

Dynamic QoS Resource Allocation in Bluetooth Piconet

by

Gaurav Tuli

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

February 6, 2001

[February 2001]

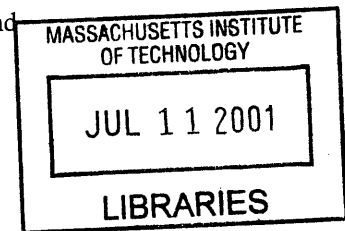
Copyright 2001 Gaurav Tuli. All rights reserved.

BARKER

The author hereby grants to M.I.T. permission to reproduce and

distribute publicly paper and electronic copies of this thesis

and to grant others the right to do so.



Author _____

Department of Electrical Engineering and Computer Science

February 6, 2001

Certified by _____

Gopal Krishnan

VI-A Company Thesis Supervisor

Certified by _____

Professor Kai-Yeung (Sunny) Siu

M.I.T. Thesis Supervisor

Accepted by _____

Arthur C. Smith

Chairman, Department Committee on Graduate Theses

Dynamic QoS Resource Allocation in Bluetooth Piconet

by

Gaurav Tuli

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

February 6, 2001

ABSTRACT

The purpose of this thesis was to address the issue of resource allocation in the bandwidth-constrained environment. We focus on the provisioning of resources to adaptive multimedia applications that can change coding schemes based on available resources. We explore the issues behind designing a general call-level QoS system that reserves paid-for resources for these applications. Our primary contribution is in the algorithms for network admission control. Here we introduce a dynamic resource negotiation scheme that not only allows for adaptive flows during traffic execution, but also for QoS re-negotiations with existing flows at the point of admission decision for a new flow. Results show noteworthy increases in call acceptance rates and average number of users receiving their requested QoS.

VI-A Company Thesis Supervisor: Gopal Krishnan

Title: Chief Architect, Motorola, Semiconductor Products Sector

M.I.T. Thesis Supervisor: Kai-Yeung (Sunny) Siu

Title: Associate Professor, MIT Department of Mechanical Engineering

Acknowledgments

This work would not have been possible if it were not for Gopal Krishnan, my supervisor at Motorola, who never ceased to amaze me with his ideas, insight and dedication. Thank you to Professor Siu for providing the essential academic perspective and reviewing our drafts. I owe my sincerest thanks to my family for their endless love and support, and instilling into me their wonderful appreciation of education. And finally, thank you to those people who have remained so close to me, you know who you are, and without you I would never have reached where I am today.

Table of Contents

- 1 Introduction
 - 1.1 Motivation and Introduction
 - 1.2 Main Contributions
 - 1.3 Thesis Structure
- 2 Background
 - 2.1 Quality of Service (QoS)
 - 2.2 Mobility
 - 2.3 Bluetooth
 - 2.4 Available Bit Rate Applications
 - 2.5 Dynamic QoS Management
- 3 Problem
 - 3.1 Problem Statement
 - 3.2 Solution Overview
- 4 Design
 - 4.1 QoS Framework
 - 4.2 QoS Parameter Selection
 - 4.3 Utility Model
 - 4.4 Literature Survey of Dynamic Schedulers
 - 4.5 Dynamic Admission Control Algorithm
 - 4.6 Interval Maintenance System Overview

- 5 Simulation and Results
 - 5.1 Simulation Environment
 - 5.2 Results
 - 5.3 Analysis
- 6 Conclusion

Bibliography

Appendix A: Simulation code

CHAPTER 1

INTRODUCTION

1.1 Motivation and Introduction

Short-range wireless connectivity is seeing an emerging demand in consumer and enterprise markets as handheld devices are growing in usage and popularity. Both business and personal users are feeling the increasing need to have effortless, instantaneous connections to local area networks wherever they are. Additionally, they are demanding ad hoc (instantaneous) connections between their personal devices to exchange information with other users and/or to synchronize data between their own electronic devices. Enterprise users are realizing the benefits of replacing the traditional tethered Ethernet in their offices with wireless connections. According to market research firm Frost & Sullivan, this wireless local area network (WLAN) market in the United States will see a 20.7% compound annual growth rate from 1996-2003 and Yankee Group projects \$1.3 billion in revenue in the WLAN space by 2002 [YAN00].

Many business trends are leading the way for the need for wireless connectivity. First and foremost, there has been significant growth in mobile infrastructure for communications. Companies depend even more on real-time information for their employees. Users' productivity increases dramatically when they can communicate easily with coworkers. Additionally, employees are working more frequently from many

different locations: home, office, and on the road. Virtual private networks and public access stations are becoming more vital.

For many small devices it will be essential to offer wireless connectivity because without it, the benefits of mobility are lost. Products such as cell phones and personal digital assistants (PDAs) become increasingly useful if synchronization amongst each other and with laptops/desktops can occur wirelessly and instantaneously. Beyond just these devices, wireless solutions offer the convenience of reaching networked appliances that cannot be tethered easily. Companies are designing network-enabled appliances such as washing machines and microwaves for which a wireless network connection would be most applicable [YAN00]. An entire home can be networked without the clutter and labor needed for wiring Ethernet.

Wireless connectivity is an alternative to traditional tethered Ethernet and new technologies such as HomePNA (a home-networking kit that allows PC's to network through traditional phone jacks). For homes with fewer phone jacks or regulations surrounding their use (such as in Europe), wireless networking is the obvious solution. In offices, the convenience of moving a laptop from one local area network to a conference room to another local area network while retaining a video conference call is invaluable. Corporate users can bring their work home more easily by having their office laptop instantly attach itself to their home network when they arrive home. These future personal area networks (PANs), where PDAs, laptops and cell phones are all communicating wireless, are expected to transport a broad spectrum of traffic including audio, video, pictures and data.

As a response to this growing demand, many proposed standards have emerged to offer this kind of connectivity to consumers. These include, among others, Bluetooth and IEEE 802.11b and HomeRF. The Bluetooth protocol aims to be the lowest cost and most robust solution to short-range wireless connectivity. Bluetooth was conceptualized by Ericsson and developed by the Bluetooth Special Interest Group as a cable replacement technology primarily for low-cost mobile devices. The first release of the protocol came in 1999 with very low power consumption and data rates of up to 1 Mbps. The range varies from 10 meters for low power to 100 meters for the full power device. For Bluetooth enabled devices, the key feature is its ability to instantaneously connect devices when they are within transmitting distance of each other. A user's PDA will automatically synchronize itself with his laptop when he brings them close together. Two Bluetooth laptops can also instantly network with each other to allow for file exchange when they are near one another. The technology supports up to 8 nodes per piconet and uses a Frequency Hopping Spread Spectrum scheme in the 2.4 GHz range. The emergence of the scatternet concept will allow a greater number of devices in a Bluetooth network. A noticeable feature for voice/data convergence is that Bluetooth can support up to 3 simultaneous voice channels while still running 5 other data flows [BLU99].

One of the major problems facing this technology however, is the fair handling of heterogeneous traffic to guarantee a specified level of quality of service (QoS). The proposed Bluetooth standard attempts to address this issue, however we believe many improvements can be made in the manner Bluetooth handles real-time traffic such as voice telephony and streaming video.

Support for such multimedia applications over the Internet is presently in very early stages. Real-time traffic can run seamlessly in corporate environments where bandwidths range from 1 Mbps to 100 Mbps, however an environment where bandwidth is constrained and inconsistent poses a large barrier for these applications. In public access networks for example, the guaranteeing or reservation of resources in a highly contentious, low resource environment is a very difficult problem. The Internet Engineering Task Force (IETF) has been developing recommendations towards issues related to security, pricing and guaranteeing of services in the Internet. Because of the best-effort nature of packet transmission in the Internet, IETF has devised protocols to expand the ability of public access networks to handle real-time traffic. These include the Resource Reservation Protocol (RSVP), Internet Protocol version 6 (IPv6), and Real-Time Transport Protocol (RTP).

QoS in Bluetooth faces similar issues as public access networks. It was designed as a low-bandwidth, low-power consumption protocol ideal for mobile environments. Thus, available resources are changing rapidly as users move in and out of a Bluetooth network, making maximum resource utilization a difficult task.

1.2 Main Contributions

The primary focus of this research is to address the issue of resource allocation in the bandwidth-constrained, mobile, Bluetooth environment. For our study, resource allocation consists of three distinct functions: admission, scheduling and maintenance. We focus on the provision of resources to adaptive multimedia applications that can change coding schemes based on available resources. We explore the issues behind

designing a general call-level QoS system that reserves paid-for resources for these applications. We hope the research is applicable in most mobile environments where available resource conditions are changing frequently. Our key contribution and focus is in the algorithms for Bluetooth network admission. Here we introduce a dynamic resource negotiation scheme that not only allows for adaptive flows during traffic execution, but also for QoS renegotiations with existing flows at the point of admission decision for a new flow.

More specifically, our main contributions include:

- **Recommendations on a QoS signaling framework suitable in the mobile environment**
- **A selection of essential QoS parameters with negotiable parameter windows**
- **A versatile model of resource utilization in a paid-for-resource environment**
- **A survey of dynamic scheduling routines**
- **Algorithms for optimal bandwidth provisioning at admission control**
- **Stochastic simulation of network at point-of-admission decision**
- **Recommendations for QoS interval maintenance routines**

1.3 Thesis Structure

The remainder of this thesis is arranged as follows:

- Chapter 2 provides background information on many of the important issues related to our research including quality of service, mobility, Bluetooth

technology, adaptive rate multimedia applications and dynamic QoS management.

- Chapter 3 presents our problem statement, proposed approaches and the solution overview.
- Chapter 4 describes our framework including models for QoS parameters and utility. We survey dynamic schedulers and detail and justify our admission algorithms. Additionally we discuss options for interval maintenance routines.
- Chapter 5 discusses the simulation environment and results of our research in admission algorithms.
- Chapter 6 offers concluding remarks and an overview of directions for future work in this area.

CHAPTER 2

BACKGROUND

2.1 Quality of Service

2.1.1 QoS Overview

Future personal area networks (PAN) are expected to reliably transport a broad spectrum of traffic including audio, video, pictures and data. Each of these traffic types places a different set of requirements on the network carrying it. Digital audio and video streams found in modern multimedia applications for example, demand a level of service quality that is not necessarily attainable by traditional "best effort" networks. The nature of this type of data requires that networks provide for low delay, jitter and packet loss so that a continuous and timely data stream is received at the destination. Multimedia data can often tolerate a certain level of loss or corruption, but its delay requirements are often stringent so that the stream remains uninterrupted. On the other hand, data traffic (such as FTP transfer), which is more latency tolerant, requires a strict level of data reliability.

The varying requirements of different data types leads to the need for a measure of *service quality* so that each type of traffic can define its requirements. The traditional measures of *service quality* are delay, jitter, bandwidth and reliability [FH98]. *Delay* is the length of time elapsed from transmission to arrival. For interactive network applications, long delay is undesirable. *Jitter* is the variation in delay experienced by the receiver. High levels of jitter result in a discontinuous stream, which is inadequate for

audio and video. *Bandwidth* is the maximum transfer rate associated with a transmission channel. Multiple flows may occupy the channel and share a given bandwidth. Some applications require low bandwidth, such as the transmission of control information, whereas others demand high bandwidth, such as video streams. Finally *reliability* is a measure of a network's error rate. This includes losing packets due to congestion or retransmitting due to corruption. An unreliable voice stream may sound broken and "crackly" at the receiver.

These measures of service quality can be used to broadly categorize traffic into two classes summarized in the table below:

	Class I	Class II
Name	Real-time	Non-real-time
Examples	Voice & video	Data services
Delay	Bounded	Unbounded
Jitter	Bounded	Unbounded
Bandwidth	Guaranteed	Not guaranteed
Reliability	Loss-tolerant	Zero-loss
Time	Probably defined	Not defined
Time Slots	Defined	N/A

Table 2.1: Traffic Classes

Having defined *measures* of service quality we can now use the following definition of quality of service (QoS):

Quality of service is a set of the measures of service quality a given application requires for proper operation.

A network that provides QoS differentiation for applications must also be able to reliably deliver this QoS consistently and predictably [FH98]. There are 3 primary approaches in guaranteeing a given set of QoS parameters. The first method treats all traffic equally by simply assigning a maximum packet delay that can be guaranteed. The other two methods rely on the fact that the data being transmitted is either class I (time-bounded) or class II (not time-bounded). Method 2 gives highest priority to class I services and allows them to run to completion before class II services can begin transmitting. Method 3 allows both classes to transmit simultaneously, however if class I services are not meeting their QoS guarantees because class II is also transmitting, then class II will be temporarily delayed towards transmission of data within the latency tolerant interval. Method 3 obviously has the best bandwidth efficiency of the three in heterogeneous traffic but may require a complex algorithm to allow for arbitration between Class I and Class II packets [SET98].

2.1.2 QoS Survey

Because most of our work involves the provision of QoS in some manner, we provide here a brief survey of existing QoS systems. We divide the mechanisms of delivering QoS by the network stacks/layers.

2.1.2.1 Physical layer

We view the physical layer as the lowest level of hard-wired paths within a network where the primary concern is the delivery of bits (in whatever form they are received from higher layers) into a channel. Thus at this layer, provisioning of QoS can only be done through providing multiple paths (multiple frequencies / tones for Digital Subscriber Lines and multiple time slots for Bluetooth) to destination. In this form, the QoS provided is termed differentiated services. Traditionally, the creation of multiple paths has been for backup purposes, as would be needed in the case that a primary link fails, but with TDM based services the multiple paths are increasingly aimed at QoS convergence. However, if the two paths offer different physical characteristics that result in distinct bandwidth and delay properties, higher layers protocols can transmit data through either path depending on the QoS requirements of the data.

Problems with this method are two-fold. First, higher layers implement signaling systems so that the receiver can determine traffic parameters. Unless intelligent signaling is implemented, the feedback mechanism between the sender and receiver will result in the lowest quality of service link in multiple paths to shape the flow. This removes the effectiveness of having multiple paths at the physical layer. Secondly, asymmetric send/receive paths could result in ineffective signaling where the receive path acknowledgments may give inaccurate measures of link latency.

2.1.2.2 Link Layer

The link layer frames data for point-to-point, error-free transport through the physical layer. QoS is implemented in this layer in Asynchronous Transfer Mode (ATM) networks and proposed by IEEE 802.1p for Ethernet LANs. ATM provides a very broad but complex virtual circuit system to provide QoS in its networks. It differentiates traffic into 4 requirement sets and guarantees service in some form for each of them. Constant Bit Rate (CBR) applications require a constant level of bandwidth and maximum delay bound. These include voice and some video applications and are given virtual circuit treatment by ATM. Real-time and non-real-time variable bit rate traffic (rt-VBR and nrt-VBR) are treated as applications that transmit at variable rates through the lifetime of their connection. Rt-VBR traffic is categorized as multimedia streams that can tolerate some cell loss or impaired cells. Because of the delay requirements of the flows, when cells rt-VBR traffic become excessively delayed, they become of little or no value to the receiver. Nrt-VBR traffic is targeted for transaction-based applications where traffic is expected in bursts. In such traffic, a bandwidth guarantee is required for the applications to run effectively, though delay is not a necessary component. Available bit rate (ABR) applications dynamically modify their coding scheme to adjust to the available resources in the network. ATM provides feedback to the originators of this traffic class so that they can fully utilize the network for their service. Finally unspecified bit rate (UBR) applications are essentially best effort. Flow control and time synchronization between the source and destination do not occur. This type of handling is appropriate for data transfers, such as with FTP.

The primary problem associated with ATM QoS handling is its complexity and as a result, network administrators have been hesitant to use its diverse features. Added to

this problem however, is that ATM is not widespread enough to have these QoS implementations be useful. For true end-to-end QoS in an ATM system, the entire connection must have ATM at the link layer. This is a rare case however. More often than not, ATM only provides the link layer at a small segment of the larger traffic path. As a result, the end-to-end characteristics are what shape the flow and the effectiveness of the QoS implemented in ATM is significantly reduced. Additionally, higher layer protocols that manage flow and congestion control in their own way will force ATM to receive inaccurate information on its own portion of the flow.

