A CHANNEL ALLOCATION SCHEME FOR MULTIMEDIA ON-DEMAND SYSTEMS

by

Jianbin Ding

B.Eng., Shanghai University of Technology, Shanghai, China, 1988

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF APPLIED SCIENCE

in the School
of
Computing Science

© Jianbin Ding 1994
SIMON FRASER UNIVERSITY
February 1994

All rights reserved. This work may not be reproduced in whole or in part, by photocopy or other means, without the permission of the author.
APPROVAL

Name: Jianbin Ding
Degree: Master of Applied Science
Title of thesis: A Channel Allocation Scheme for Multimedia On-demand Systems

Examining Committee: Dr. Peter Triantafillou, Chairman

__________________________
Dr. Tiko Kameda
Senior Supervisor

__________________________
Dr. Dave Fracchia
Supervisor

__________________________
Dr. Lou Iafer
Supervisor

__________________________
Dr. Stella Atkins
External Examiner

Date Approved: February 22, 1994
PARTIAL COPYRIGHT LICENSE

I hereby grant to Simon Fraser University the right to lend my thesis, project or extended essay (the title of which is shown below) to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users. I further agree that permission for multiple copying of this work for scholarly purposes may be granted by me or the Dean of Graduate Studies. It is understood that copying or publication of this work for financial gain shall not be allowed without my written permission.

Title of Thesis/Project/Extended Essay

A Channel Allocation Scheme for Multimedia On-demand Systems.

Author: Jianbin Ding

(date)

February 24, 1994
ABSTRACT

Rapid advances in high speed networks have fueled the development of multimedia on-demand services. However, because of the high bandwidth and low delay requirements of such services, conventional networks cannot provide a guaranteed quality-of-service (QOS). The thesis proposes a reservation-based channel allocation scheme and an associated control mechanism that provide the requested QOS.

In general, there are two important QOS requirements for multimedia on-demand applications: deadlines for frame delivery and initial wait time. In our model, we consider observing deadlines as a necessity and initial wait time as of only secondary importance. Given this requirement and the available buffer space at the destination, we study the buffer underflow and overflow problems for both fixed-size and variable-size media, which enables us to calculate the minimum and maximum transfer rates. Based on this range, our channel allocation scheme reserves bandwidth in the form of uniformly distributed time slots. The resulting system satisfies both bandwidth and real-time requirements of multimedia on-demand applications, and also simplifies the channel establishing procedure. The associated node controller, transmission scheduler, and other related problems are also discussed.
ACKNOWLEDGMENTS

First and foremost, I would like to express my appreciation and gratitude to my senior supervisor, Professor Tiko Kameda, for his advice and patience. His support and encouragement helped me make this thesis a reality. Dr. Kameda has given me a great help in almost every aspect of this work – from the technical contents to the text processing of this thesis.

I would also like to thank Dr. Dave Fracchia for his assistance and encouragement throughout the whole work. Dr. Fracchia has spent his valuable time helping me to correct many language problems and his great personality made this work much more interesting. Thanks are to Dr. Lou Hafer for his help from the first minute after I came to this country and to Dr. Stella Atkins for being the external examiner. Thanks are also to Mrs. Elma Krbavac for her help and encouragement, and Mr. Pimplapure and Ms. Teo for proof-reading my draft.
# CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td></td>
<td>iii</td>
</tr>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td></td>
<td>iv</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td></td>
<td>viii</td>
</tr>
<tr>
<td>1</td>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1</td>
<td>Motivation</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>Background</td>
<td>5</td>
</tr>
<tr>
<td>1.3</td>
<td>Previous Work</td>
<td>7</td>
</tr>
<tr>
<td>1.4</td>
<td>Thesis Overview</td>
<td>13</td>
</tr>
<tr>
<td>2</td>
<td>The Model</td>
<td>15</td>
</tr>
<tr>
<td>2.1</td>
<td>General Architecture</td>
<td>15</td>
</tr>
<tr>
<td>2.2</td>
<td>Quality of Service (QOS)</td>
<td>21</td>
</tr>
<tr>
<td>2.2.1</td>
<td>QOS Specification</td>
<td>22</td>
</tr>
<tr>
<td>2.2.2</td>
<td>QOS for Multimedia On-demand Applications</td>
<td>23</td>
</tr>
<tr>
<td>2.2.3</td>
<td>Playback Performance</td>
<td>24</td>
</tr>
<tr>
<td>3</td>
<td>Channel Allocation</td>
<td>30</td>
</tr>
<tr>
<td>3.1</td>
<td>Transfer Rate</td>
<td>31</td>
</tr>
<tr>
<td>3.2</td>
<td>Channel Allocation Scheme</td>
<td>45</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

