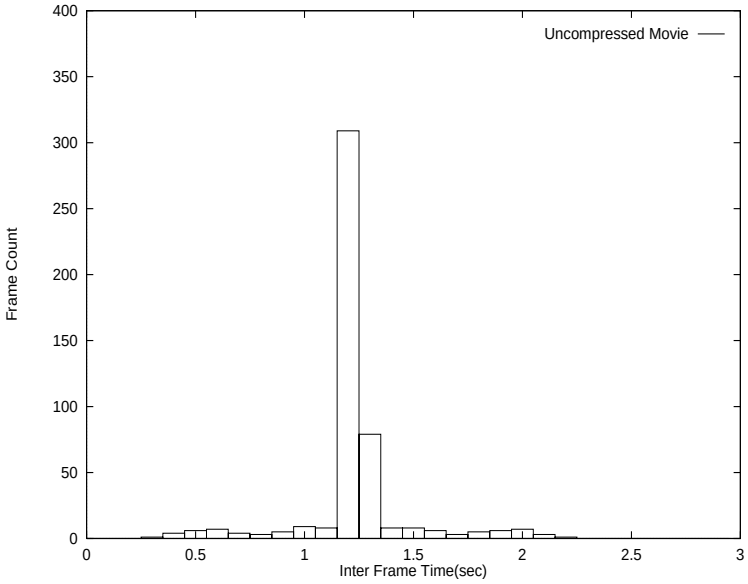

Figure 5.12: After Link 2 fails, with QoS support

Transmission scheme	QoS-VC		UDP(no QoS)	
Network load	Heavy	None	Heavy	None
Mean delay (sec)	1.224	1.224	1.474	1.193
Overhead	2.6%	2.6%	23.6%	0%
Variance	0.056	0.056	0.112	0.064

Table 5.2: Summary of delay jitters observed in two schemes

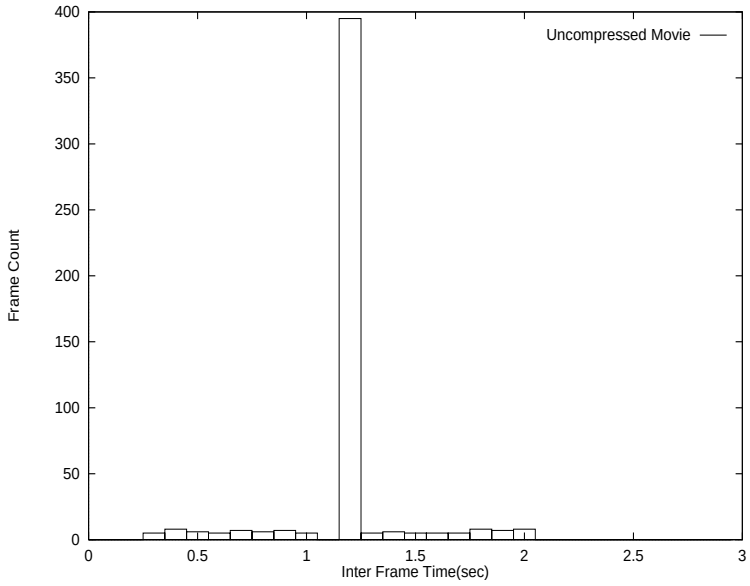

Figure 5.13: Before Link 2 fails, without QoS support

5.6 Summary

In this chapter, we propose a new concept of renegotiating QoS between network system and applications in the ad hoc wireless network. We define such renegotation scheme with an API which allows applications to specify the required QoS for a connection. Our QoS VC scheme delivers guaranteed QoS when the radio link is stable. When the quality changes, the applications gets feedback from this mechanism. Thus instead of an inadequate performance due to insufficient and varying bandwidth, the traffic source has a chance to adjust and best utilize the changing link quality without dropping the connection.

The scheme has been successfully implemented on the SWAN system, with ABR, CBR and VBR supports made available in our current prototype. Modern multimedia applications can be classified into these three categories. The results obtained from the experimental studies on QoS management in the presence of network failures show

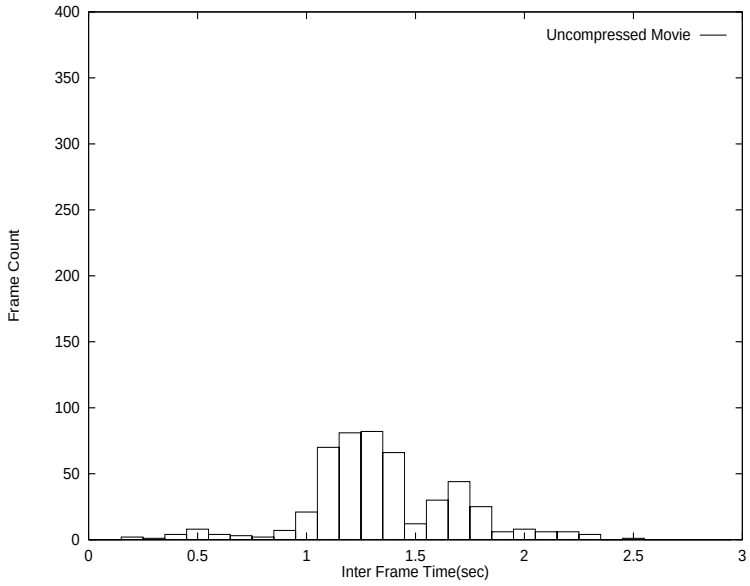

Figure 5.14: After Link 2 fails, without QoS support

that our QoS implementation on the wireless ATM network is a valid, efficient, and powerful mechanism which provides guaranteed service quality in an unpredictable, error-prone mobile environment.