IEEE 802.1p provides mechanisms for prioritized traffic in an Ethernet or token ring environment. At the link layer, it defines a user priority field that offers up to 8 different priority levels. Specialized queuing systems ideally would be able to map this priority field into a relative queuing order in LAN switches and routers.

2.1.2.3 Network Layer

At the network layer in the global Internet, we look to TCP to provide any mechanisms for quality of service. For reliable, adaptive rate transmission (this does not necessarily imply adaptive-rate applications as in ABR traffic), TCP provides several end-to-end procedures for congestion avoidance. In an attempt to find a stable point where the sender and receiver are performing at optimal and equal rates, TCP uses two mechanisms. First TCP slow start incrementally injects traffic at a higher rate (by increasing TCP window sizes) until congestion in the network is seen. When this occurs, the second mechanism, congestion avoidance, significantly (often halving) reduces the window size and lets slow start take over again. This system of slow start and congestion

avoidance works well in most situations. However, where many flows are simultaneously existing, an unstable state can be reached when all are hitting congestion at the same time on a consistent basis.

Other network layer mechanisms are:

- **Resource Reservation Protocol (RSVP) provides a signaling mechanism for notification of necessary allocations**
- **Prioritized packet discarding in queues based on Internet Protocol's (IP) Type of Service parameter**
- **Scheduling algorithms that provide preferential treatment towards differentiated services (e.g. priority queuing, weighted fair queuing)**

2.2 Mobility

The introduction of mobility into personal area networks introduces problems and complexities not found in traditional stationary networks. A mobile ad hoc network is characterized as a distributed system of independent mobile nodes connected through a wireless medium. These nodes act as both hosts and routers in an ever-changing network topology. Mobile ad hoc networks are often relatively low-bandwidth (as compared to stationary LANs) and are dynamically interference prone. Because mobility allows freedom to move over short distances, network conditions can change frequently and drastically. Users in a mobile environment can move in and out of the ad hoc network as they wish, and thus available resources within the network are changing unpredictably. Thus, the issue of predictable response to this changing environment is an essential

problem to QoS in a mobile system. Such networks require highly adaptive architectures to maintain a reasonable level of network utilization.

The impact of mobility on QoS has two key components:

- **Link type**

Link properties in mobile systems are significantly different than traditional cabled networks because of low bandwidth and higher unreliability. Modern wireless systems are often an order of magnitude less in bandwidth than fixed networks. Therefore, multimedia applications which can have bandwidth requirements on the order of Mbps may run seamlessly in a wired Ethernet, but will face considerable obstacles in the wireless environment. Therefore, in providing QoS, mobile systems must try to dynamically utilize as much of the available resources as possible.

- **Movement**

As devices move from one mobile network to another, a handoff occurs so that the flow may continue uninterrupted. This puts stringent requirements on the QoS management system because it must either predict mobility or face frequent connection blocking. Because movement causes variation in the available resources of a particular link, QoS systems must provide operational ranges rather than hard allocations. Movement also adds the requirement that the mobile QoS architecture must accommodate the frequent entrances and exits of devices. However the prediction of mobility is beyond the scope of this thesis.

2.3 Bluetooth

2.3.1 Overview

The Bluetooth protocol is an open standard for mobile wireless connectivity between PCs and communication devices. It is an industry-backed development targeted for voice and data transfer amongst devices over short-range (approximately 10m). It operates in the unlicensed 2.4 GHz Industrial-Scientific-Medical (ISM) band where it performs coordinated frequency hopping. Initially created as a cable-replacement technology, its applications were meant to include simple ad hoc networking between PCs and peripheral devices (such as PDAs and cell phones). Because of low-cost and low-power consumption projections, the 9mm by 9mm Bluetooth chip's applications have expanded to also include home networking and advanced network-enabled wireless appliances. Additionally, the protocol benefits by connectivity that does not require line-of-sight and works well through most walls and objects.

The Bluetooth Special Interest Group (SIG), the certification and standardization body for the protocol, consists of 2000 member companies and was founded by Ericsson, IBM, Intel, Nokia and Toshiba. Additionally, primary supporters include Motorola, Microsoft and Lucent. By using the resources and expertise of each of these companies, the technology has grown to be a robust and rapidly growing protocol. In its present form, it supports 780 Kbps in asymmetric mode (721 Kbps down-link, 57.6 Kbps up-link) and 432.6 Kbps in symmetric form. Additionally, it can support one asynchronous data channel with 3 synchronous voice channels simultaneously or a single voice/data

channel. With these facilities, applications such as hands free headsets and automatic synchronization with personal devices are intended to work seamlessly.

2.3.2 Network Profile

The Bluetooth ad hoc and stationary networking scheme allows for grouping of clients into piconets and the extension of piconets into Scatternets. Devices can exist standalone or as members of a piconet as they move within and out of range. Piconets grow in an ad hoc manner from 2 members, to include up to one master and 7 slaves. Although all members have equivalent device specifications for Bluetooth, a master is designated to synchronize the clock and frequency hopping pattern amongst piconet members. Additionally, a client can be a slave in 2 piconets or the master in one and a slave in another. This allows for scatternets, or several interconnected piconets (see Figure 2.1). Each non-parked member of a piconet has a 3-bit active member address to distinguish it from other clients. Sniff and hold modes are also low-power modes (similar to park), however members retain their addresses and still participate in the network periodically. Clients enter a piconet from standby mode when they are in range of another Bluetooth device by either sending or receiving an inquiry or page instruction.

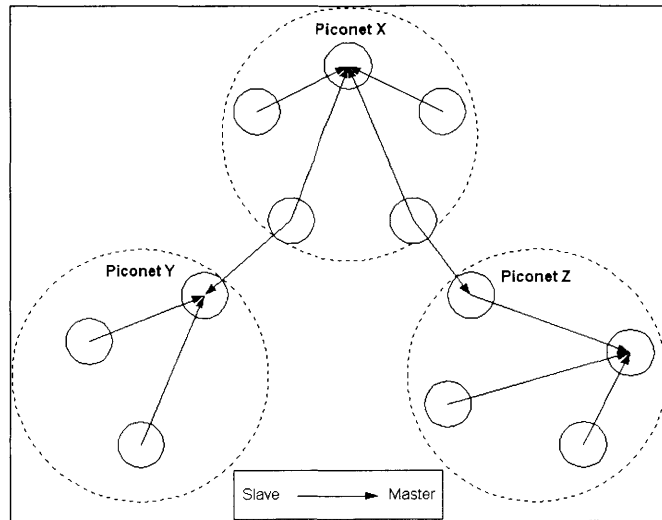


Figure 2.1: Bluetooth Scatternet

2.3.3 Architecture

The Bluetooth architecture, for our purposes, can be generally categorized into 5 roles. The RF unit operates the antenna and Bluetooth frequency hopping scheme. The Link Controller (LC), or baseband, manages the low-level link responsibilities and provisioning of data and voice channels for higher layers. The Link Manager (LM) directs the management and creation of baseband links. The Logical Link Control and Adaptation Layer (L2CAP) is responsible for providing connection oriented and connection-less links to higher layers. Finally, above L2CAP are additional interface protocols to the application layer.

2.3.3.1 RF

The RF control hardware operates the Bluetooth radio at 2.4 GHz using a 0 dBm antenna which results in very low power consumption. It uses a frequency hopping spread spectrum scheme to hop between 2.402 GHz and 2.480 GHz. Using hops of 1

MHz, it can achieve hopping rates up to 1600 hops/sec. This results in a 625 μ s time-slot. Bluetooth has a maximum symbol rate of 1 Mb/s.

2.3.3.2 Link Controller

The Link Controller is the baseband operations of Bluetooth devices. Bluetooth LC provides physical layer link specifications for two link types. Both use Time Division Duplex (TDD) schemes and support 16 different packet types. The packet types provide a variety of services including various levels of error checking, relative packet prioritization and multiple time-slot packets (1, 3 or 5 time-slot lengths). The link types may change at any point during transmission. The two link types are:

- **Synchronous Connection Oriented Link (SCO)**

SCO links are point to point, connection-oriented links, primarily used for time-constrained voice applications. They offer no packet retransmission but have reserved time-slots for ordered and timely packet delivery. The link is symmetric and both the master and slave can transmit without polling. Slaves can have up to 2 SCO connections to different masters. The master can have up to 3 simultaneous SCO links outgoing to slave(s).

- **Asynchronous Connection-less Link (ACL)**

ACL links provide packet-oriented connection-less transmission primarily for data applications. The master designates link bandwidth and transmission is selected on a per-slot basis. The link can be asymmetric and slaves can only transmit when polled by the master. Link integrity is

protected with packet retransmission. Broadcast messages are also supportable over this type of link.

The baseband also provides mechanisms for error correction and security:

- **1/3 and 2/3 rate forward error correction (FEC) coding**

The FEC can be selected as needed based on the quality of the link. This provides some flexibility for bandwidth utilization at this layer.

- **Automatic repeat request (ARQ) with time-outs**

Bluetooth utilizes an unnumbered stop-and-wait protocol for transmissions. The receiver is expected to acknowledge a packet receipt in the next time-slot if the header error check and cyclic redundancy check both pass.

- **Security**

The Bluetooth LC allows one-way, two-way or no authentication based on a challenge-response system. This allows for personal device authentication. Additionally, encryption is provided to protect individual connections through a multi-bit stream cipher.

2.3.3.3 Link Manager

The Link Manager directs the Link Controller state machine shown in figure 2.2. It is a signaling mechanism that handles baseband link construction and destruction, security levels and control. It provides link abstractions to higher layers by communicating with other device Link Managers using the baseband protocols. It

negotiates the link types as well as monitoring of these links. Within the links, it designates frame types on a per-packet basis. It provides authentication and encryption to higher layers using the mechanisms in the baseband described in the previous section. Finally the LM is responsible for setting devices into the various active and low-power modes:

- **Sniff Mode**

Sniff mode is a low-power setting for Bluetooth devices where it is only required to listen for transmissions at a specified time interval (negotiated by the LM). At only these time-slots, the master can transmit to this slave.

- **Hold Mode**

In Hold mode, a device turns off its receiver for all packets except specific notices to reactivate the connection. Either the slave or master can send this notice. Extended periods where the receiver is off will save power for the client.

- **Park Mode**

Park mode allows devices to remain synchronized with the piconet but requires them to give up their active member address. They do not participate in the piconet but are alert for page commands.

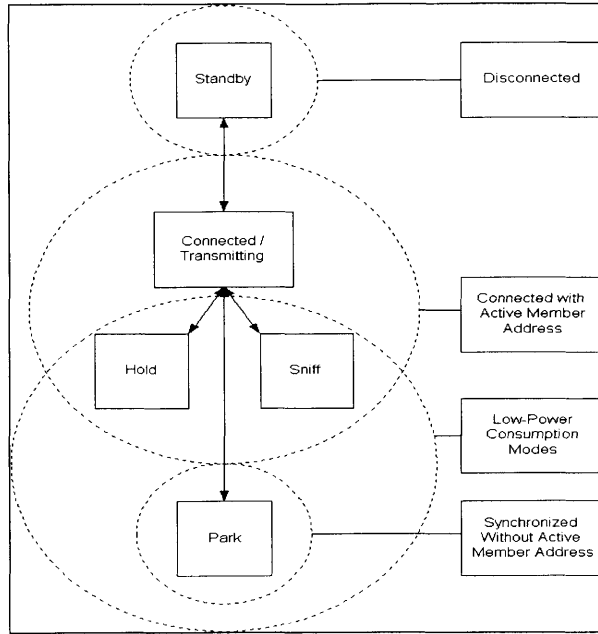


Figure 2.2: Device Mode State Machine

2.3.3.4 Logical Link Control and Adaptation Protocol

The L2CAP layer provides 3 types of channels to higher layers using the LM to setup baseband links. These are:

1. **Command signaling channels**
2. **Connection-oriented, bi-directional, symmetric or asymmetric channels for point-to-point communication**
3. **Connection-less, unidirectional, point-to-multipoint channels**

These channels allow for several key features of the L2CAP layer. Specifically, L2CAP manages higher-layer protocol multiplexing, signaling, and data segmentation and re-assembly. These essential tasks are handled at this layer because lower layers have no protocol ID's within packets and have very small transmission units. The L2CAP layer

provides an interface for larger packet sizes to be passed in and segmented packets to be sent to lower layers.

2.3.3.5 Additional Interface Protocols

Above L2CAP lie additional interface protocols to the application layer. These include, amongst others, the Service Discovery and Association Protocol (SDAP). SDAP is a mechanism for clients to find out what Bluetooth services are available on a specific device. Each device can run a SDAP server that can be queried for available services. To use SDAP, a client searching for a service will set up a L2CAP channel to a remote Bluetooth device. Next it will perform a query for services, asking for either a class of services or a specific service. The SDAP will respond with instructions to connect to the chosen service. Because SDAP's services are now complete, the client may connect through a separate channel to the specific service requested.

2.4 Available Bit Rate Applications

Many modern applications have redefined themselves to accommodate the lack of prevalent QoS systems in public networks. Because the resources available to a multimedia application can change over time (even more so in a mobile environment as noted previously), some real-time applications have become adaptive. The simple, inefficient solution to variable rate applications is over-provisioning resources to accommodate peak rate bandwidths and other requirements. This will waste significant resources that are very valuable in mobile environments.

Many methods of accommodating or designing adaptive multimedia applications exist, and at various layers in the network [VAN00]. The following table summarizes the techniques used:

Physical Layer	Adaptive power control
Data Link Layer	Error control and adaptive reservation
Network Layer	Dynamic rerouting
Transport Layer	Dynamic renegotiation of services
Application Layer	Variable error control, encoding and compression Bandwidth smoothing and rate shaping Adaptive synchronization

Table 2.2: Adaptive Media Techniques

Application layer techniques can be categorized into active or passive methods. Active methods will dynamically redefine the applications traffic characteristics to fully utilize the available network resources. Passive techniques have pre-prepared mechanisms of adapting to changing traffic conditions. Two available active techniques we would like to highlight are rate shaping and adaptive error control. Rate shaping is the dynamic adjustment of video encoding schemes to adapt to variable network conditions. The adaptable parameters include frame rate and quantization level. Adaptive error control is a technique where a variable forward error check is used to match resource availability with user's error service requirements [BT98, BFT99]. A popular example of many of these techniques is found in MPEG compression standards and Real Networks audio and video packages.

2.5 Dynamic QoS Management

Dynamic QoS management addresses the changing nature of QoS requirements of traffic sources and the available system resources. Allocated resources are adjusted based on the current level of performance achieved. Such a mechanism is necessary when applications are given control of network usage. For instance, variable bit-rate applications have changing bandwidth needs through the life of their connections. Therefore, many dynamic QoS methods exist for catering to these changes by monitoring and updating a set of QoS parameters associated with each flow [GVSS96, ROMW, FR97, ACH98, LB99, BS].

The following table summarizes some of the existing dynamic management methods:

In [BDDM93] flows are distinguished into several classes as characterized by their QoS specifications. By monitoring the number of refused connections and cell loss, bandwidth is dynamically updated.
In [FR97] heuristics are used to determine the amount of change for real time bandwidth adjustments based on assessment of the actual verses requested loss ratio.
In [ZK95] flows are individually divided into segments because of the variable requirements of real-time traffic. Thus QoS requirements are renegotiated on a per-segment basis for all traffic flows.
In [GVSS96] required bandwidth is estimated by counters kept at both the sender and receiver and these amounts are used to compute an adjustment value.
In [BS] the allocated resources for a flow are dynamically modified as a function of the current delay and loss performance achieved.

Table 2.3: Dynamic Management Schemes

The major deficiency in most dynamic QoS management schemes is their inability to renegotiate the level of service provided to *existing* flows during the time of admission of a *new* flow. In other words, attempts to downgrade services provided to other flows, within the negotiated boundaries, during the consideration process of admitting a new flow, are currently not being performed. Rather, a new flow, which can not be accommodated based on current resource utilization, is immediately rejected.