2.1 A sample multimedia system architecture. ........................ 17
2.2 Structure of a network node. ...................................... 20
2.3 Fixed-size media. ....................................................... 25
2.4 Variable-size media. ..................................................... 25

3.1 Calculation of minimum transfer rate, $R_{t,min}$, when $R_t < R_c$. .................. 33
3.2 Calculation of maximum transfer rate when $R_t > R_c$ (the case where \( f/R_c < (F - f)/R_t \) and \( 2 > F/f - [(F/f - 1) \cdot R_c/R_t] \cdot R_t/R_c \)) ................................. 34
3.3 Calculation of maximum transfer rate when $R_t > R_c$ (the case where \( f/R_c < (F - f)/R_t \) and \( 2 \leq F/f - [(F/f - 1) \cdot R_c/R_t] \cdot R_t/R_c \)). .................. 35
3.4 Calculation of maximum transfer rate when $R_t > R_c$ (the case where \( f/R_c > (F - f)/R_t \)). ........................................ 36
3.5 $U_{max}/f$ vs. $R_t/R_c$. (Maple was used to draw this figure.) .................. 38
3.6 $U_{max}/f$ vs. $F/f$ when $R_t/R_c > 1$. (Maple was used to draw this figure.) 40
3.7 $F/f$ vs. $R_t/R_c$. (Maple was used to draw this figure.) .......................... 42
3.8 $U_{max}$ vs. $R_t$ (in frames/sec) for PCM voice. (Maple was used to draw this figure.) ........................................ 43
3.9 Binary tree for channel allocation scheme. ........................................ 47
3.10 Channel allocation scheme. .......................................... 50
3.11 Controller and scheduler ........................................ 53
3.12 Conflicting Requests ............................................. 61

4.1 Playback of frame a followed by frame b \( (f_a \cdot R_t^f < R_t < f_b \cdot R_t^f) \) ........................................ 69
4.2 Playback of frame b followed by frame a \( (f_a \cdot R_t^f < R_t < f_b \cdot R_t^f) \) ........................................ 70
4.3 Frames b and a are replaced by frames \( b_1 \) and \( a_1 \), respectively. \( (f_a \cdot R_t^f < R_t < f_{a_1} \cdot R_t^f) \) ........................................ 71
4.4 Calculation of maximum transfer rate (Case 2: when transmission finishes after the first frame of the second part of the file finishes its playback). ........................................ 82
4.5 \( U_{max}/f_{min} \) vs. \( R_t/R_{c,min} \). \( (Maple \ \text{was used to draw this figure.}) \) ................ 84
4.6 \( U_{max} \) vs. \( R_t \). \( (Maple \ \text{was used to draw this figure.}) \) ................ 85
LIST OF TABLES

4.1 Problems for variable-size media ........................................ 73
CHAPTER 1

Introduction

With rapid advances in storage and compression techniques, an increasing number of multimedia applications are being developed. Coupled with advances in high speed networks, multimedia on-demand systems can be built which service many users simultaneously. For example, consumers could select a video program in the comfort of their home through such a system instead of going to a video rental store.