CHAPTER 6

Adaptive QoS for Multimedia Applications in Wireless Networks

There is currently a wide variety of multimedia applications that deliver audio and video over a network. For instance, VIC [MJ95] and VAT [KHW96] are multimedia applications widely used in Internet. However, few of these applications have mechanisms which can take advantage of QoS information from the network, such as packet loss rate, delay jitter and available bandwidth so as to achieve a flexible operation under varying network conditions. As a result, adverse traffic conditions can cause significant degradation in the quality of a multimedia stream in environments like Internet or wireless networks, where packet loss can be large and packet delivery time cannot be guaranteed. As reported in [BV98], the perceived audio quality drops sharply as packet loss reaches 20% for non-adaptive applications even if packet retransmission techniques are used to replace lost packets.

Hence, applications operating in such environments must be able to dynamically adjust the characteristics of the multimedia stream to the changing network conditions. In this paper we propose an adaptation scheme for audio applications based on our QoS Notification Programming Model, in which senders use QoS feedback information from the receivers to dynamically select audio encoding most appropriate to the reported network conditions. The selection of audio encoding is based on the principle of media-scaling. By this principle, the bitrate (and, hence, the quality) of an

audio or a video stream is varied to be consistent with the available bandwidth.

Similar concept of application adaptation on data rate has been addressed in [Sis97]. However, this work focused on a traditional wired network, where lost packets are mostly caused by congestion. In that case, adaptation of data rate may be sufficient to improve the audio or video quality. Our work deals with wireless networks, where packet loss may also be caused by interference. The radio interference is not easily controllable through input rate regulation, therefore a different strategy must be used. In this paper, we describe and evaluate an adaptive, flexible audio application, AudioTool, specifically designed for wireless environments. We demonstrate the effectiveness of this tool using different QoS adaptation strategies. The experimental results collected while applying the AudioTool in a real wireless network will also be reported.

6.1 Overview

AudioTool is a Windows NT client/server application for delivering audio streams to clients over a network connection. It consists of the audio server application and the client playback application parts. The audio server accepts calls from the clients, accesses the requested audio record on the hard drive, and streams it to clients over the network. The client application receives the audio packets and plays them out on the audio I/O device. The playback takes place in real-time except for the initial buffering delay.

The characteristic feature of this audio-on-demand application is its ability to dynamically adapt the characteristics of the audio stream to changing network QoS. Our application follows the principles of Application Level Framing (ALF) [CT90], which

advocates closer cooperation between functions traditionally associated with network and application layers. This way, the application can take advantage of the QoS parameter information provided by the network to adapt to changes in network's behavior [Du 97].

AudioTool implements the QoS notification programming model (see Sec. 2) to provide the audio server with the ability to monitor network conditions at the client end and react adequately when congestion occurs. More specifically, the server uses 1) the knowledge of bandwidth available on the link for the audio stream, 2) packet loss rate, and 3) the amount of bandwidth required to support the audio stream, to find the optimal audio encoding parameters.

6.2 QoS Notification Programming Model

We have developed a network-independent programming model for streaming multimedia applications, which defines a general mechanism for QoS monitoring of network conditions by the receiver application and for feeding that QoS information back to the server.

The model is comprised of three main parts: 1) the network layer API, which provides the interface to network services, 2) the Network Monitor module, which collects and analyses QoS information from the network, and 3) the Application QoS Notification API, which accepts this QoS information from the Network Monitor and processes it. As Fig. 6.1 shows, such separation of the model into three modules allows the application to achieve the desired network independence: the first two components shield the Application QoS Notification API from the particulars of the underlying network technology; hence, it can stay the same even though the services provided

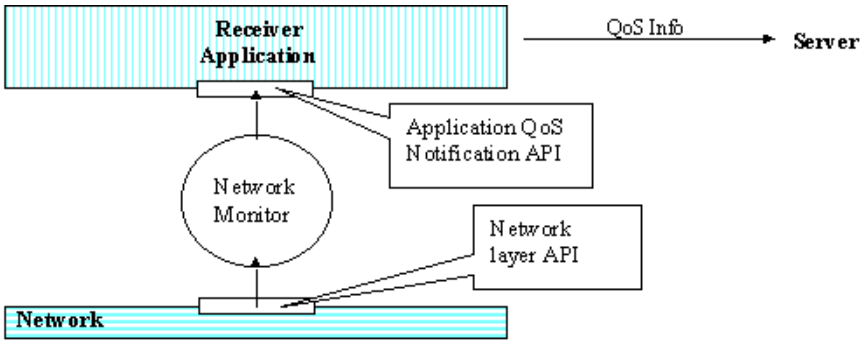

Figure 6.1: A schematic of QoS notification programming model

by the network are likely to evolve with time. We will now describe the model's components in greater detail.

6.2.1 Network Layer API

Because AudioTool is a Windows NT application, we used WinSock 2 library as our network API. WinSock functions closely resemble those of the well-known Berkeley Sockets for UNIX systems. They provide standard network operations for client/server type applications such as sending and receiving packets, requesting a connection and accepting a connection request.

In addition to the support for basic network I/O operations, WinSock 2 has a number of additional features that allow an application to utilize QoS services from the network. Specifically, it allows the application to negotiate the desired level of QoS during connection setup time or even re-negotiate the QoS contract after a connection has already been setup. These services are supported, correspondingly, by WSAConnect() and WSAIoctl() WinSock API functions, which take the extra flowspec descriptor information containing the requested QoS information.