Research has been done on dynamically updating the conservativeness of the admission decision as a function of the system's present ability to provide the desirable QoS specifications in the network [BS]. Inaccuracies in the methods of estimation however, may exclude flows that could otherwise be admitted if existing flows were to relax their given QoS. Additionally, such schemes cannot fully utilize bandwidth if flows may be rejected even though resources are available.

CHAPTER 3

PROBLEM

3.1 Problem Statement

The provisioning of quality of service to multimedia applications is a complex problem due to many factors related to the applications themselves and the networks they operate in. Their stringent requirements in terms of bandwidth, delay, jitter and reliability coupled with the lack of effective QoS systems in public networks results in often unpredictable service quality. The added effects of mobile environments require a more robust and complex QoS system to handle such applications.

With this research, we attempt to provide recommendations towards a QoS system for the mobile Bluetooth piconet. Deficiencies exist in the proposed Bluetooth standard because only SCO links can efficiently handle Class I traffic. With the restriction of only 3 SCO links on a piconet, an ad hoc LAN will not be able to efficiently support multimedia traffic for all users on the piconet. If one were to transmit Class I traffic on an ACL link, then it is very difficult to guarantee an efficient QoS because all data is treated equally.

Another problem that arises and must be resolved to provide QoS is the response mechanism in Bluetooth. Currently, a master is not required to respond to a slave request in the next available time slot. The master can service other slaves first before responding. This becomes a problem when dealing with Class I services because there is no guaranteed consistency in delivery delay.

QoS maintenance gets increasingly difficult when the Bluetooth device is placed in a mobile environment. For example, if the device moves from one piconet to another piconet and wants to maintain its Class I connection, the new piconet will have to remodify all client QoS guarantees in order to accommodate this new member. Additionally, the old piconet will have excess resources that should be redistributed.

We see deficiencies in applying existing QoS management systems to such a mobile environment because current dynamic QoS management schemes lack the ability to renegotiate the level of service provided to *existing* flows at the point of admission decision of a *new* flow. The demands of mobility imply a strong effort by the QoS management system to accommodate as many users as possible in a changing environment. We feel that current systems do not attempt to downgrade or even evaluate the services realized by other flows so that a new flow can be accommodated. Thus new flows, for which resources can not be readily found, will be immediately rejected rather than further examined.

We also see the need for a robust notification mechanism that can provide for QoS management in both upstream and downstream directions. The notification system is necessary for a dynamic QoS scheme where changes in guarantees are continuously occurring. Clients can notify the master of their minimum acceptable bandwidth (or other QoS parameters) and the master can then take control by negotiating a realizable QoS with that client. All clients of the same master can be notified of changes in their realizable QoS. These changes can occur frequently as clients move between transferring different types of traffic. Additionally, when multiple clients are in a network, arbitration of QoS allocation must occur by negotiating with the master. For example, when one

client (C1) has an outgoing buffer full of pre-formatted data and a new client (C2) joins the network and requires an unrealizable QoS, then the master must be able to renegotiate with C2 or delay C2's acceptance into the network.

Because many modern multimedia applications are showing a general trend towards having adaptive rate mechanisms, we focus our work on the class of available-bit-rate applications that can dynamically adjust coding schemes in response to changing network conditions. Such applications can most benefit from a dynamic QoS system in an environment where allocation of resources may be ever-changing.

Therefore, our goal is to propose a new Bluetooth QoS architecture. For this system, we will make recommendations or contributions towards:

- **Recommendations towards the underlying QoS signaling system**
- **QoS parameter selection**
- **User utility model**
- **Recommendations towards dynamic scheduling techniques**
- **Dynamic admission control algorithms**
- **Recommendations towards interval maintenance techniques**

3.2 Solution Overview

To facilitate the new method of dynamic resource allocation, we begin with the need to introduce a modified set of QoS parameters. Upon request for admission, a client must present to the admitting node more descriptive information regarding the type of flow they request. Using these new parameters, we should be able to create an “image” of near-term traffic conditions that can help determine if and how to admit a new client.

With these parameters in consideration, we now present our system objective:

To allow maximum utilization of the network by admitting the maximum number of users and maximum, fair bandwidth to each user.

The consequences of this objective imply that our system will treat all flows equally, so that dynamic adjustments on QoS will be done in a fair manner. Our approach towards the admission decision is to develop an algorithm that ensures fairness in dynamically redistributing resources upon admission and during execution of network traffic. This algorithm will make tradeoffs amongst clients so that the most number of users can enter the system. The algorithm developed will maximize a utility function of the given parameters so that degradations still attempt to maximize user utility. The issue we face in the design of this algorithm is achieving optimal utilization for given network conditions.

For example, if a user requests entry into the system with a bandwidth requirement of 200 kbps, however the available bandwidth in the network is only 150 kbps, then the admitting node will attempt to renegotiate with the existing flows. Using the description of the flows presented at admission, the master will attempt to fairly redistribute other flows so that they can still operate in an agreeable region.

The concept of user utility we will develop will have the potential to be used for other purposes as well. Upgrading realized QoS can occur fairly using this utility function. If a flow leaves the network, then the remaining resources can be redistributed among the existing flows by using the same utility function to compare flows.

In addition to the development of the algorithms for optimal admission control, we also present surveys of existing dynamic scheduling techniques and make recommendations towards an effective interval maintenance system. Finally, we test the admission algorithms in a MATLAB simulation environment to see expected results in mobile networks.

Our goal in this work is to provide a robust mechanism for QoS management in Bluetooth systems. However, we feel our work can be extrapolated in to most mobile architectures because of its general nature and the similarity of Bluetooth to other mobile environments. Our work in QoS parameter selection and utility are particularly relevant in mobile systems and with the advent of available-bit-rate applications, our algorithms for admission control will be very valuable as well.

CHAPTER 4

DESIGN

4.1 QoS Framework

A quality of service framework forms a modular system of individual components that work together to provide QoS in a network. Developing a QoS framework allows for the integration of QoS methods at various network layers and the separation of tasks within them. This additionally provides an integrated system for end-to-end quality of service that is essential in any distributed network.

For our purposes, we design our QoS framework to include two primary divisions: QoS specifications and QoS mechanisms. QoS specification at the higher layers allows applications to indicate user requirements or flow characteristics in the form of higher-layer resources. Integrated into our definition of QoS specifications is also the QoS mapping phase, where these high level resources are translated into QoS parameters (such as bandwidth and delay) that can be regulated and monitored by the QoS system.

QoS mechanisms are responsible for using the QoS specifications to provide a realized service quality to a given user. The mechanisms needed can be broadly categorized as:

- **QoS Provisioning**

The provisioning task in QoS mechanisms is responsible for establishing a flow properly. It will perform admission testing to determine if the available resources can provide the requested QoS by a flow and resource

reservation to designate the end-to-end ownership of requested and deliverable resources.

- **QoS Control**

QoS control techniques provide real-time traffic administration at traffic time-scales. For example, flow control is a passive technique that uses either deterministic agreements with the flow or a feedback system (like ABR applications) to control the flow of data leaving a source. Flow shaping is the enforcement of a specific data injection pattern of a source. Using flow shaping in conjunction with flow scheduling, the process of ordering the forwarding of packets, performance guarantees can be made.

- **QoS Management**

The goal of QoS management is to insure the agreed upon service levels with a flow are being met. These key tasks include QoS monitoring and QoS maintenance. Monitoring on either an end-to-end or individual node basis is the process of assessing the level of service quality received by a flow at that location. QoS maintenance is the process of fine tuning the monitored parameters versus the requested ones.

- **QoS Signaling**

QoS signaling is an essential task because it is the underlying mechanism to a QoS framework that provides a communication system between nodes and layers for building, demolishing and renegotiating links. Such a system is invaluable in a mobile environment where links are frequently being created, destroyed and modified.

Our simplified QoS framework is depicted in figure 4.1.

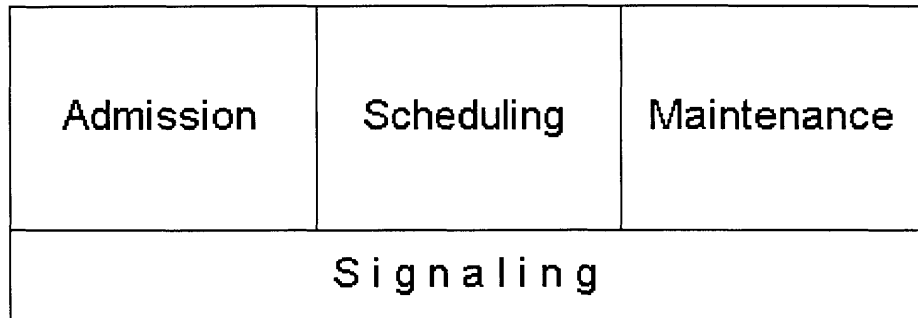


Figure 4.1: QoS Framework

The signaling mechanism used in our QoS framework must be comprehensive and robust enough to address the issues of mobility within a Bluetooth piconet. We seek for a signaling system that can accommodate available bit rate applications with a protocol for adaptive service. Thus the protocol must provide a communication mechanism so that applications can adapt to time-varying network resources. Additionally, it should offer signaling capabilities so that monitored conditions can be reported through various nodes in the system. Finally, it should offer mechanisms for fast reservations and recovery from broken links, as these occur frequently in mobile systems.

Although many signaling systems exist, we highlight one that we feel would be very applicable for the mobile Bluetooth environment. The INSIGNIA project [LZC00] is a QoS framework for adaptive services in mobile ad hoc networks. Within the framework is a very powerful and robust signaling system that would be a good choice for the highly dynamic Bluetooth piconet. Its key features include:

- **In-band signaling** - piggy-backed notification and reservation mechanism

- **Adaptive services with max/min bandwidth designation and scaling commands**
- **Soft-state resource management** – maintains flexible, temporary end-to-end connections
- **QoS reporting**

Using a signaling protocol such as that which underlies INSIGNIA, we can overlay admission control, scheduling routines and maintenance algorithms. In our framework, we create a conservative admission algorithm for paid resource reservation. Users request a specified level of service as described by the QoS parameters presented in the next section. These requests are measured against available resources by the admission algorithm. If admitted, scheduling routines provide QoS control for the sharing of a single channel by multiple flows. Finally the maintenance schemes will offer fine tuning to ensure realized and requested QoS levels agree. We do not concern ourselves with the payment schemes or cost structure.

4.2 QoS Parameter Selection

QoS parameters vary at different layers of the protocol stack. As noted previously, QoS mapping procedures are responsible for translating higher level requests in to the pertinent parameters at each layer. In a broad sense, all parameters can be characterized in to 5 categories. Performance parameters quantify received resources such as bandwidth, delay, etc. Format parameters specify rates, data formats and compression techniques. Synchronization parameters describe coupling between various flows. Cost descriptors specify network utilization prices for given resources or service

levels. User preference parameters may be particular opinion-based descriptors of received service, such as image or sound quality.

To facilitate a robust method of dynamic resource allocation targeted for available bit rate applications, we introduce a modified set of QoS parameters. Upon request for admission, we propose that a client present to the master the following information for the requested flow:

- **Maximum and minimum bandwidth (B)**

These parameters reflect the highest level of bandwidth the application could use while still increasing its utility and the lowest level of bandwidth the application can tolerate to still function properly.

- **Acceptable bit error rate (E)**

This parameter defines the application's flexibility in terms of bit errors in the data. A highly tolerant application such as voice would have a very different BER requirement than would a data application.

- **Transfer size (T)**

This parameter may be used by data or multimedia applications that can accurately estimate transmitted file size before connecting.

- **Maximum and minimum tolerable delay (D)**

This parameter is essential for interactive applications where a long delay between sending a request and receiving a response is not acceptable. An average delay window allows clients to specify acceptable delay levels.

Therefore, each requesting flow submits the following set **{B, E, T, D}** and for each admitted flow the master maintains this set for the given flow until the flow halts. In

addition to these parameters, the master also keeps track of total bandwidth usage (at current time), the current bandwidth usage of each flow in the piconet, and the length of time each flow has been active.

4.3 Utility Model

In this section we introduce a concept of user utility. When multiple QoS resources are available to users, comparison of these resources amongst multiple users becomes challenging. Alternatives such as higher bandwidth and high error rate may be better than low bandwidth and low error rate. These distinctions are dependent on application type or user preferences. Therefore we must define user utility, a systematic method of measuring the relative quality of service a user is receiving versus what he or she requested. Summing the user utility for all users gives an approximation to total satisfaction in the network, a value which should be optimized for best system performance.

To understand utility on a per resource basis, we look first at bandwidth as an example. Holding other resources constant, a user's utility (satisfaction) as a function of increasing bandwidth may look like figure 4.2.

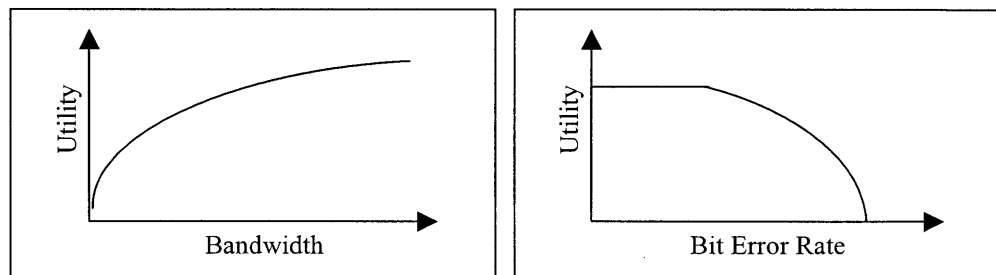


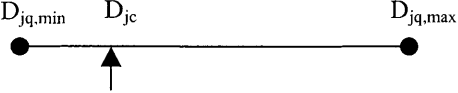
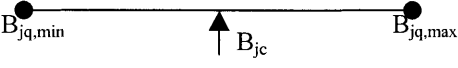
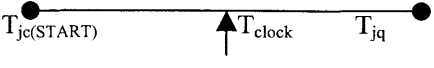
Figure 4.2: Utility Functions

This is derived from an application which performs better as bandwidth increases, however as received bandwidth becomes very high, performance is only marginally improved (such as in video applications). Other applications may receive no performance benefit unless a given bandwidth is achieved and beyond that bandwidth, there is no added utility benefit. This situation is true for some real-time applications that can not adapt to changing network conditions.

Error rate on the other hand has the inverse effect on user utility as is seen in figure 4.2. At a high error rate, the utility is zero because the application cannot operate over such a high BER channel. As the error rate lowers however, utility increases until a point where lowering the error rate further has no effect on the application. For example, voice and video applications can tolerate reasonable error rates because humans cannot notice the improvement after a certain BER is achieved.