This research is motivated by the high bandwidth and low delay requirements of multimedia on-demand systems. We believe that conventional packet switching network systems cannot provide a guaranteed quality-of-service (QOS) which is required for most multimedia communications, and that a new network resource allocation and control mechanism is needed to replace conventional protocols. In this thesis, we focus on the following major issues:

1. What are the major performance requirements for multimedia on-demand applications?
CHAPTER 1. INTRODUCTION

2. Given these performance requirements, what kind of bandwidth allocation scheme should be used to provide a user with a guaranteed QoS without excessive overhead?

3. What type of control mechanism should be used to fulfill the QoS requirements?

In a conventional network, end-to-end control with a stateless network model is often used. This thesis argues that, to ensure a guaranteed performance, a network must play an active role in traffic control in multimedia on-demand systems. We propose to allocate and manage network resources based on the transfer rate, which is the rate of transmitting packets (the number of packets/sec) from the server to the client. In order to ensure the required QOS, we propose a new reservation scheme for allocating logical channels to users.

In this chapter we first present the argument that conventional networks are inadequate for multimedia communications. Then, an introduction to the basic characteristics of Broadband Integrated Services Digital Network (B-ISDN) is provided, since it is the platform for the research in this thesis. This is followed by a brief review of previous work and an outline of the rest of this thesis.

1.1 Motivation

Traditionally, voice, audio, data and video communications have been provided by different companies. Voice communications most often use public telephone lines, while cable companies deliver video and audio information by using special cable systems. Data communications are commonly provided by privately owned networks,
CHAPTER 1. INTRODUCTION

such as LANs, or leased lines. The major reason for this separation is the difference in characteristics of these media. For example, audio and video communications are real-time, so they are delay sensitive; they usually require high bandwidth but can tolerate relatively high error rates. On the other hand, data communications may be loss sensitive, but can tolerate some delay. Thus, voice and video communications usually rely on the transmission of analog signals through circuit-switched networks, while data are sent through digital packet-switched networks.

With the advances in digital communications techniques, different types of information can all be transmitted in digital form. Thus, if an integrated network, which aims to meet the different QOS requirements of different media, can be shared, then we can have better usage of network resources. Before we discuss integrated networks, the following question should be answered: why should a new kind of network be introduced for multimedia communications? In the past two decades, digital telecommunications technology has gained wide acceptance. Network protocols like X.25 and TCP/IP are used almost everywhere. However, none of these conventional networks can be used directly as an integrated network for multimedia communications.

In general, the conventional networks use either circuit-switching or packet-switching. Circuit-switching networks, in which a dedicated end-to-end path needs to be set up before any data can be sent, can be used to provide real-time performance guarantees in the current commercial environment. However, the dedicated end-to-end path cannot be shared with other users and since an application must reserve the maximum bandwidth that it intends to use, bandwidth which is not used is wasted. Although circuit-switching allows for simpler networks which can operate at high speeds, they are only attractive when the requirements of most applications are not too different.
Because of the above mentioned problems with circuit-switching networks, this thesis considers only packet-switching networks.

Current packet-switching protocols, which include the Internet protocols IP, TCP and UDP, do not need to set-up and use a dedicated end-to-end path for transmission but cannot satisfy real-time requirements. In particular, IP can provide neither a guarantee on bounded delay nor reliable or ordered packet transmission. Packets under IP can traverse different routes and may fail to arrive at their destinations. TCP, the most widely used transport protocol, uses end-to-end acknowledgement and retransmission to achieve reliability. This approach is not suitable for real-time communications because the retransmitted packets may arrive at their destination past their deadlines. In this case, these packets are useless and retransmission is only a waste of network resources. UDP, a connectionless transport protocol, can only provide unreliable transmission upon IP protocol.