Supplying QoS information to WSAConnect() or WSAIoctl(), however, is only meaningful when the underlying network provides QoS support. For instance, such QoS specification is essential when using WinSock 2 over an IP network with RSVP [Tec96]. Our QoS notification programming model, however, does not assume that such QoS support exists. Therefore, with a network without QoS support, the model needs to allow for measurement and reporting of network QoS parameters. The Network Monitor described in the next sections serves this purpose.

6.2.2 Network Monitor

The Network Monitor (NetMon) provides a quality of service abstraction for the application, so that it can always assume that the network provides QoS support, while in reality it may not. Its activity consists of the following three parts: 1) it monitors the multimedia stream from the server to the receiver; 2) it measures QoS parameters of this stream; and 3) it reports this QoS information to the application, so that it can act appropriately. We will now address these three activities in greater detail and describe the implementation of Network Monitor in our AudioTool application.

6.2.2.1 Monitoring the multimedia stream

The Network Monitor analyses the information contained in every packet of the multimedia stream to infer the stream QoS parameters. In our AudioTool application, the NetMon provides an API function `AcceptNewQoSInfo()`, which accepts information about audio packets coming in from the network (Fig. 6.2).

The audio server generates the sequence number and timestamp information when it sends a packet over to the receiver. It stores the two numbers in the seq_no and timestamp fields of the audio packet's transport header. When the packet arrives to the

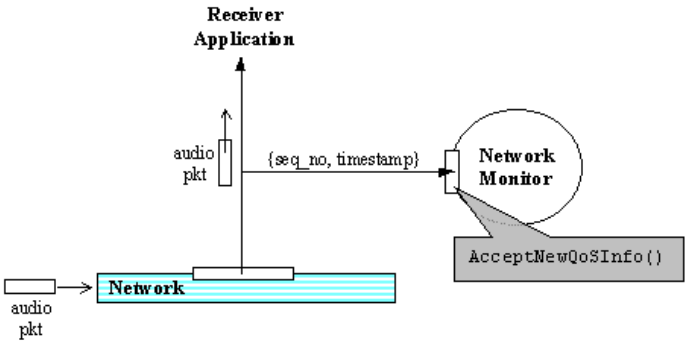

Figure 6.2: QoS Monitoring and Analysis

destination, the receiver extracts these parameters from the header and passes them to the Network Monitor.

6.2.2.2 Measuring QoS parameters

The Network Monitor uses the sequence number, the timestamp, and the local time information to determine two QoS parameters of the stream: (1) packet loss rate, lr, and (2) delay jitter, jt. The NetMon divides time into measuring periods of duration t_{mp} = 1 sec. During each measuring period, it counts the total number of packets received, n_{total}, and the number of packets lost, n_{lost}. It also records the arrival and send times of the last packet in the measuring period: $t_{LastArrival}$ and $t_{LastSend}$. The arrival time is taken to be the system time when the packet arrives to the receiver, while the send time is extracted from the packet header. At the end of every period k, the Network Monitor computes the two QoS parameters with the following simple calculations:

$$lr(k) = n_{lost}(k)/n_{total}(k)$$

$$jt(k) = [InterArrivalTime(k) - InterSendTime(k)]/InterSendTime(k)$$

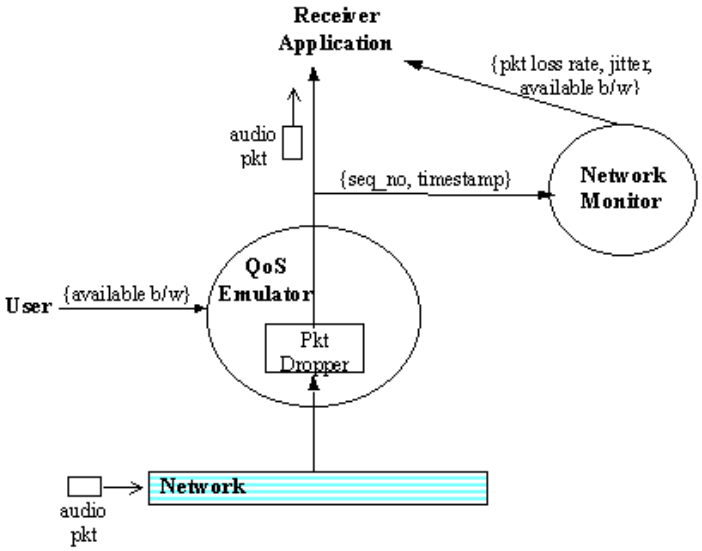

Figure 6.3: QoS Reporting

where

$$InterArrivalTime(k) = t_{LastArrival}(k) - t_{LastArrival}(k-1),$$

$$InterSendTime(k) = t_{LastSend}(k) - t_{LastSend}(k-1)$$

The two parameters are then reported to the receiver application.

6.2.2.3 Reporting QoS information to the application

As shown on Fig. 6.3, the Network Monitor reports three QoS parameters to application: the loss rate, delay jitter, and the amount of bandwidth available to the receiver. While the first two are determined by monitoring the actual packet stream, the third parameter is not computed but is provided by the user. In the future, available bandwidth can be provided by QoS routing [CTG97].