Our linear utility model is presented below:

- **Assume all users receiving minimum QoS requested**
- **User j is receiving QoS = $\{D_{jc}, E_{jc}, B_{jc}, T_{jc}\}$**
- **User j requested QoS = $\{D_{jq}, E_{jq}, B_{jq}, T_{jq}\}$**

 $D_{j,util} = \frac{D_{jc} - D_{jq,min}}{D_{jq,max} - D_{jq,min}}$ <p>% Delay Performance Beyond Minimum Requested</p>	$E_{j,util} = \frac{E_{jq} - E_{jc}}{E_{jq}}$ <p>% Bit Error Performance Beyond Quantity Requested</p>
 $B_{j,util} = \frac{B_{jc} - B_{jq,min}}{B_{jq,max} - B_{jq,min}}$ <p>% Bandwidth Allocation Beyond Minimum Requested</p>	 $T_{j,util} = \frac{T_{clock} - T_{jc}}{T_{jq}}$ <p>% Requested Size Transferred</p>

We further define a total user utility that is based on the weighted sum of delay, error and bandwidth utilities. The transfer size parameter is not included because we view it as a supplementary parameter that will be useful towards fair resource distribution techniques. Assigning weights allows users to differentiate their own emphasis on a given QoS parameter. For example a user who values bandwidth more than delay (such as in a data application) would have a relatively higher bandwidth weight versus delay weight. We feel the value of these weights can either be assigned directly by the user at call-request or assigned from a lookup table, by the process of declaring an application type. Thus we have:

$\alpha_{DELAY} =$	% weight of delay utility
$\alpha_{ERROR} =$	% weight of error utility
$\alpha_{BW} =$	% weight of bandwidth utility
$\alpha_{DELAY} + \alpha_{ERROR} + \alpha_{BW} = 1$	

$$U_j = \left[\left(\alpha_{DELAY} \times D_{j,util} \right) + \left(\alpha_{ERROR} \times E_{j,util} \right) + \left(\alpha_{BW} \times B_{j,util} \right) \right]$$

Having defined user utility and total system utility, we now present its uses in the process of admission decision and interval QoS updating or monitoring. In the admission process, when a user can not be clearly admitted (i.e. $B_{jq,min} < B_{avail}$) an admission decision must be made. We can use the user utility function to determine how well existing clients are being treated in comparison to their requested QoS and one another. From this evaluation, we can further determine which clients should be downgraded and by how much in order to admit new members. In the process of interval QoS updating, upgrades and downgrades may be necessary to ensure clients are realizing their QoS requests. The individual user utility quantities for each QoS variable, in conjunction with the weightings, can be used to determine how tradeoffs should be made internal to a client's QoS levels. The total user utility quantities can be used to determine which clients should be modified if the system is overloaded.

4.4 Literature Survey of Dynamic Schedulers

We present a survey of existing QoS based scheduling mechanisms with an emphasis on dynamic schedulers. Most scheduling techniques can be categorized into the following groups:

- Priority-based
- Latency-based
- Rate-based

- Dynamic Resource Allocation

Priority-based schedulers order packets based on pre-assigned, relative priorities. Latency-based algorithms transmit packets based on their delay requirements. Rate-based algorithms give minimum allocations to users sharing the same resources. Dynamic resource allocation schemes use changing network conditions as feedback for dynamically adjusting resource sharing.

4.4.1 Priority-based Scheduling

Priority-based scheduling approaches use relative priorities to give preferences to different flows. Traditional priority scheduling at a buffer, involves transmitting highest priority packets first before lower priority packets are serviced. If a flow is given highest priority, then all of its packets currently at a switch are transmitted in a FIFO ordering before the next priority flow is allowed to transmit. Some schemes offer preemption as well, which will interrupt a lower priority flow's transmission if a higher priority packet arrives and is ready to be sent. Priority scheduling has been shown to have significantly better delay performance for higher priority flows, however it performs poorly for lower priority transmissions because they can be indefinitely set-aside for higher priority packets.

A solution to the previous problem is priority jumping [PTW88]. If each priority is viewed as a separate queue, then each queue can be associated with a given set of delay bounds. Then as packets in lower priority queues have experienced a certain amount of delay, they are moved to the next higher priority queue until they are finally transmitted.

This provides a delay bound for lower priority packets as well as higher priority ones. This method can not only guarantee average delay bounds, but it can also serve statistical loss ratios for different priorities.

4.4.2 Latency-based Scheduling

Many real-time applications have stringent delay requirement for individual packets. If these requirements are not met, then some packets may be deemed useless by them. Therefore, latency-based schedulers order packets based on the delay they are experiencing or requesting. A simple mechanism for ordering packets is Minimum Laxity Threshold (MLT) [CKT89]. For each packet, this system computes the amount of time the server can spend serving other packets before this one must be served to meet its delay requirements. Then all packets are ordered based on this calculation as they enter the switch. MLT has been proven to provide highest utilization for flows with strict delay requirements. A variation of MLT is Shortest-Time-To-Extinction. This mechanism drops over-delayed packets in such a way as it minimizes the number of total packets that do not meet their delay specifications. FIFO+ is a scheduling algorithm that offers predicted delay by ordering packets based on their expected arrival times. It compares the average delay along each link of the traversed path to the experienced delay to set an ordering such that packets receive closer to the average delay.

Latency-based algorithms experience problems similar to prioritization schemes. When packets of high MLT times are in the presence of many low-MLT packets, they may never achieve their desired QoS requests because they will be rarely served. This becomes a problem when flows have diverse QoS requests such that some have firm

requirements and some have looser requirements. The firm requirements will be met well, but the less stringent requirements may not be met at all.

4.4.3 Rate-based Scheduling

Rate-based algorithms use resource reservation to guarantee a minimum bandwidth to sources that behave well. Flows use descriptors to characterize their traffic. These parameters are often peak rate and burstiness. The General Processor Sharing (GPS) scheme divides network capacity amongst flows in a weighted manner. It has been shown that in conjunction with leaky bucket flow shaping, GPS can guarantee an upper bound to packet delay. At a packet level, Weighted Fair Queuing (WFQ) and VirtualClock are well known GPS approximations. They isolate flows to guarantee a delay bound on individual packets as well as provide an average throughput for each flow. Rate-based schemes have been shown to utilize resources poorly for very bursty traffic.

4.4.4 Dynamic Resource Allocation

Dynamic Resource allocation attempts to efficiently match the changing QoS needs to the time-varying network conditions. In chapter 2 we presented dynamic QoS management techniques and here we will further detail mechanisms focused specifically on resource allocation. Dynamic resource allocation schemes are generally responsive to either QoS feedback or changes in presented traffic. One method that responds to QoS feedback is found in [BOL93]. This scheme categorizes flows by their QoS requests into separate classes. An allocation controller dynamically redistributes resources by

computing a cost function of the number of refused connections and total cell loss. This allows admission decisions to be made by delay and loss estimations. In the Dynamic Search Algorithm (DSA+) method presented in [FR97], resource allocations are dynamically adjusted based on measured QoS. The method attempts to achieve a requested loss ratio by adapting bandwidth so that the actual loss ratio approaches the desired one. It uses a stable heuristic that is a function of number of arrivals, losses and requested loss ratios.

Dynamic methods that rely on traffic change as opposed to QoS measurements are more appropriate for variable bit rate applications that have continuously changing needs and cannot tolerate the slow process of renegotiation based on QoS measurements. Instead, such methods try to evaluate the variability of traffic. One proposed scheme [ZK95] uses time segments with exact traffic quantity bounds to guarantee delay for short periods. Thus, sources can renegotiate allocations over certain time periods as needed to account for highly bursty periods of traffic. Other methods [ADA96] use dynamic linear error predictors to estimate future bandwidth requirements of a traffic source.

4.4.5 Conclusion

In our opinion, the combination of a rate-based scheduler and a dynamic allocation scheme provide the most benefit to handling paid-for resource reservations in broadband communication solutions using multiple frequencies or multiple time slots. The rate-based scheduler will guarantee the minimum throughput request in the parameter specification. The dynamic allocation scheme can allow for over-committal of resources (admitting additional users) by adjusting the minimum throughput value when

resources are not being efficiently utilized. Thus, the rate-based scheduler can have the minimum throughput dynamically adjusted as needed.

4.5 Dynamic Admission Control Algorithm

A mobile ad hoc network's admission control system must be able to quickly redistribute and re-negotiate resources amongst its members if it wishes to admit a new client under scarce resource conditions. Such a system is superior to a static one in which no attempt at reallocation is made because in mobile environments, changes in network conditions occur frequently as users move in and out of piconets rapidly. Modern multimedia applications have tolerable bandwidth windows which can be used to re-negotiate received bandwidth so that new clients may be admitted even if their demands are not immediately available under current network conditions. By reallocating the bandwidth being used, additional resources may be freed for use by the new client.

To address the issue of fair redistribution of resources in a mobile environment, we have developed admission algorithms that optimize resource distribution under various constraints. We define the admission problem as:

- N clients in piconet at time t_0 : (c_1, c_2, \dots, c_N)
- Each client c_i has:
 1. A tolerable bandwidth window $BW_i = [B_{i,\min} \dots B_{i,\max}]$
 2. A client or system defined $\alpha_{i,BW}$ which, as defined previously, is a constant associated with either the weight of the bandwidth resource of that user or the weight of that link

3. A received bandwidth B_i
- Client c_{N+1} arrives at time t_1 with requests of BW_{N+1} and $\alpha_{N+1,BW}$
 - C = Total network capacity (kbps)

Our objective, based on this problem statement, is to find the fair redistribution of bandwidth on each user after time t_1 so that the following conditions are met:

1. User c_{N+1} can be successfully admitted in to the network after a possible redistribution of resources
2. Redistributions (if necessary) occur such that we maximize the total utility of the network
3. All users are receiving at least their minimum requested allocation
4. Network capacity C is not exceeded

To achieve a measure of fairness so that we can compare different users, we have developed a concept of utility for each resource parameter in our system. Because our admission algorithms are concerned with bandwidth redistribution, we concern ourselves only with the bandwidth utility at this time. We define a user utility u_j as a representation of the excess bandwidth user j is receiving beyond his or her minimum request. This can be realized as the following:

$$U_j = \frac{B_j - B_{j,\min}}{B_{j,\max} - B_{j,\min}}$$

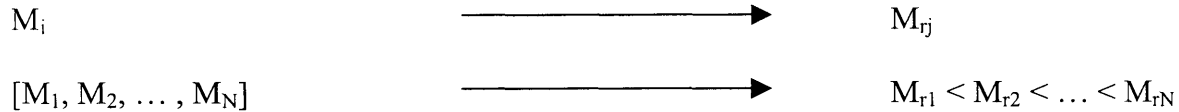
Additionally, we define total utility as the sum of all weighted user utilities for clients currently maintaining flows: $U_T = \sum_N \alpha_{j,BW} \times U_j$

In the following we present two algorithms, each which looks to optimize total utility but under different constraints. The first aims to optimize total utility without regard for equitable treatment amongst users. The second tries to make fair reductions to create bandwidth for new users while considering their relative weights in the redistribution scheme. In both, the equivalent situation is the following: a user c_{N+1} arrives with a request such that the minimum required bandwidth is beyond the unused bandwidth in the system and the additional amount needed to accommodate that user can be created by redistribution of resources. This is also stated as: $B_{N+1,\min} > C - \sum_N B_i$.

Algorithm 1:

Using this user utility, the proposed algorithm operates as follows. We would like to create the necessary bandwidth to accommodate the newest user while maximizing total utility of the existing users (at a normalized position in his bandwidth window that is fair relative to the existing users in the system). Because we are trying to optimize total utility we want to take as much bandwidth as possible from users who make little contributions to total utility. This is called a greedy algorithm because it takes as much as it can for the lowest cost (the greedy choice). It finds a globally optimal solution by making locally-optimal choices. We will prove formally that total utility achieves the optimal value in following sections.

A user's contribution to total utility U_T can be found on a per kbps basis. A user contributes $\alpha_{i,BW} \times U_i$ to total system utility. Over the interval $[B_{i,min}, B_{i,max}]$ a user's utility varies linearly from 0 to $\alpha_{i,BW}$. Therefore for every 1 kbps, a user contributes $M_i = \alpha_{i,BW} / BW_i$. Using this value, we rank users $[1 \dots N]$ by increasing M_i :



Define *bwCreated* as the present amount of bandwidth created through the current iteration and *bwNeeded* as the amount of bandwidth we are attempting to create:

$r_j = 1; bwCreated = 0;$

While (*bwCreated* < *bwNeeded*)

 For user r_j with marginal utility M_{r_j} :

$$X_{r_j} = \min ([B_{r_j} - B_{r_j,min}], bwNeeded - bwFound)$$

$$bwCreated = bwCreated + X_{r_j}$$

$$B_{r_j} = B_{r_j} - X_{r_j}$$

 increment (r_j)

end

In other words, from the user with the lowest value of M , we take as much bandwidth as we can, then from the user with the next lowest, we take as much as we can...and so forth until *bwNeeded* has been created. This guarantees the total utility is optimal and will give the same solution as the SIMPLEX optimization algorithm.

EXAMPLE:

$N = 4$ users

- User 1, 2 and 3 have bandwidth windows of $[10, 20]$ and $\alpha_{1,2,3,BW} = 0.1$
- User 4 has bandwidth window of $[100, 140]$ and $\alpha_{4,BW} = .9$
- Assume $C = 200$ kbps, and all of it is being used (i.e. $U_{1,2,3,4} = 100\%$)

Assume a new user requests entrance and needs 35 kbps created for him. Then the algorithm ranks users by their M values.

$$M_{1,2,3} = .1 / 10 = .01$$

$$M_4 = .9 / 40 = .0225$$

Since $.01 < .0225$ we will begin our reductions from M_1 to M_4 (in order).

$$\text{User 1: } \min(10, 35) = 10 \text{ (new } U_1 = 0)$$

$$\text{User 2: } \min(10, 25) = 10 \text{ (new } U_2 = 0)$$

$$\text{User 3: } \min(10, 15) = 10 \text{ (new } U_3 = 0)$$

$$\text{User 4: } \min(40, 5) = 5 \text{ (new } U_4 = .875)$$

The sum of created bandwidths is 35 and $U_T = .1 \times 0 + .1 \times 0 + .1 \times 0 + .9 \times .875 = .7875$

As an example that other solutions are not optimal, assume we take all 35 kbps from user 4 instead of the way we did above. So:

$$U_T = .1 \times 1 + .1 \times 1 + .1 \times 1 + .9 \times .125 = .4125$$

Thus there is a significant difference between the optimal and a sub-optimal solution. The greedy method will always result in the optimal total utility.

Proof of optimality:

Proving a greedy algorithm's optimality generally involves showing optimal substructure and proving that the problem exhibits the greedy-choice property. Optimal substructure is the claim that the optimal solution consists of optimal solutions to its sub-problems. The greedy-choice states that the globally optimal solution can be found by making locally optimal choices. We will first show that this problem exhibits the optimal substructure property:

Consider the optimal (least expensive in terms of utility) formation of *bwNeeded*. If we remove user *i*'s contribution of b_i then the remaining ($bwNeeded - b_i$) kbps must also be an optimal formation of what remains. If however, we can make a re-arrangement of what was available to get a lower cost solution to this sub-problem, then when b_i is added back in, the total solution is "more optimal." Therefore there is a contradiction and the sub-solution must also be optimal.

We now prove that the problem also exhibits the greedy-choice property. To show this, we first prove that the greedy choice is necessarily part of the optimal solution. Then we show that the greedy choice can be made first. And finally, using induction, it can be shown that the greedy choice is best at each iteration.

Once again, consider the least expensive distribution that creates $W = bwNeeded$.

If we define a weight, W_i as the available bandwidth of user i then, the total value (towards utility) of the available bandwidth of user i is simply

$$V_i = W_i \times \frac{\alpha_i}{B_i}.$$

$P(i)$ is the amount of bandwidth we take from user i , therefore the total value of $bwNeeded$ will be:

$$\sum_{i=1}^n P(i) \times \frac{V_i}{W_i}$$

We further define V_L, W_L as the value and bandwidth available respectively for the user with the lowest value / weight ratio. So we must show that as much as possible of the lowest value/pound item (L) must be included in the final solution: if some of item L is left and $P(j) \neq 0$ for some $j \neq L$, then replacing some of j with some of L will yield a lower total value because:

$$P(j) \times \frac{V_j}{W_j} \geq P(j) \times \frac{V_L}{W_L}$$

since we know by definition that $\frac{V_j}{W_j} \geq \frac{V_L}{W_L}$.

We have shown that as much of the greedy choice as possible is part of an optimal solution. Next, we show that the *initial* greedy choice of item L leads to the optimal solution. There are two cases. If $W \leq W_L$ then we simply set $P(L) = W$ and for all $j \neq L$, $P(j) = 0$. If however $W > W_L$ then room still remains after a greedy choice is made. Let $P(i), P(j) > 0$ for some $i \neq j$. Assume item i was the first choice and $j = L$ was a subsequent choice. To show that j could have been chosen first instead of later, consider

that $V = P(i) \times \frac{V_i}{W_i} + P(L) \times \frac{V_L}{W_L}$. Had we chosen item P(L) of item j first then V would

be $V = P(L) \times \frac{V_L}{W_L} + P(i) \times \frac{V_i}{W_i}$. These are of course equivalent, therefore the greedy

choice could have been made first.