The reason for the available b/w information being emulated, rather than being

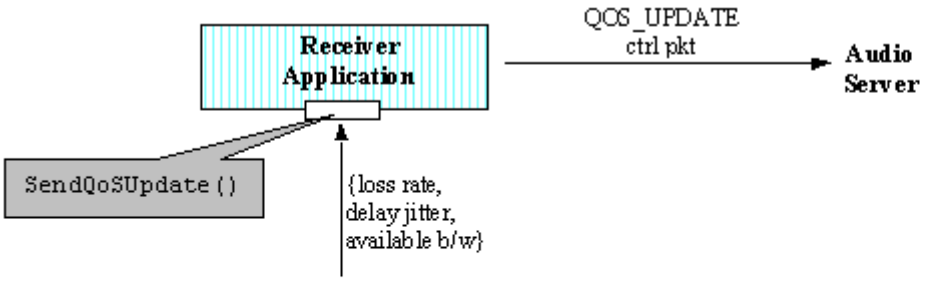

Figure 6.4: QoS Notification

measured like the rest of QoS parameters is because the network layer currently lacks the facilities necessary to provide such information. If the QoS reporting functions were incorporated into the network layer (as proposed, for instance, in [Che97]), the Network Monitor would be able to obtain b/w information directly from the OS. In this case, there would be no need for QoS Emulator and the value for the available bandwidth will be based on actual measurements. It is important to note, however, that regardless of whether some of the QoS information is emulated or not, the Network Monitor module shields the application from QoS measurement details. Therefore, the future evolution of network layer services will not affect the generality of our QoS notification application programming model.

6.2.3 QoS Notification API

The final component in our QoS Notification model is the API that the receiver application provides for QoS reports from the Network Monitor. As shown on Fig. 6.4, this API consists of a single function, SendQoSUpdate(), which accepts the new QoS parameters from the Network Monitor. The receiver application then packetizes the QoS information and transmits it in a special QOS_UPDATE control packet to the audio server, which uses it to maintain the optimal audio encoding for the connection.

The rate at which the receiver application transmits QOS_UPDATE packets equals the Network Monitor measuring rate, which is once every second. This relatively high rate is dictated by the requirements of real-time transmission of audio: if network conditions worsen at the receiver end, the audio server must react quickly to gracefully degrade the audio stream quality so that the client does not experience interruptions in the transmission.

6.3 Source Adaptation to QoS Change

By receiving QOS_UPDATE packets, an audio server is continuously aware of the network conditions at the receiver end. Upon receiving an update, it makes a decision on whether to change the current audio sampling rate or leave it intact. This decision is based upon the following heuristics:

1) If

$$lr > LR_{UpperThreshold}$$

Then

$$SamplingRate_{Current} = \texttt{OneStepDownSamplingRate}(SamplingRate_{Current})$$

$$PacketSize_{Current} = PacketSize_{Current}/2$$

2) If

$$(lr <= LR_{LowerThreshold})$$

and

$$(SamplingRate_{Current} < \texttt{BestFitSamplingRate}(AvailableBW)$$

Then

$$SamplingRate_{Current} = \texttt{BestFitSamplingRate}(AvailableBW)$$

where:

- lr is the loss rate reported by the receiver in the QOS_UPDATE message.
- AvailableBW is the bandwidth available to the receiver as reported in the QOS_UPDATE message.
- $SamplingRate_{Current}$ is the sampling rate currently in use by the audio server.
- $PacketSize_{Current}$ is the packet size currently in use by the audio server.
- `OneStepDownSamplingRate`() is a function that takes a sampling rate and returns the next lower sampling rate value. In all, there are only a few possible sampling rates that the PC audio hardware can work with: 8,000 Hz, 11,025 Hz, 22,050 Hz, and 44,100 Hz. So, for instance, `OneStepDownSamplingRate` (22.050 Hz) = 11,025 Hz.
- `BestFitSamplingRate`() is a function that takes the available b/w of a link as an argument and returns an appropriate sampling rate for PCM audio stream to be transmitted over this link with no packet loss. So, for instance, `BestFitSamplingRate`(75 kbps) = 11,025 Hz.
- $LR_{UpperThreshold}$ and $LR_{LowerThreshold}$ are constants.

A state diagram illustrating the dynamics of this heuristic function's execution is shown on Fig. 6.5.

This heuristics is based on the assumption that the primary cause of packet loss is congestion. Hence, when the audio server decreases the audio sampling rate, and therefore, its transmission rate, the packet loss should decrease and the perceived speech quality should increase.

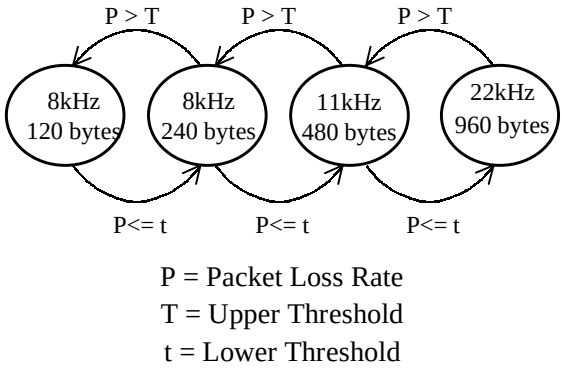

Figure 6.5: A state diagram of sampling rate and packet size adaptation mechanism.

Another, related heuristics decreases the audio packet size in order improve packet loss characteristics of the channel. When discussing experimental results in the next section we will show the effectiveness of these two approaches and of their combination.

Currently, we have defined $LR_{UpperThreshold}$ to be 10%. This number has been determined by subjectively evaluating the perceived speech quality in the presence of varying packet loss rates: we believe that 10% is about the highest tolerable packet loss rate for general-purpose audio connection, after which the perceived audio quality drops dramatically. So, if the loss rate exceeds 10%, the server will switch to a lower sampling rate in the hopes that the packet loss will decrease. $LR_{UpperThreshold}$ is defined to be zero.