With the optimal substructure property, we showed that the total solution consists of optimal solutions to sub-problems. We also showed that the initial greedy choice is part of the optimal solution and the first selection reduces the problem to an equivalent, but smaller optimization problem. Thus, by induction on the number of choices made, an optimal solution can be reached by making the greedy selection at each iteration. Therefore, our algorithm for bandwidth reductions is optimal.

Algorithm 2:

The second algorithm addresses the problem of maximizing total utility with the additional constraint of using a fair reduction scheme. When a new user requests admission, we attempt to reduce existing users in such a manner that their relative weights are considered so that relative equal reductions can occur. Using the same user utility and the link weights, we can solve this problem. Once again, the user is entering at a normalized position in his bandwidth window that is also fair relative to the existing users in the system. We approach this problem mathematically:

Define B_i' as the new bandwidth user i is allocated after reductions take place and $\Delta B_i = B_i - B_i'$. Because we create only exactly what is needed: $bwNeeded = \sum_N \Delta B_i$. We

begin with the assumption that equal reductions of utility are made across all users.

Thus:

$$\alpha_{1,BW} \times \Delta U_1 = \alpha_{2,BW} \times \Delta U_2 = \dots = \alpha_{N,BW} \times \Delta U_N$$

This translates into the following set of equations:

$$\begin{aligned} \alpha_{1,BW} \times \left(\frac{B_1 - B'_1}{BW_1} \right) &= \alpha_{2,BW} \times \left(\frac{B_2 - B'_2}{BW_2} \right) \\ \alpha_{2,BW} \times \left(\frac{B_2 - B'_2}{BW_2} \right) &= \alpha_{3,BW} \times \left(\frac{B_3 - B'_3}{BW_3} \right) \\ \dots & \\ \alpha_{N-1,BW} \times \left(\frac{B_{N-1} - B'_{N-1}}{BW_{N-1}} \right) &= \alpha_{N,BW} \times \left(\frac{B_N - B'_N}{BW_N} \right) \end{aligned}$$

If we define $M_i = \alpha_{i,BW} / BW_i$ and solve the above equations for ΔB_i as a function of ΔB_1

we find the following result:

$$\forall i \in [1 \dots N]: \Delta B_i = \frac{M_1}{M_i} \Delta B_1$$

Now we must simply find ΔB_1 . Substituting this previous result into $bwNeeded =$

$\sum_N \Delta B_i$, we see that $\Delta B_1 = \frac{bwNeeded}{J}$ where $J = M_1 \times \left(\frac{1}{M_1} + \frac{1}{M_2} + \dots + \frac{1}{M_N} \right)$. Thus we

reach the final result:

$$\forall i \in [1 \dots N]: B'_i = B_i - \frac{M_1}{M_i} \times \frac{bwNeeded}{J}$$

This example will work in a large majority of cases where the values of α do not have a very large distribution. When that occurs, users with small values of α may not have enough bandwidth that can be taken from them to make equal relative reductions across all users. Thus the optimal solution is to take as much from those users as possible. This will result in a distribution that is optimal under the given constraints. If all users weigh bandwidth equally then the relative weights will drop out of the equations and we are left with a solution that works under all conditions.

EXAMPLE:

- $N = 2$ users
- User 1 has bandwidth window of $[100, 375]$ and $\alpha_{1,BW} = 0.4$
- User 2 has bandwidth window of $[100, 375]$ and $\alpha_{2,BW} = 0.8$
- Assume $C = 750$ kbps, and all of it is being used (i.e. $U_{1,2} = 100\%$)

Assume a new user requests entrance and needs 100 kbps created for him. Execution results in $bwNeeded / J = 66.67$. Thus $\Delta B_1 = 66.67$ and $B_1' = 375 - 66.67 = 308.33$ kbps and $B_2' = 375 - 33.33 = 341.67$ kbps. The total utility $U_T = 0.4 \times .757 + 0.8 \times .879 = 1.01$.

4.6 Interval Maintenance System Overview

We propose recommendations for an interval maintenance system that would work well in conjunction with an end-to-end maintenance mechanism for broadband frequency hopping systems. With a robust underlying signaling system, many essential

functions such as link recovery or re-routing are well accommodated for already. Thus, the responsibilities of ensuring realized QoS is the same as requested QoS are left to the maintenance algorithms. We are of course working under the assumption that the wireless solution offers facilities for the measurement of QoS resources received by specific devices in the networks either on a real-time or interval basis. If such facilities are available, then further improvements can be made with the addition of utility balancing, network assistance, control-knob adjustments and variable forward error checking.

- **Utility Balancing**

For long-term steady state equality in a network, we introduce utility balancing - a mechanism for gradually bringing all existing members of a network to equal levels of utility. If the flows in the network can tolerate more frequent adjustments to their available bandwidth, then upon some regular interval or when a new client is admitted, the bandwidth allocated to each flow can be incrementally adjusted toward the mean bandwidth utility in the network. If larger variations in utility occur across a network (i.e. users are very diversely receiving the excess resources available) then incrementally balancing utility will provide for equal treatment amongst users for longer-term connections.

- **Network Assistance**

Existing maintenance systems provide for fine-tuning adjustments when specific flows are not receiving their requested QoS specifications. When a well-behaved flow is experiencing an undesired service level, minor

modifications can be made to help achieve the requested levels. In general these modifications are based on stochastic estimations and are beyond the scope of this thesis. However, in a paid for reservation scheme, additional resources may not be available because they are already being utilized. Therefore, in a window-specified bandwidth system, we propose network assistance that uses user utility to compare the excess resources received by clients in the network. If a user is not receiving his minimum bandwidth request, then a fair redistribution algorithm (such as Algorithm 2 proposed previously) can help to create new bandwidth from those users that are receiving beyond their minimums. Thus, we are essentially taking bandwidth from those that have more than necessary and give to those that do not have enough to work within their operating region.

- **Control-knob Adjustments**

In previous sections, we have introduced utility concepts for individual resources, in addition to weights associated with these resources. Using the utility weights, we can now differentiate resources within individual users. We view each resource and its associated weight as being controlled by an adjustable “knob.” Fine tuning adjustments can be made as tradeoffs amongst resources.

- **Variable Forward Error Checking**

We recommend the implementation of a variable forward error check such that applications that can tolerate slightly errored packets, may use a lower quality forward error checking algorithm. Thus they will not have to suffer through re-

transmissions and ultimately reserve less bandwidth. Variable FEC schemes can be found for audio and video in [BT98, BFT99].

- **Departure Induced Bandwidth Recovery**

Finally, on the interval maintenance schedule, we leave open the possibility of re-allocation of recovered bandwidth when a departure occurs. Re-distributing the newly available bandwidth by essentially running the inverse of algorithms 1 or 2, can give a new fair allocation. It may additionally be beneficial to allocate more bandwidth to those users that are close to completion because that bandwidth will soon be free again for new clients.

CHAPTER 5

SIMULATION AND RESULTS

5.1 Simulation Environment

For visualization and evaluation of our admission algorithms, we have developed a simulation environment in MATLAB. The simulator allows for the substitution of any admission decision algorithm into an ad hoc local area network. The network begins with zero users and grows with a Poisson arrival schedule. The arbitrary inter-arrival exponential has mean of one clock tick (or any unit of time). Each arrival presents its QoS requirements to the network in the form:

1. A tolerable bandwidth window $BW_i = [B_{i,\min} \dots B_{i,\max}]$
2. A client or system defined $\alpha_{i,BW}$ which, as defined previously, is a constant associated with either the weight of the bandwidth resource of that user or the weight of that link

Next, given the current load on the network, the admission decision algorithm evaluates whether or not the client can be successfully admitted into the network. Ultimately, the new client is either rejected or admitted into the system. Each user remains in the network for a randomized time interval and there is a cap on the maximum number of users in a network. Please see figure 5.1 for the simulator state machine.

The simulator was designed such that we could make network snapshots at any given time. We can view how utility changes for any admission decision in the history of

the network, as well as see what parameters were presented to the admission algorithm that lead to the admit or deny decision. The simulator code is supplied in Appendix A.

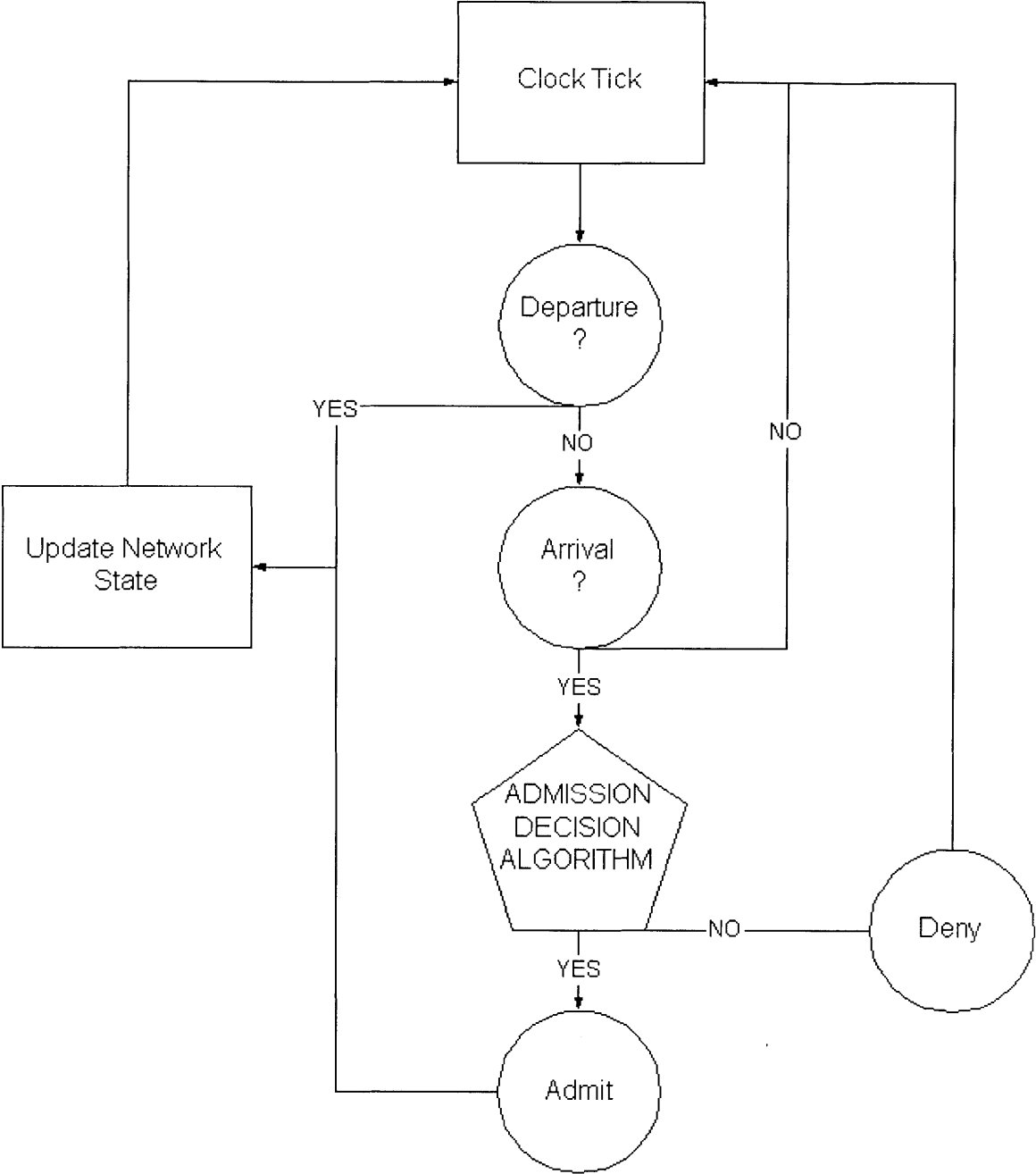


Figure 5.1: Simulator State Machine

5.2 Results

We substituted Algorithms 1 and 2 from Chapter 4 into our simulation environment and benchmarked them against a best-effort algorithm that gave each incoming user the highest available bandwidth in its negotiable range on a first come, first serve basis. The organization of the testing strategy was as follows:

- Demonstrate correct operation of Algorithms 1 and 2 in simulated environment
- Compare each algorithms efforts to admit as many users as possible into the piconet, and the average number of users per unit time
- Measure system bandwidth utilization under each admission algorithm

Beginning with Algorithm 1, we see an existing piconet from $t = 1$ with the following properties:

	Min BW request	Max BW request	BW weight	Received
User 1	277.79 kbps	346.37 kbps	.666	277.79 kbps
User 2	172.95 kbps	222.97 kbps	.333	222.97 kbps
User 3	137.52 kbps	160.09 kbps	.666	160.09 kbps

At $t = 6$ a new user requests admittance and execution of Algorithm 1 results in the reductions of users 2 and 3 in order to accommodate the minimum bandwidth request of the new user at 160.03 kbps. These reductions can be seen in figure 5.2 (a). Correspondingly, the excess utilities of users 2 and 3 change from 100% to 0% and 7.6% respectively. The utility variations can be seen in figure 5.2 (b).

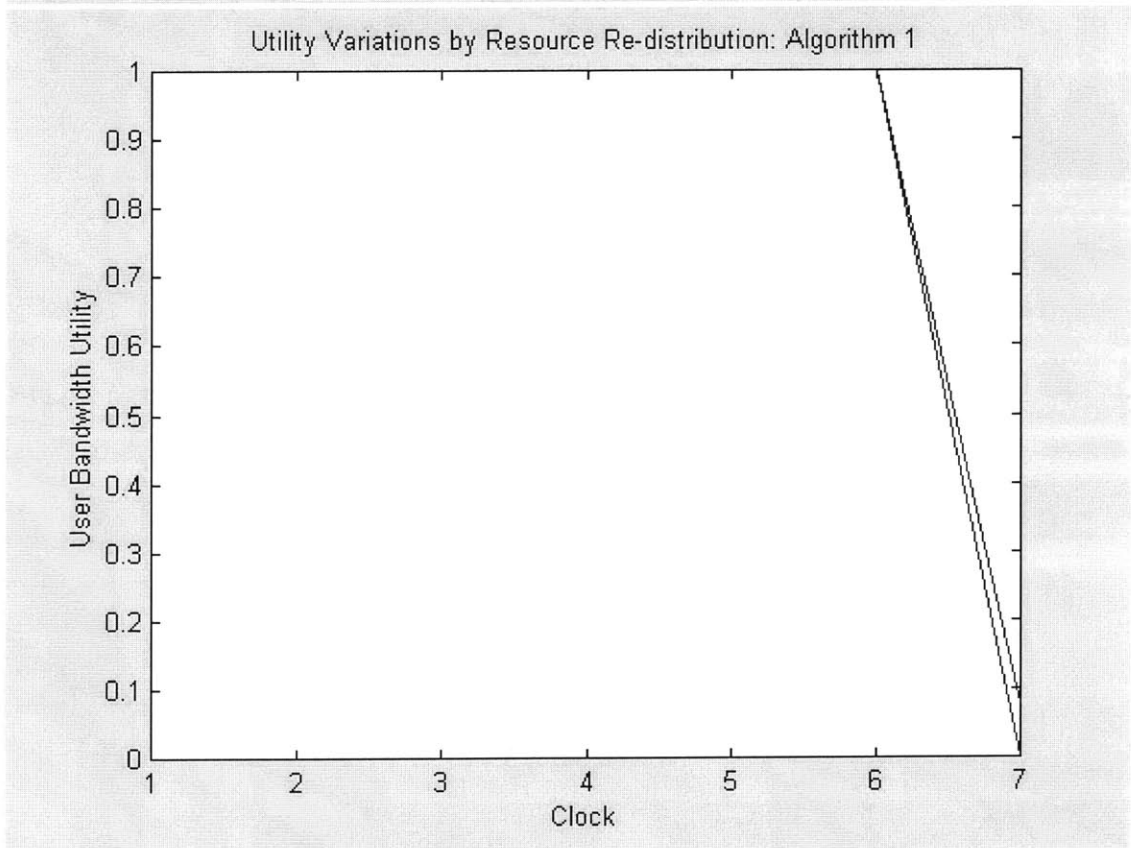
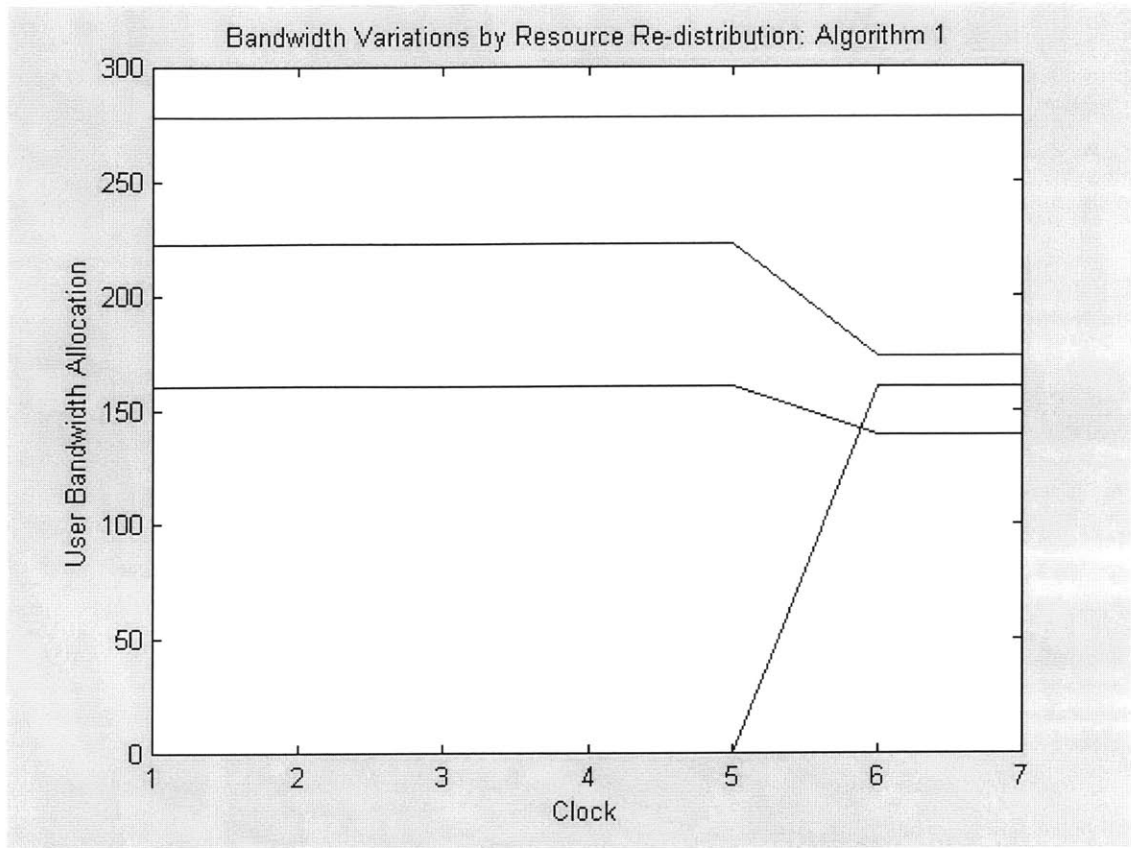


Figure 5.2 (a) and (b): Algorithm 1 Results

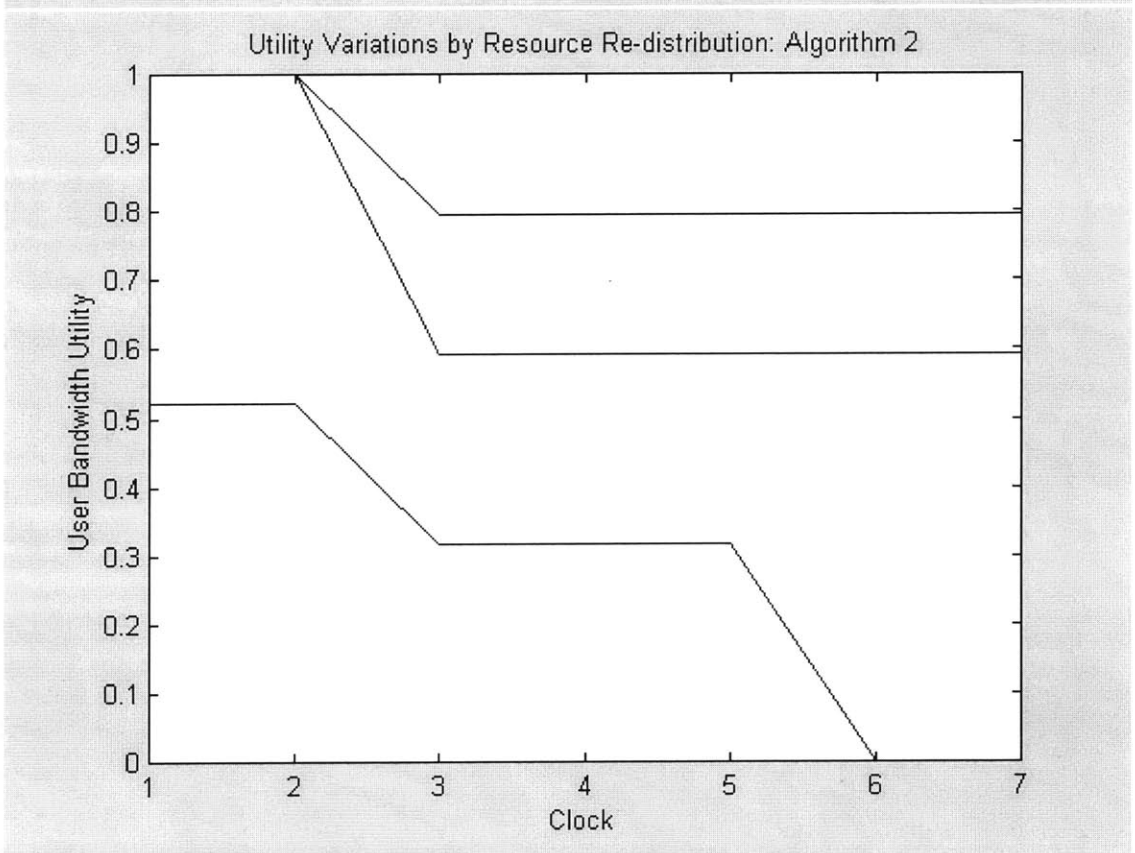
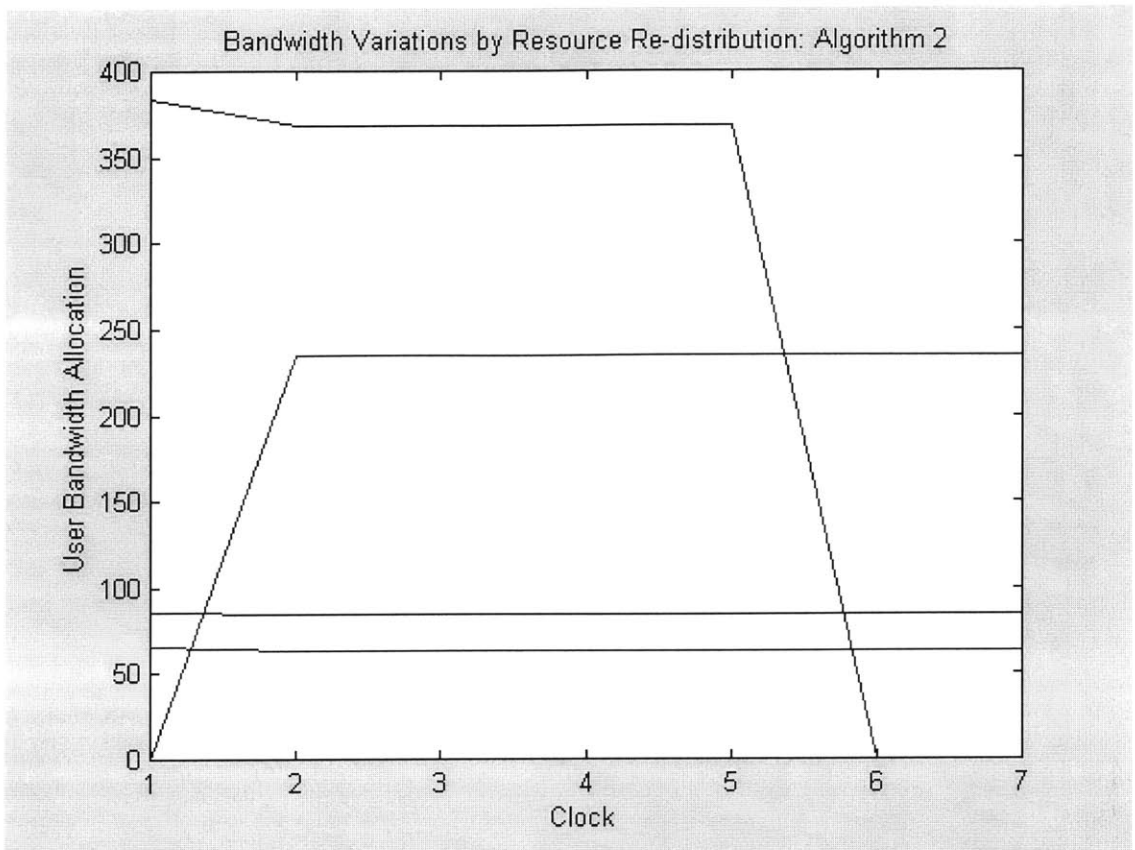


Figure 5.3 (a) and (b): Algorithm 2 Results

We now present the results of Algorithm 2 simulations. The existing piconet at $t = 1$ is described in the following table:

	Min BW request	Max BW request	BW weight	Received
User 1	342.69 kbps	421.64 kbps	.666	383.94 kbps
User 2	59.11 kbps	65.34 kbps	.333	65.34 kbps
User 3	76.80 kbps	86.14 kbps	.666	86.14 kbps

To accommodate the introduction of a new user at 235.16 kbps, Algorithm 2 re-distributed bandwidth for the existing users as can be seen in figure 5.3 (a). As a result, user utility was also modified as demonstrated in figure 5.3 (b).

Having confirmed proper execution of the algorithms, we performed benchmark simulations to compare each algorithm's ability to accommodate as many users in the network as possible. We created 10 sample arrival schedules for 1000 time units, using the Poisson process with one time unit as the mean inter-arrival time. Next all 10 inputs were presented to:

- Network 1, which ran Admission Algorithm 1 (the greedy algorithm)
- Network 2, which ran Admission Algorithm 2
- Network 3, which ran the best effort, benchmark algorithm

For each network, we computed the average (over all 10 arrival schedules) number of acceptances and rejections and the average number of users per unit time. The results are as follows:

	Acceptances	Rejections	Average # of Users
Network 1	135.9	491.7	3.855
Network 2	123.5	504.1	3.714
Network 3	115.8	511.8	2.784

This divergence in acceptances can be seen for the average of 100 test runs in figure 5.4.

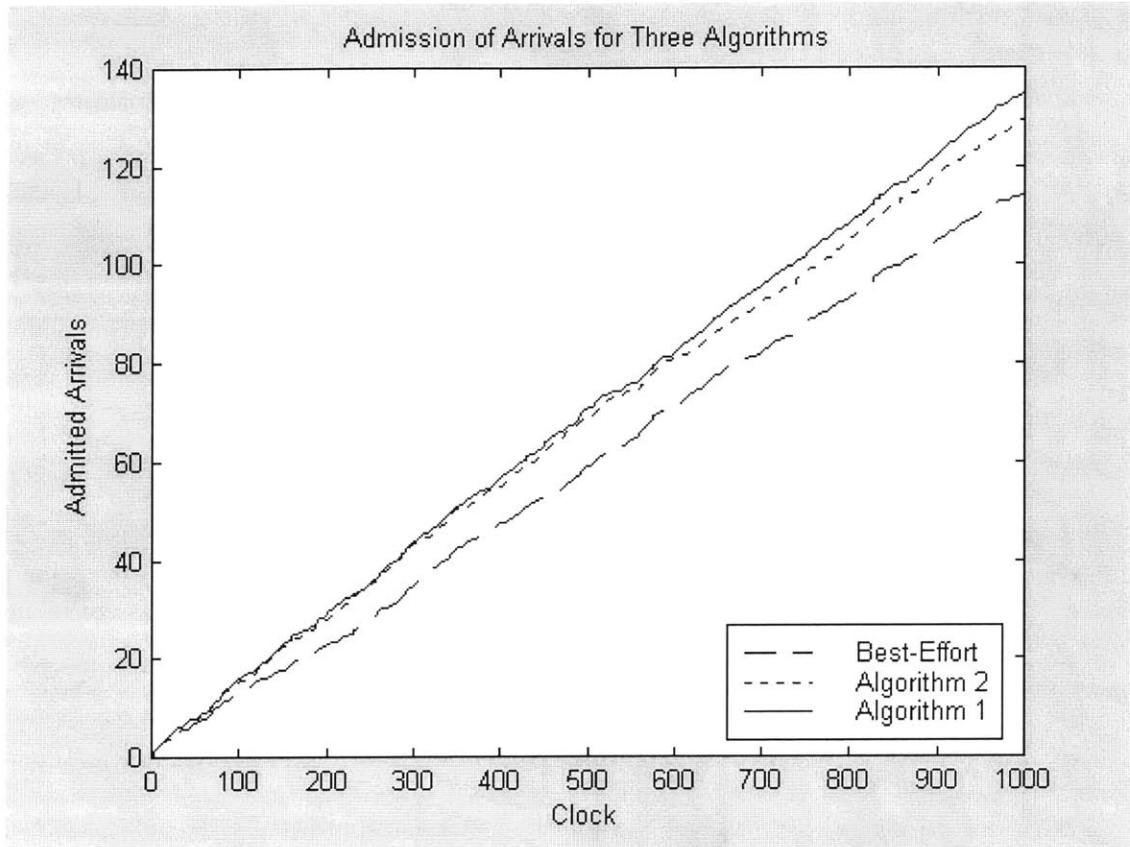


Figure 5.4: Acceptances Over 100 Test Runs

Finally, we performed simulations with a new set of 10 arrival schedules for 1000 time units to measure total system bandwidth utilization on a per time unit basis:

	Bandwidth Utilization
Network 1	77.43%
Network 2	77.01%
Network 3	72.96%

5.3 Analysis and Discussion

We see from our analysis in Chapter 4 and the results presented in the previous section that the admission algorithms we have designed operate as expected. However each algorithm has its advantages and disadvantages depending on expected use. As noted earlier, Algorithm 1 attempts to optimize total utility and therefore does not view degradation of a single user as a bad property. Algorithm 2, on the other hand, attempts to make equitable re-distributions across all users, and therefore prevents a single or a few users from realizing drastic changes in their utility. Algorithm 1 has the advantage of performing re-distributions on only a few users, or as many are needed to create the required bandwidth. Algorithm 2 however, performs re-distributions on all users.

As an alternative to reducing all users simultaneously, we introduce two methods of reducing the re-distributable user space (i.e. the set of users that are available for re-distribution). Specifically, we first remove all users from the space who are approaching the close of their connection. If users were able to specify a transmit size with their flow specification, then when users are close (by some network specified percentage) to the end of their connection, they are no longer available for reductions. This is done because these users will soon have to re-negotiate their connections, and as such, it is unfair for them to be reduced just as they are about to finish. Additionally, the re-distributable space can be further reduced by removing users from it who have been recently reduced. By tagging each user with a last reduced time-stamp, the network can specify the minimum number of cycles a reduced user can wait before it is available for reduction again. The network creates a subset of users that are least recently reduced. Thus, no users are consistently reduced upon each execution of the algorithms.