In an environment subject to heavy external interference, such as the wireless network, it is very efficient to dynamically adjust packet rate as a function of interference. When interference is low, packet size should be large in order to minimize overhead. When interference is high, packet size should be reduced to minimize the probability to packet corruption.

In our experimental testbed, the WaveLAN radios drop corrupted packets, so, it is not possible for us to distinguish between loss due to congestion (which should be counteracted by reducing sampling rate) and the loss due to noise corruption (which should be corrected by shortening packet length). To overcome this problem, we assume that a given fraction (say 50 %) of the packet loss is due to interference. Thus, the server reduces the packet size by half whenever excessive lost packets are reported from the receiver, until its value reaches the minimum packet size for the WaveLAN hardware.

6.4 Experimental Results

To investigate the effectiveness of our adaptation model and the improvement in content preservation, a number of experiments were performed using the AudioTool. Two test environments were used: one that uses an emulated channel, and the other, which is a real multihop wireless testbed. For experiments with emulated channel, both server and client run on the same PC and packets are delivered through a virtual channel. This virtual channel can be manipulated to produce different levels of packet loss according to user specifications through the QoS manipulator.

The wireless testbed, on the other hand, consists of Lucent WaveLANs with multihop routing implementation [Fer97]. For experiments using real wireless network, the server runs on a stationary Windows NT workstation and the client runs on a mobile Windows 95 laptop. Speech records are transmitted by the server to the client wirelessly. Since there are several interference sources in the real world, the channel behavior can not be controlled.

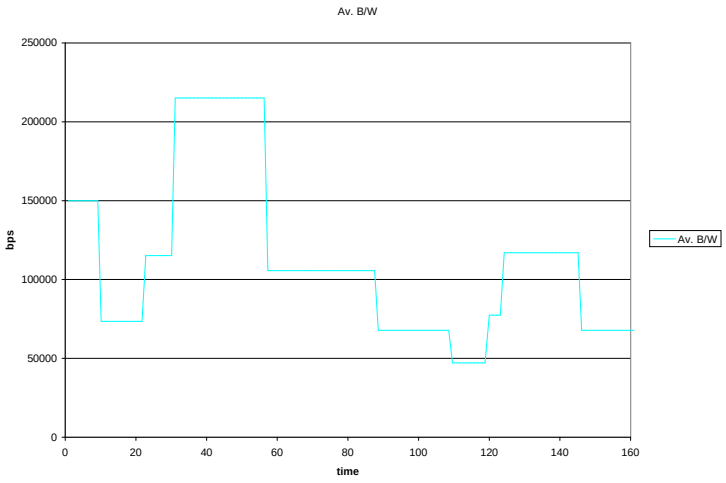

Figure 6.6: Histogram of the available bandwidth on the emulated channel.

6.4.1 Emulated Channel

The effectiveness of AudioTool is verified on the emulated channel. As shown in Fig. 6.6, we vary the available bandwidth on the channel manipulator. This causes congestion and packet drop on the emulated channel. The AudioTool reacts by reducing the sampling rate. Fig. 6.7 shows the histogram of packet loss rate and the corresponding audio sampling rates. We can see that whenever packet loss rate exceeds 10%, the server drops the sampling rate one step (until it reaches 8 kHz, which is the minimum supported sampling rate). On the other hand, when there is no packet loss, the server increases the sampling rate if there is enough available bandwidth. In this experiment, we can see how the server has managed the packet loss rate by varying the sampling rate according to the reported QoS parameters from the client.

Human perceptive evaluation was done by having a group of subjects listen to the playback generated by the client application. Overall, the audience was able to recognize most of the speech content during the playback, but it also detected the change in the audio quality whenever the server switched the sampling rate. This indicates that

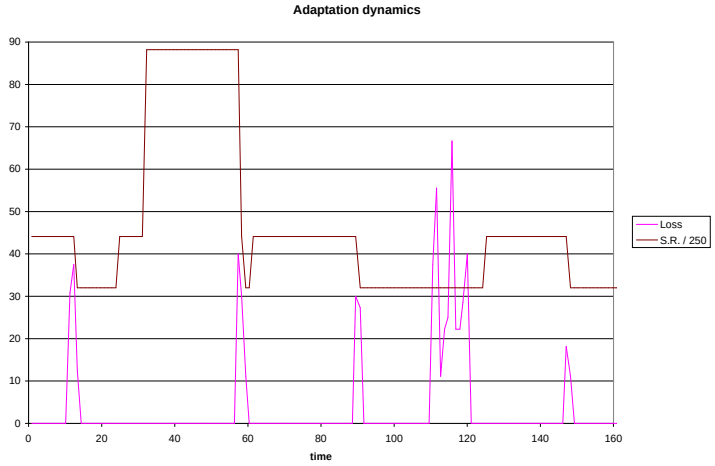

Figure 6.7: Histogram of audio transmission in the emulated channel.

most of the content are preserved with the existence of packet loss.

6.4.2 Wireless Network

The second set of experiments was performed in a real mobile wireless environment, where the packet loss resulted from actual network conditions rather than from manual adjustment of the channel emulator, as used in the previous experiment. Fig. 6.8 to Fig. 6.11 illustrate the results for following QoS adaptation policies: Fig. 6.8 shows the results when neither sampling rate nor packet size are used. Fig 6.4.2 and 6.10 show the effect when either sampling rate or packet size is used, respectively. Finally, Fig. 6.11 shows when both sampling rate and packet size are used for QoS adaptation.

From these results, we make the following observations:

1. The adaptive schemes react promptly to packet loss and jitter, leading to adjustments in sampling rate and packet size which improve performance and drive the system to the most effective operational regime.