We now consider the results of the acceptances and rejections simulations. Between the best effort algorithm and Network 2, we saw a 6.6% increase in the number of acceptances. This is an expected result because Algorithm 2 attempts to make re-distributions so users that the best effort algorithm would immediately reject, can potentially be accommodated. The increase in acceptances is not overwhelmingly large because opportunities for re-distributions to occur (i.e. the requesting client can join the network if re-distributions occur) is not a frequent event. Between the best effort algorithm and Network 1, we saw a 17.4% increase in the number of acceptances, which consequently lead to a 38.4% increase in the average number of clients in the network, per unit time. This is a significant increase over the best effort algorithm because Algorithm 1 uses a simple technique to accommodate more users. While Algorithm 2 will only make re-distributions if they can be done fairly, Algorithm 1 will make them as long as the bandwidth can be created from the existing network. As such, Algorithm 1 will run successfully every time a potential user can join if re-distributions occur.

Finally, as expected, there were not significant increases in system utilization using the new algorithms. Both offered approximately a 4% increase in average network bandwidth utilization. All three networks attempt to give each user as much bandwidth as reasonably possible, thus utilization is hardly effected with the introduction of re-distributions. The primary goal of our re-distribution algorithms is to accommodate as many new users as possible. The increase in utilization can be attributed to the higher number of acceptances, and thus more users existing in the system at any given time.

CHAPTER 6

CONCLUSION

Our goal with this research was to address the issue of resource allocation in the bandwidth-constrained environment. We focus on the provisioning of resources to adaptive multimedia applications that can change coding schemes based on available resources. We explore the issues behind designing a general call-level QoS system that reserves paid-for resources for these applications. Our primary contribution is in the algorithms for network admission control. Here we introduce a dynamic resource negotiation scheme that not only allows for adaptive flows during traffic execution, but also for QoS renegotiations with existing flows at the point of admission decision for a new flow. These algorithms have been tested and shown to produce increases in call accept / reject ratios as well as system utilization. They are based on a linear utilization model, however we feel they can be easily modified for non-linear utility models. Additionally we have contributed recommendations towards a QoS architecture using the resource parameters we specified.

For future work we see the need for proper integration of negotiating schemes such as ours into an underlying QoS and signaling system. Additionally, a measurement based renegotiation algorithm based on dynamic tradeoffs between bandwidth, error rates and delay could allow for increased system utilization. Such tradeoffs, in conjunction with the methods of fair re-distribution presented in this thesis, may help maximize the number of users in a network.

BIBLIOGRAPHY

- [ACH98] C. Aurrecoechea, A.T. Campbell, and L. Hauw. A Survey of QoS Architectures. ACM/Springer Verlag Multimedia Systems Journal, Special Issue on QoS Architecture, Vol. 6 No. 3, May 1998.
- [ADA96] A. Adas. Supporting Real Time VBR Using Dynamic Reservation Based on Linear Prediction. Proceedings of IEEE INFOCOM '96, Vol. 3, San Francisco, CA, March 1996.
- [BDDM93] R. Bolla, F. Danovaro, F. Davoli, and M. Marchese. An Integrated Dynamic Resource Allocation Scheme for ATM Networks. Proceedings of IEEE INFOCOM '93, San Francisco, CA, 1993.
- [BFT99] J. Bolot, S. Fosse-Parisis, and D. Towsley. Adaptive FEC-Based Error Control for Interactive Audio in the Internet. Proceedings of IEEE INFOCOM, March 1999.
- [BLU99] Bluetooth SIG. Specification of the Bluetooth System - Core. V1.0B December 1999.
- [BS] Y. Bao and A. Sethi. Predictive Control of Delay and Packet Loss Network QoS for Multimedia Applications. Department of Computer and Information Sciences, University of Delaware, Newark, DE.
- [BT98] J. Bolot and T. Turletti. Experience With Rate Control Mechanisms for Packet Video in the Internet. Computer Communications Review, Vol. 29, No. 1, 1998.

- [CKT89] C. Casetti, J. Kurose, and D. Towsley. A New Algorithm for Measurement-Based Admission Control in Integrated Services Packet Networks. Proceedings of Protocols for High Speed Networks '96, Sophia Antipolis, France, October 1996.
- [FH98] P. Ferguson and G. Huston. Quality of Service: Delivering QoS on the Internet and in Corporate Networks. John Wiley & Sons, January 1998.
- [FR97] E. Fulp and D. Reeves. Dynamic Bandwidth Allocation Techniques. Technical Report Center for Advanced Computing and Communication, 1997.
- [GVSS96] P.Goyal, H. Vin, C. Shen, and P. Shenoy. A Reliable, Adaptive Network Protocol for Video Transport. Proceedings of INFOCOM '96, San Francisco, CA, March 1996.
- [LB99] Y. Lu and R. Brodersen. Integrating Power Control, Error Correction Coding, and Scheduling for a CDMA Downlink System. IEEE Journal on Selected Areas in Communications, Vol. 17, No. 5, June 1999.
- [LZC00] S. Lee, G. Ahn, X. Zhang, and A. Campbell. INSIGNIA: An IP-Based Quality of Service Framework for Mobile ad Hoc Networks. Journal of Parallel and Distributed Computing, 60, 2000.
- [PTW88] S.S. Panwar, D. Towsley, and J.K. Wolf. Optimal Scheduling Policies for a Class of Queues with Customer Deadlines to the Beginning of Services. Journal of the ACM, Vol. 35, No. 4, 1988.
- [ROMW] D. Reininger, M. Ott, G. Michelitsch, and G. Welling. Dynamic Bandwidth Allocation for Distributed Multimedia with Adaptive QoS. <ftp://lrcftp.epfl.ch/pub/people/ferrari/djr.ps.gz>.

- [SET98] P. Setthawong. A Fair Control Mechanism with QoS Guarantee Support for Dual Ring LANs/MANs. Master Thesis, University of Tokyo, February 1998.
- [VAN00] B. Vandalore. Traffic Management to Enhance Quality of Service of Multimedia over Available Bit Rate Service in Asynchronous Transfer Mode Networks. PhD Thesis, Ohio State University, 2000.
- [YAN00] Yankee Group. Fighting for Air: The Wireless Home Network Technology Wars. Yankee Group Report, Consumer Market Convergence, Vol 17, No. 2, February 2000.
- [ZK95] H. Zhang and E. Knightly. A New Approach to Support Delay-Sensitive VBR Video in Packet-Switched Networks. Proceedings of 5th International Workshop on Network and Operating System Support for Digital Audio and Video. Durham, NH, April 1995.

APPENDIX A SIMULATION CODE

A.1 Best Effort Network

```
% Bluetooth Client Utility Simulation
% Gaurav Tuli (gtuli@mit.edu)
% Massachusetts Institute of Technology
% Semiconductor Products Sector, Motorola

%clear

fprintf('\n\n*****\n')
fprintf('Bluetooth Client Utility Simulation\n\n');
fprintf('Gaurav Tuli (gtuli@mit.edu)\n');
fprintf('Massachusetts Institute of Technology\n');
fprintf('Semiconductor Products Sector, Motorola\n');
fprintf('*****\n\n')

% initialize state variables
max_members = 8;
clock = 1;
n_users = 0;

run_time = input('Run simulation for t = ? ticks -> ');
take_t = input('Use t ? -> ');

% these represent values at end of last clock tick

bw_utility = zeros(max_members, run_time);
bw_allocated = zeros(max_members, run_time);
admitted = zeros(max_members, 6, run_time); % min,max,b_wt,t_in,t,uid
bw_utilization_sum = 0;

bw_total = 750.0;
bw_avail = 750.0;
max_bw_avail = 750.0;
time_cutoff = 2/3;

accepts = 0;
rejects = 0;
arrivals = 0;

departure = 0;
arrival = 0;
next_arrival_time = 1;
uid = 0; % user ID for tracking

fprintf('-> max_members = %d\n', max_members);
fprintf('-> n_users = %d\n', n_users);
fprintf('-> total_bw = %4.3f kbps\n', bw_total);
```

```

fprintf('-> BT clock = %d * 625 * 10^-6 s\n\n', clock);
fprintf('-> BEGIN\n');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CREATE ARRIVAL
ARRAY%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

arrival_schedule = zeros(run_time, 4);

while (next_arrival_time < run_time)

    r_bw_min = (rand * 700) + 15;
    r_max_percent = (rand * 20) + 10; % 10 - 30% window
    r_bw_max = r_bw_min * (1 + .01 * r_max_percent);
    r_time = round((rand * 45) + 5); % 5s - 50s connection
    r_bw_wt = rand;

    if r_bw_wt < .5
        r_bw_wt = .333;
    else
        r_bw_wt = .666;
    end % if r_bw

        arrival_schedule(next_arrival_time, :) = [r_bw_min, r_bw_max, r_bw_wt, r_time];

    next_arrival_time = next_arrival_time + round(-log(rand));
end % while (next_arrival...

if (take_t == 1) arrival_schedule = t;
end %if (ta

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for clock = 2 : run_time

    fprintf('\n-> clock = %d', clock)

    % copy over previous admitted clock tick

    admitted(:, :, clock) = admitted(:, :, clock - 1);
    bw_allocated(:, clock) = bw_allocated(:, clock - 1);

    % recompute available resources - max_bw_avail, bw_avail, etc.

    bw_avail = bw_total - sum(bw_allocated(:, clock));
    max_bw_avail = bw_total - sum(admitted(:, 1, clock));

```

```

%%%%%%%%% add to average system utility
bw_utilization_sum = bw_utilization_sum + (sum(bw_allocated(:, clock)) / bw_total);

% compute bw_utility
for user = 1:max_members

    if (admitted(user, 6, clock) ~= 0)

        bw_utility(user, clock) = bw_allocated(user, clock) - admitted(user, 1, clock);
        bw_utility(user, clock) = bw_utility(user, clock) ./ (admitted(user, 2, clock) -
admitted(user, 1, clock));

    end % end if (admit...)

end % end for user...

% reset arrival / departure state

departure = 0;
arrival = 0;

%%%%%%%%%
%%%%%%%%%

% IS there a departure on this clock tick?

%%%%%%%%%
%%%%%%%%%

% for simplicity, if there is a departure, you can have no arrivals
% on that clock tick

for counter = 1 : max_members

    t_out = admitted(counter, 4, clock) + admitted(counter, 5, clock);

    if (t_out == clock)
%%%%%%%%%        fprintf('\n\t-> uid = %d departed', admitted(counter, 6, clock))

        % REMOVE USER

        % set global variables for departed member
        departure = 1;
        n_users = n_users - 1;

        % zero out admitted, utility, allocated vectors
        admitted(counter, 1:6, clock) = 0;
        bw_allocated(counter, clock) = 0;
        bw_utility(counter, clock) = 0;

    end        % (t_out == clock)

```

```

end % end counter for loop

% recompute available resources - max_bw_avail, bw_avail, etc.

bw_avail = bw_total - sum(bw_allocated(:, clock));
max_bw_avail = bw_total - sum(admitted(:, 1, clock));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% IS there an arrival on this clock tick
% (enforcing no departure)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% if (departure == 0) % then no departures occurred

if (arrival_schedule(clock, 1) ~= 0) %arrival occured within this next clock tick

%%          fprintf('\n\t-> We have an arrival')
arrival = 1;
arrivals=arrivals+1;
% compute resource requests

r_bw_min = arrival_schedule(clock, 1);
r_bw_max = arrival_schedule(clock, 2);
r_bw_wt = arrival_schedule(clock, 3);
r_time = arrival_schedule(clock, 4);

%%          fprintf('\n\t Arrival req: %f %f, weight: %f, r_bw_min, r_bw_max, r_bw_wt);
%%          fprintf('\n\t Avail: min - %f max - %f, bw_avail, max_bw_avail);

% compute next arrival time
next_arrival_time = round(-log(rand));

end % if (arrival_schedule...

% end % end (departure == 0)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% If there was an arrival, can we admit it?

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if (arrival == 1)

if (n_users == max_members)

%%          fprintf('\n\t-> ** User rejected, max_members reached **');
rejects = rejects + 1;

```

```

elseif (bw_avail > r_bw_min)
    % automatically can admit user at max available for him

    n_users = n_users + 1;
    uid = uid + 1;

    % find first empty slot in storage

    counter = 1;
    while (bw_allocated(counter, clock) > 0)
        counter = counter + 1;
    end % while(bw_...)

    % set bw to either his max request, or as much as is avail
    % which is necessarily greater than his min

    bw_allocated(counter, clock) = min(r_bw_max, bw_avail);
    admitted(counter, 1, clock) = r_bw_min;
    admitted(counter, 2, clock) = r_bw_max;
    admitted(counter, 3, clock) = r_bw_wt;
    admitted(counter, 4, clock) = clock;
    admitted(counter, 5, clock) = r_time;
    admitted(counter, 6, clock) = uid;

%%%      fprintf('\n\t-> ** User admitted with %f **', bw_allocated(counter, clock));

        accepts = accepts + 1;
    else
%%%      fprintf('\n\t-> ** Preferences of user can not be realized **');
        rejects = rejects + 1;
    end

    end % end (arrival == 1)

end % end clock for loop

fprintf('\n\n-> * accepts: %d, rejects %d\n', accepts, rejects);

t = arrival_schedule;

```

A.2 Algorithm 2 Network

```

% Bluetooth Client Utility Simulation
% Gaurav Tuli (gtuli@mit.edu)
% Massachusetts Institute of Technology
% Semiconductor Products Sector, Motorola

```

```

fprintf('\n\n*****\n')
fprintf('Bluetooth Client Utility Simulation\n\n');

```

```

fprintf('Gaurav Tuli (gtuli@mit.edu)\n');
fprintf('Massachusetts Institute of Technology\n');
fprintf('Semiconductor Products Sector, Motorola\n');
fprintf('*****\n\n')

% initialize state variables
max_members = 8;
clock = 1;
n_users = 0;

%run_time = input('Run simulation for t = ? ticks -> ');
%take_t = input('Use t ? -> ');

% these represent values at end of last clock tick

bw_utility = zeros(max_members, run_time);
bw_allocated = zeros(max_members, run_time);
admitted = zeros(max_members, 6, run_time); % min,max,b_wt,t_in,t,uid
bw_utilization_sum = 0;

bw_total = 750.0;
bw_avail = 750.0;
max_bw_avail = 750.0;
time_cutoff = 2/3;

accepts = 0;
rejects = 0;

departure = 0;
arrival = 0;
arrivals = 0;
next_arrival_time = 1;
uid = 0; % user ID for tracking

fprintf('-> max_members = %d\n', max_members);
fprintf('-> n_users = %d\n', n_users);
fprintf('-> total_bw = %4.3f kbps\n', bw_total);
fprintf('-> BT clock = %d * 625 * 10^-6 s\n\n', clock);
fprintf('-> BEGIN\n');

%%%%%%%%%%
%%%%%%%%%%

%%%%%%%%%%
%%%%%%%%%%
% CREATE ARRIVAL
ARRAY%%%%%%%%%
%%%%%%%%%
%%%%%%%%%
%%%%%%%%%

arrival_schedule = zeros(run_time, 4);

while (next_arrival_time < run_time)

```



```

r_bw_min = (rand * 700) + 15;
r_max_percent = (rand * 20) + 10; % 10 - 30% window
r_bw_max = r_bw_min * (1 + .01 * r_max_percent);
r_time = round((rand * 45) + 5); % 5s - 50s connection
r_bw_wt = rand;

if r_bw_wt < .5
    r_bw_wt = .333;
else
    r_bw_wt = .666;
end % if r_bw

arrival_schedule(next_arrival_time, :) = [r_bw_min, r_bw_max, r_bw_wt, r_time];

next_arrival_time = next_arrival_time + round(-log(rand));
end % while (next_arrival...

if (take_t == 1) arrival_schedule = t;
end % if (tak

%%%%%%%%%%

for clock = 2 : run_time

%%%    fprintf('\n-> clock = %d', clock)

    % copy over previous admitted clock tick

    admitted(:, :, clock) = admitted(:, :, clock - 1);
    bw_allocated(:, clock) = bw_allocated(:, clock - 1);

    % recompute available resources - max_bw_avail, bw_avail, etc.

    bw_avail = bw_total - sum(bw_allocated(:, clock));
    max_bw_avail = bw_total - sum(admitted(:, 1, clock));

    %%%%%%%%% add to average system utility
    bw_utilization_sum = bw_utilization_sum + (sum(bw_allocated(:, clock)) / bw_total);

    % compute bw_utility
    for user = 1:max_members

        if (admitted(user, 6, clock) ~= 0)

            bw_utility(user, clock) = bw_allocated(user, clock) - admitted(user, 1, clock);
            bw_utility(user, clock) = bw_utility(user, clock) ./ (admitted(user, 2, clock) -
admitted(user, 1, clock));

        end % end if (admit...)

    end % end for user...

```

```

% reset arrival / departure state

departure = 0;
arrival = 0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% IS there a departure on this clock tick?

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% for simplicity, if there is a departure, you can have no arrivals
% on that clock tick

for counter = 1 : max_members

    t_out = admitted(counter, 4, clock) + admitted(counter, 5, clock);

    if (t_out == clock)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%      fprintf("\n\t-> uid = %d departed', admitted(counter, 6, clock))

        % REMOVE USER

        % set global variables for departed member
        departure = 1;
        n_users = n_users - 1;

        % zero out admitted, utility, allocated vectors
        admitted(counter, 1:6, clock) = 0;
        bw_allocated(counter, clock) = 0;
        bw_utility(counter, clock) = 0;

    end      % (t_out == clock)

end % end counter for loop

% recompute available resources - max_bw_avail, bw_avail, etc.

bw_avail = bw_total - sum(bw_allocated(:, clock));
max_bw_avail = bw_total - sum(admitted(:, 1, clock));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% IS there an arrival on this clock tick
% (enforcing no departure)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

% if (departure == 0) % then no departures occurred

    if (arrival_schedule(clock, 1) ~= 0) %arrival occured within this next clock tick

%%%         fprintf('\n\t-> We have an arrival')
            arrival = 1;
arrivals = arrivals + 1;
            % compute resource requests

            r_bw_min = arrival_schedule(clock, 1);
            r_bw_max = arrival_schedule(clock, 2);
            r_bw_wt = arrival_schedule(clock, 3);
            r_time = arrival_schedule(clock, 4);

%%%         fprintf('\n\t Arrival req: %f %f, weight: %f, r_bw_min, r_bw_max, r_bw_wt);
%%%         fprintf('\n\t Avail: min - %f max - %f, bw_avail, max_bw_avail);

            % compute next arrival time
            next_arrival_time = round(-log(rand));

            end % if (arrival_schedule...

% end % end (departure == 0)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% If there was an arrival, can we admit it?

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if (arrival == 1)

    if (n_users == max_members)

%%%         fprintf('\n\t-> ** User rejected, max_members reached **');
            rejects = rejects + 1;

    elseif (bw_avail > r_bw_min)
        % automatically can admit user at max available for him

            n_users = n_users + 1;
            uid = uid + 1;

            % find first empty slot in storage

            counter = 1;
            while (bw_allocated(counter, clock) > 0)
                counter = counter + 1;
            end % while(bw_...)

            % set bw to either his max request, or as much as is avail
            % which is necessarily greater than his min

```

```

    bw_allocated(counter, clock) = min(r_bw_max, bw_avail);
    admitted(counter, 1, clock) = r_bw_min;
    admitted(counter, 2, clock) = r_bw_max;
    admitted(counter, 3, clock) = r_bw_wt;
    admitted(counter, 4, clock) = clock;
    admitted(counter, 5, clock) = r_time;
    admitted(counter, 6, clock) = uid;

%%%      fprintf('\n\t-> ** User admitted with %f **', bw_allocated(counter, clock));

                                accepts = accepts + 1;
    elseif (max_bw_avail > r_bw_min)
        % must reduce other clients bandwidths to accomodate user

        bwNeeded = r_bw_min - bw_avail;
        bwFound = 0;
        bwUtilitySum = sum(bw_utility(:,clock));
    flag = 0;

%%%      fprintf('\n\t-> bwFound = %f ; bwNeeded = %f, bwFound, bwNeeded);

    % determine what M1 should be
    counter = 1;
    while (bw_allocated(counter, clock) == 0)
        counter = counter + 1;
    end % while(bw_...)

                                M1 = admitted(counter, 3, clock) / (admitted(counter, 2, clock) -
admitted(counter, 1, clock));
    J = 1 / M1;

    counter = counter + 1;
    while (counter < max_members)
        if (bw_allocated(counter, clock) > 0)
            Mk = admitted(counter, 3, clock) / (admitted(counter, 2, clock) - admitted(counter, 1, clock));
            J = J + 1/Mk;
        end % if(bw_al...
    counter = counter + 1 ;
    end % while (counter...

    J = J * M1;

    reduce(1:counter) = 0;
    counter = 1;
    while (counter < max_members)
        if (bw_allocated(counter, clock) > 0)
            Mk = admitted(counter, 3, clock) / (admitted(counter, 2, clock) - admitted(counter, 1, clock));
            reduce(counter) = M1 * bwNeeded / (Mk * J);
            if (reduce(counter) > (bw_allocated(counter, clock) - admitted(counter, 1, clock)))
                flag = 1;
            end % if (redu...
            bwFound = bwFound + reduce(counter);
            bw_allocated(counter, clock) = bw_allocated(counter, clock) - reduce(counter);
        end % if (bw_allo

```

```

        counter = counter + 1;
end % while (counter <...

if (flag) %then undo
%%%      fprintf('\n\t-----User could not be accomodated fairly by redistribution-----');
counter = 1;
while (counter < max_members)
    bw_allocated(counter, clock) = bw_allocated(counter,clock) + reduce(counter);
    counter = counter + 1;
end % while (cou
rejects = rejects + 1;
else
%%%%%% we able to do it

n_users = n_users + 1;
uid = uid + 1;

% find first empty slot in storage

counter = 1;
while (bw_allocated(counter, clock) > 0)
    counter = counter + 1;
end % while(bw_...)

% set bw to his min request

bw_allocated(counter, clock) = r_bw_min;
admitted(counter, 1, clock) = r_bw_min;
admitted(counter, 2, clock) = r_bw_max;
admitted(counter, 3, clock) = r_bw_wt;
admitted(counter, 4, clock) = clock;
admitted(counter, 5, clock) = r_time;
admitted(counter, 6, clock) = uid;

%%%      fprintf('\n\t-> ----- User admitted with redistribution with %f -----',
bw_allocated(counter, clock));
    accepts = accepts + 1;
end % else

else
%%%      fprintf('\n\t-> ** Preferences of user can not be realized **');
    rejects = rejects + 1;
end

end % end (arrival == 1)

end % end clock for loop

fprintf('\n\n-> * accepts: %d, rejects %d\n', accepts, rejects);

```

A.3 Algorithm 1 Network

% Bluetooth Client Utility Simulation

```

% Gaurav Tuli (gtuli@mit.edu)
% Massachusetts Institute of Technology
% Semiconductor Products Sector, Motorola

%clear

fprintf('\n\n*****\n\n')
fprintf('Bluetooth Client Utility Simulation\n\n');
fprintf('Gaurav Tuli (gtuli@mit.edu)\n');
fprintf('Massachusetts Institute of Technology\n');
fprintf('Semiconductor Products Sector, Motorola\n');
fprintf('*****\n\n')

% initialize state variables
max_members = 8;
clock = 1;
n_users = 0;

%run_time = input('Run simulation for t = ? ticks -> ');
%take_t = input('Use t ? -> ');

% these represent values at end of last clock tick

bw_utility = zeros(max_members, run_time);
bw_allocated = zeros(max_members, run_time);
admitted = zeros(max_members, 6, run_time); % min,max,b_wt,t_in,t,uid
bw_utilization_sum = 0;

bw_total = 750.0;
bw_avail = 750.0;
max_bw_avail = 750.0;
time_cutoff = 2/3;

accepts = 0;
rejects = 0;

departure = 0;
arrival = 0;
next_arrival_time = 1;
uid = 0; % user ID for tracking

fprintf('-> max_members = %d\n', max_members);
fprintf('-> n_users = %d\n', n_users);
fprintf('-> total_bw = %4.3f kbps\n', bw_total);
fprintf('-> BT clock = %d * 625 * 10^-6 s\n\n', clock);
fprintf('-> BEGIN\n');

%%%%%%%%%%
%%%%%%%%%%

%%%%%%%%%%
%%%%%%%%%%

```

```

% CREATE ARRIVAL
ARRAY%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

arrival_schedule = zeros(run_time, 4);

while (next_arrival_time < run_time)

    r_bw_min = (rand * 700) + 15;
    r_max_percent = (rand * 20) + 10; % 10 - 30% window
    r_bw_max = r_bw_min * (1 + .01 * r_max_percent);
    r_time = round((rand * 45) + 5); % 5s - 50s connection
    r_bw_wt = rand;

    if r_bw_wt < .5
        r_bw_wt = .333;
    else
        r_bw_wt = .666;
    end % if r_bw

        arrival_schedule(next_arrival_time, :) = [r_bw_min, r_bw_max, r_bw_wt, r_time];

    next_arrival_time = next_arrival_time + round(-log(rand));
end % while (next_arrival...

if (take_t == 1) arrival_schedule = t;
end % if

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for clock = 2 : run_time

%%%   fprintf('\n-> clock = %d', clock)

    % copy over previous admitted clock tick

    admitted(:, :, clock) = admitted(:, :, clock - 1);
    bw_allocated(:, clock) = bw_allocated(:, clock - 1);

    % recompute available resources - max_bw_avail, bw_avail, etc.

    bw_avail = bw_total - sum(bw_allocated(:, clock));
    max_bw_avail = bw_total - sum(admitted(:, 1, clock));

    % add to average system utility
    bw_utilization_sum = bw_utilization_sum + (sum(bw_allocated(:, clock)) / bw_total);

    % compute bw_utility
    for user = 1:max_members

```

```

        if (admitted(user, 6, clock) ~= 0)

            bw_utility(user, clock) = bw_allocated(user, clock) - admitted(user, 1, clock);
            bw_utility(user, clock) = bw_utility(user, clock) ./ (admitted(user, 2, clock) -
admitted(user, 1, clock));

        end % end if (admit...)

        end % end for user...

% reset arrival / departure state

departure = 0;
arrival = 0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% IS there a departure on this clock tick?

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% for simplicity, if there is a departure, you can have no arrivals
% on that clock tick

for counter = 1 : max_members

    t_out = admitted(counter, 4, clock) + admitted(counter, 5, clock);

    if (t_out == clock)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        fprintf('\n\t-> uid = %d departed', admitted(counter, 6, clock))

        % REMOVE USER

        % set global variables for departed member
        departure = 1;
        n_users = n_users - 1;

        % zero out admitted, utility, allocated vectors
        admitted(counter, 1:6, clock) = 0;
        bw_allocated(counter, clock) = 0;
        bw_utility(counter, clock) = 0;

    end % (t_out == clock)

end % end counter for loop

% recompute available resources - max_bw_avail, bw_avail, etc.

bw_avail = bw_total - sum(bw_allocated(:, clock));
max_bw_avail = bw_total - sum(admitted(:, 1, clock));

```



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% IS there an arrival on this clock tick
% (enforcing no departure)
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% if (departure == 0) % then no departures occurred
```

```
if (arrival_schedule(clock, 1) ~= 0) %arrival occured within this next clock tick
```

```
%%%          fprintf('\n\t-> We have an arrival')
arrival = 1;
```

```
    % compute resource requests
```

```
    r_bw_min = arrival_schedule(clock, 1);
    r_bw_max = arrival_schedule(clock, 2);
    r_bw_wt = arrival_schedule(clock, 3);
    r_time = arrival_schedule(clock, 4);
```

```
%%%          fprintf('\n\t Arrival req: %f %f, weight: %f', r_bw_min, r_bw_max, r_bw_wt);
%%%          fprintf('\n\t Avail: min - %f max - %f, bw_avail, max_bw_avail);
```

```
    % compute next arrival time
    next_arrival_time = round(-log(rand));
```

```
end % if (arrival_schedule...
```

```
% end % end (departure == 0)
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% If there was an arrival, can we admit it?
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
if (arrival == 1)
```

```
    if (n_users == max_members)
```

```
%%%          fprintf('\n\t-> ** User rejected, max_members reached **');
rejects = rejects + 1;
```

```
    elseif (bw_avail > r_bw_min)
        % automatically can admit user at max available for him
```

```
        n_users = n_users + 1;
        uid = uid + 1;
```

```
        % find first empty slot in storage
```

```

counter = 1;
while (bw_allocated(counter, clock) > 0)
    counter = counter + 1;
end % while(bw_...)

% set bw to either his max request, or as much as is avail
% which is necessarily greater than his min

bw_allocated(counter, clock) = min(r_bw_max, bw_avail);
admitted(counter, 1, clock) = r_bw_min;
admitted(counter, 2, clock) = r_bw_max;
admitted(counter, 3, clock) = r_bw_wt;
admitted(counter, 4, clock) = clock;
admitted(counter, 5, clock) = r_time;
admitted(counter, 6, clock) = uid;

%% fprintf('\n\t-> ** User admitted with %f **', bw_allocated(counter, clock));
                                accepts = accepts + 1;

elseif (max_bw_avail > r_bw_min)
    % must reduce other clients bandwidths to accomodate user

    bwNeeded = r_bw_min - bw_avail;
    bwFound = 0;
    bwUtilitySum = sum(bw_utility(:,clock));

%% fprintf('\n\t-> bwFound = %f ; bwNeeded = %f', bwFound, bwNeeded);

    while (bwFound < bwNeeded)

        % find lowest marginal utility

        counter = 1; min_marg_util = inf; user_num = 0;
        while (counter <= max_members)
            window = admitted(counter, 2, clock) - admitted(counter, 1, clock);
            if (bw_allocated(counter, clock) > admitted(counter, 1, clock))
                if ((admitted(counter, 3, clock) / window) < min_marg_util)
                    min_marg_util = admitted(counter, 3, clock) / window;
                    user_num = counter;
                end % if ((admi
            end % if (window ~=
        counter = counter + 1;
    end % while (counter

    % we have user with minimum marginal utility, so take as much as possible
    % from that user

    reduce = min(bw_allocated(user_num, clock) - admitted(user_num, 1, clock), bwNeeded -
bwFound);
    bw_allocated(user_num, clock) = bw_allocated(user_num, clock) - reduce;
    bwFound = bwFound + reduce;

    fprintf('\n\t-> user_num %d is being reduced by %f', user_num, reduce);

```

```

%         disp(bwFound); disp (bwNeeded);

                end % end while (bwFound...)

n_users = n_users + 1;
uid = uid + 1;

% find first empty slot in storage

counter = 1;
while (bw_allocated(counter, clock) > 0)
    counter = counter + 1;
end % while(bw_...)

% set bw to his min request

bw_allocated(counter, clock) = r_bw_min;
admitted(counter, 1, clock) = r_bw_min;
admitted(counter, 2, clock) = r_bw_max;
admitted(counter, 3, clock) = r_bw_wt;
admitted(counter, 4, clock) = clock;
admitted(counter, 5, clock) = r_time;
admitted(counter, 6, clock) = uid;

fprintf('\n\t-> !!!!!!! User admitted with redistribution with %f !!!!!!!', bw_allocated(counter,
clock));
                                accepts = accepts + 1;
else
%%%%      fprintf('\n\t-> ** Preferences of user can not be realized **');
rejects = rejects + 1;
end

end % end (arrival == 1)

end % end clock for loop

fprintf('\n\n-> * accepts: %d, rejects %d\n', accepts, rejects);

```