2. The function of lost packets is typically smaller when adaptation is used. Furthermore, sampling rate adaptation permits to operate at higher sampling rates (up to 22kHz) than without adaptation, where the sampling rate is fixed at 8kHz. Higher sampling rate gives better quality, for equal packet loss rate. The heavy loss periods are much longer in the non-adaptive case than in the adaptation one (compare, for example, Fig. 6.8 and Fig. 6.11). Long audio "black outs"(several seconds) are extremely disruptive.

3. Packet size adjustment appears to be particularly effective in reducing loss rate (as shown in Fig. 6.10). This was expected since random channel interference is the major problem in our experiments. In these conditions, the shorter the packet, the lower the probability of corruptions. The long burst of lost packets in Fig. 6.10 in the interval (130-180) is due to particularly strong, persistent external interference (recall that we have no control on external interference in our experiments).

4. The adaptation of both packet size and sampling rate provides the best performance. Height and width of loss bursts are smallest in this case (see Fig. 6.11).

Interestingly, the traces also seem to show an apparent correlation between the observed jitter and packet loss at the receiver. The jitter follows the same tendencies as the packet loss, so that its increases and decreases follow those of the packet loss curve. This observation adds extra justification for our adaptation scheme. Although our scheme explicitly manages only the packet loss on the channel and does not directly attempt to decrease the jitter, it does, in fact, appear to improve both of these parameters because of the correlation that exists between the two.

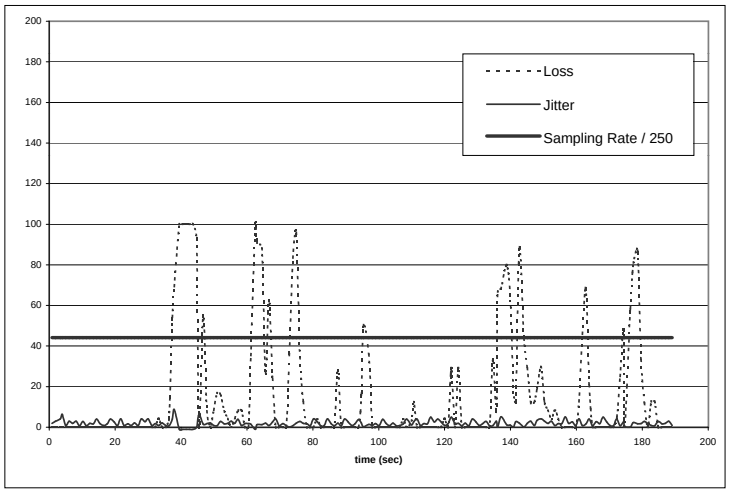

Figure 6.8: No Adaptation

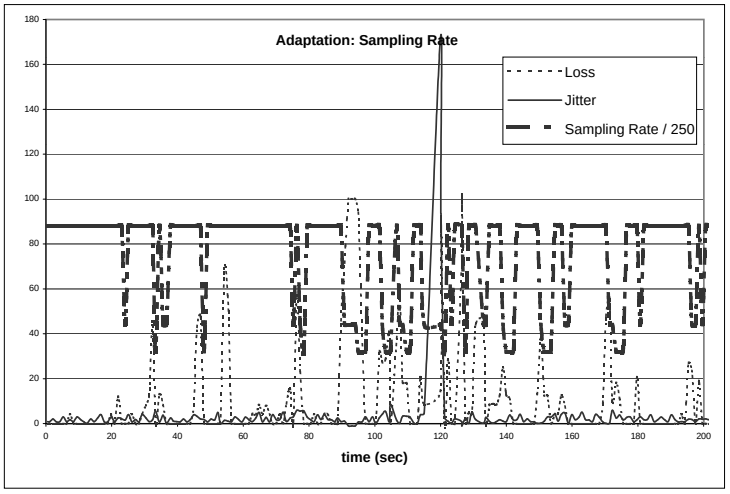

Figure 6.9: Adaptation on Sampling Rate

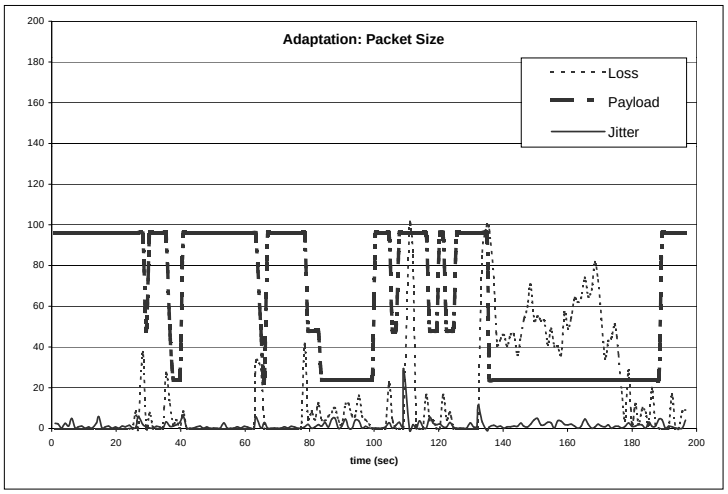

Figure 6.10: Adaptation on Packet Size

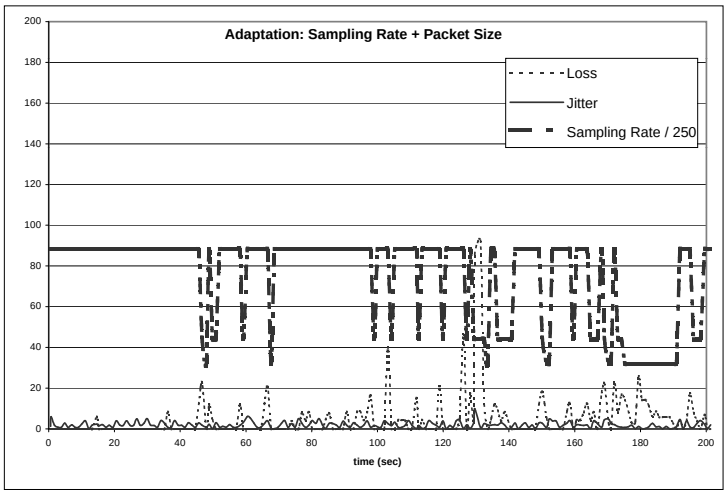

Figure 6.11: Adaptation on Both Sampling Rate and Packet Size

6.5 Summary

In this paper we presented an adaptation model for audio applications that enables them to utilize QoS information from the network in order to adapt to the changing network conditions. The adaptation mechanism is based on a media-scaling technique, in which an audio server varies the audio sampling rate and packet size to keep the bandwidth consumption in line with the amount of available network resources. Such continuous adjustment of the source encoding based on the feedback information from the client brings down the overall packet loss and results in better perceived content quality.

We have implemented the AudioTool, an audio-on-demand client/server system, based on the adaptation model. We have run a number of experiments with this application on an emulated channel and have confirmed that the use of this model does bring about audio quality improvements. We have also used this application in a real mobile multihop testbed and have discovered a correlation between the packet loss and delay jitter, which suggests that our controls will contribute to an overall lower delay jitter as well as packet loss rate.

Future work in this area involves extending the model to allow the delivery of audio information to multiple clients over either unicast or multicast networks. This extension will be compliant with RTP/RTCP [Sch96] specification, which has explicit support for multiple recipients. The adaptation mechanism can also be improved by using other methods in addition to media-scaling, such as varying audio codec to trade off bandwidth consumption for audio quality. For instance, an audio server may switch to CELP encoding (4.8 kbps) from its usual PCM encoding method (64, 88.2, 176.4 kbps), if it finds that lowering the sampling rate will still not bring down the bitrate of the stream to the appropriate levels.

Finally, the model could be improved by giving clients a greater role in the adaptation scheme. One possibility is for a client to vary the amount of buffering that is performed before audio playback. The client could trade-off a longer buffering time and higher memory consumption for an improvement in audio quality that would come from the reduction of playback jitter.

CHAPTER 7

Conclusion

The ad hoc wireless network presents different challenges in the protocol design in all layers. The goal of our research is to provide multimedia services in ad hoc wireless network, and we focus our work on the design of an efficient network routing scheme and the QoS adaptation in the application and the operating system layers. In this chapter, we conclude our work proposed in this dissertation.

7.1 Contributions

- **Global State Routing** Based on the current research trend of ad hoc wireless network, it has been shown that link state type of routing schemes will have more potential in the future, as long as the control overhead can be kept reasonably low. Our GSR, which disseminates the topology information without flooding small packets and reduces the control overhead by the fresh update and the fisheye technique, presents itself as a best candidate for the routing protocol in wireless networks.

- **QoS Routing in Wireless Network** QoS routing is the key to multimedia support. The goal is to find n feasible path that provides better end to end QoS. Our work in QoS routing includes extending the wireless routing protocols, DBF and GSR, with QoS parameters to help with admission control. Even though

these QoS parameters are still preliminary, improvement of network effective throughput is observed. With GSR providing a more accurate global network information, the achievable throughput is even more than it is in DBF.

- **Renegotiable QoS** Even with a perfect QoS support in the network facilities, there are still some uncontrollable issues that make constant QoS guarantee impossible. As a result, multimedia service may not satisfy the user need if the application can not adapt to the changing condition. Such adaptation requires cooperation between the user application and the network systems. Our work with renegotiable QoS support in the network system, together with the QoS adaptation in application layer, provide a better information preservation and a better tolerance to the network failures, which usually result in a decrease in quality of service.

7.2 Future Work

Future research has several directions. First of all, hierarchical routing with GSR exhibits a great potential in further reducing the control overhead and local storage. But the challenge will be the complexity of maintaining the hierarchy information, and the inaccuracy due to the abstraction of network information. Secondly, the optimal QoS routing is not achievable in polynomial time, but a simple heuristic algorithm may still provide valuable improvement if proper QoS parameters are chosen. We believe that in a wireless network, mobility should also be considered in making routing decisions. Finally, all the research results are meaningful only if these results can be carried out in the real world. Therefore, the ultimate goal for our research is to realize a wireless environment that provides multimedia service to mobile users. Currently, a prototype

that realizes some of our current results has been implemented at UCLA; the next step will be to further improve what we have now, and make it a more robust system.

REFERENCES

[Agr96] P. Agrawal and et al. "SWAN: A Mobile Multimedia Wireless Network." In *IEEE Personal Communications*, pp. 18–33, April 1996.

[BG87] D. Bertsekas and R. Gallager. *Routing in Data Networks*, chapter 5. Prentice Hall, second edition, 1987.

[BV98] J.-C. Bolot and A. Vega-Garcia. "The Case for FEC-Based Error Control for Packet Audio in the Internet." In *ACM Multimedia Systems*, 1998.

[CE95] M. S. Corson and A. Ephremides. "A destributed routing algorithm for mobile wireless networks." *ACM-Baltzer Journal of Wireless Networks*, **1**:61–81, January 1995.

[CG] C.-C. Chian and M. Gerla. "Routing in Multihop, Wireless Network." submitted for publication.

[Che97] T.-W. Chen and et al. "A New Scheme for Fault Tolerance in Wireless Networks." In *The 27th Annual International Symposium on Fault-Tolerant Computing*, 1997.

[CT90] D. Clark and D. Tennenhouse. "Architecural considerations for a new generation of protocols." *Computer Communication Review*, **20**(4), September 1990.

[CTG97] T.-W. Chen, J.T. Tsai, and M. Gerla. "QoS routing performance in a multihop, wirelss network." In *IEEE 6th ICUPC*, October 1997.

[Du 97] J. Du and et al. "An Extensible Framwork for RTP-based Multimedia Applications." In *Network and Operating System support for Digial Audio and Video*, pp. 53–60, May 1997.

[Fer97] R.M. Fernie. "Implementation of various routing algorithms for TCP/IP wireless networking." Technical report, UCLA, Computer Science Department, June 1997.

[Gar89a] J.J. Garcia-Luna-Aceves. "A Minimum-hop Routing Algorithm Based on distributed Information." In *Computer Networks and ISDN systems*, volume 16, pp. 367–82, 1989.

[Gar89b] J.J. Garcia-Luna-Aceves. "A unified Approach to Loop-Free Routing Using Dsitance Vectors or Link States." In *ACM SIGCOM*, pp. 212–23, 1989.

[Ger86] M. Gerla. “Bandwidth routing in integrated service networks.” In *ICCC*, 1986.

[Goo90] D.J. Goodman. “Cellular Packet Communications.” *IEEE Transactions on Communications*, **38**, August 1990.

[GT95] M. Gerla and J. T. Tsai. “Multicluster, mobile, multimedia radio network.” *ACM-Baltzer Journal of Wireless Networks*, **1**(3):255–65, 1995.

[Has96] Z. J. Hass. “A new routing protocol for the reconfigurable wireless networks.” In *IEEE 6th ICUPC*, October 1996.

[Hed88] C. Hedrick. “Routing Information Protocol.” In *IETF RFC 1058*, 1988.

[Iwa96] A. Iwata and et al. “ATM routing algorithms with multiple QOS Requirements for Multimedia Internetworking.” In *IEICE Transactions on Communications*, volume E79-B(8), pp. 999–1007, August 1996.

[Jaf86] J.M. Jaffe and et al. “Subtle Design Issues in the implementation of Distributed, Dynamic Routing Algoriths.” In *Computer Networks and ISDN systems*, volume 12, pp. 147–58, 1986.

[JT87] J. Jubin and J. D. Tornow. “The DARPA Packet Radio Network Protocols.” *Proceedings of IEEE*, **75**(1), January 1987.

[KHW96] I. Kouvelas, V. Hardman, and A. Watson. “Lip Synchronization for use over the internet: Analysis and implementation.” In *IEEE Globecom*, 1996.

[KS71] L. Kleinrock and K. Stevens. “”Fisheye: A Lenslike Computer Display Transformation”.” Technical report, UCLA, Computer Science Department, 1971.

[MG95] S. Murthy and J.J. Garcia-Luna-Aceves. “A Routing Protocol for Packet Radio Networks.” In *Proc. IEEE Mobicom*, pp. 86–95, November 1995.

[MJ95] S. McCanne and V. Jacobson. “`vic`: A flexible framework for packet video.” In *Proc. of ACM Multimedia*, 1995.

[Moy94] J. Moy. “OSPF Version 2.” In *IETF RFC 1583*, 1994.

[PB94] C.E. Perkins and P. Bhagwat. “Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers.” In *ACM SIGCOMM’94*, pp. 234–44, 1994.

[PC97] V. D. Park and M. S. Corson. “A Highly Adaptive destributed routing algorithm for mobile wireless networks.” *IEEE Infocom*, 1997.

[PH95] J. Porter and A. Hopper. “An overview of the ORL Wireless ATM System.” In *IEEE ATM Workshop*, Washington, DC, September 1995.

[RW94] D. Raychaudhuri and N. D. Wilson. “ATM-based Transport Architecture for Multiservices Wireless Personal Communication Networks.” *IEEE JSAC*, **12**(8):1401–14, October 1994.

[Sch96] H. Schulzrinne and et al. “RTP: A Transport Protocol for Real-Time Applications.” In *RFC 1889*, January 1996.

[Sed83] R. Sedgewick. *Weighted Graphs*, chapter 31. Addision-Wesley, 1983.

[Sis97] D. Sisalem. “End-to-End Quality of Service Control using Adaptive Applications.” In *IFIP Fifth International Workshop on Quality of Service*, 1997.

[SM96] C.J. Sreenan and P.P. Mishra. *Equus: A QoS Manager for Distributed Applications*, pp. 496,509. Publishers Chapman & Hall, 1996.

[Tan96] Andrew S. Tanenbaum. *Computer Networks, Third Edition.* Prentice Hall, 1996.

[TC94] J. Trotter and M. Cravatts. “A Wireless Adapter Architecture for Mobile Computing.” In *Proc. 2nd USENIX Symp. on Mobile and Location-Independent Comp.*, pp. 25–31, April 1994.

[Tec96] StarDust Technologies. *Windows Sockets 2 Protocol-Specific Annex, Rev. 2.0.3*. May 1996.

[WC95] Z. Wang and J. Crowcroft. “QoS Routing for Supporting Resource Reservation.” In *University College London White Paper*, 1995.