Since conventional packet-switching networks are not required to support real-time communications, an adaptive scheme is often used with suitable corrective actions, such as flow control and congestion control. For example, when a congestion occurs, the size of a sliding congestion window will be decreased, which forces the associated transfer rate to be smaller and thus some packets may fail to arrive at their destination before deadlines. In order to provide a timely and reliable transmission, the conventional feedback-based adaptive scheme must be replaced by a reservation-based predictive scheme. This approach is based on the idea that if each request provides a suitable set of transfer parameters, then the network can be designed to provide services which satisfy the requests of different users, while maintaining relatively good usage of network resources.
In summary, conventional packet-switching networks cannot support real-time communications with hard deadlines, while circuit-switching networks can only provide a service which, while good enough to meet the most stringent requirements, may result in a severe waste of resources when applications require different QoS. Thus, a new kind of network is needed to support multimedia communications.

1.2 Background

This thesis focuses on a channel allocation and control mechanism, which aims to replace the network and transport layers in the Open Systems Interconnection (OSI) model [25]. In this section, we provide an introduction to the network on which our scheme will be based – a high-speed B-ISDN, using Asynchronous Transfer Mode (ATM), which is intended to replace the data link and physical layers in the conventional OSI model.

B-ISDN is a network which evolved from a telephone Integrated Digital Network (IDN). It is designed to provide a wide range of services, including voice and non-voice services [25].

In the synchronous transfer mode, packets of a connection occupy only certain predetermined time slots on a transmission link. ATM is a more flexible and attractive approach for multimedia communication than the synchronous transfer mode. The term “asynchronous” means that packets allocated to the same connection may not be transmitted regularly, since they are sent according to the actual demand instead of a predetermined pattern in time.
As in a conventional network, an ATM-based network allocates channels requested by users and deals with routing and resource allocation issues. In an ATM-based network, all information transmitted from the source host to the destination host is grouped into fixed-size packets (called "cells" in B-ISDN terminology). These packets are identified and switched by means of a label in the packet header, so the header must at least contain a **virtual circuit** identification (a logical connection which is established before any packets are sent).

ATM takes advantage of the reliability and fidelity of modern digital communications facilities to provide faster packet switching than X.25. Some expected characteristics are:

1. No link-by-link error control.
2. No link-by-link flow control.
3. End-to-end error control if needed.
4. Use of virtual circuits.
5. Switching based on table lookup; the network node looks at the header of an incoming packet to make a routing decision.

After a request for a channel is accepted, network nodes move packets from an input link to an output link. A survey of switching schemes can be found in [28]. To simplify the problem, we assume that input delay at each node in an ATM network can be ignored. Thus, we can concentrate on the output links of a node.

One thing worth mentioning is that in ISDN, separate channels are commonly used for transmitting management information, such as that for setting up a channel,
and transmitting "real" data. Usually each user uses several "B" channels for data communication and one "D" channel for the transmission of management information. A "D" channel is shared by several users. In TCP/IP or X.25, in contrast, management information and data are transmitted in the same logical channel, which makes the protocols more complex. In this thesis, we assume separate channels are used for management information and data.

1.3 Previous Work

A number of solutions have been proposed for bandwidth allocation and control of delay sensitive packet-switching networks, and retrieval and storage of multimedia information. The following are some of the most significant ones.

Fair Queue

A fair queue [9] is a simple control strategy that provides users with an equal allocation of network resources at each link. The basic idea is to transmit data from each user in turn, like the round-robin scheduling algorithm. It may not be easy to use this approach to support multimedia applications with different bandwidth requirements.

Leaky-bucket

The leaky-bucket algorithm has been proposed for accessing high-speed networks. Various versions of the leaky-bucket algorithm have been suggested. In one version [23], at every network link, "credits" for each user are generated at a constant rate, and a certain number of credits (up to the bucket size) can be saved. If a user
has credits, his packets can be transmitted immediately. Otherwise, packets can be dropped. This approach may result in "bursty" transmissions (which makes it difficult to satisfy packets’ deadline requirements) if buckets in a network node are served in FIFO order.

VirtualClock

Zhang [33] proposed a rate-based network control system. An outline of her algorithm is as follows. During the channel set-up phase, a user requests the average transfer rate, the minimum transfer rate, and the time period (called average interval) over which the average rate will be maintained. At every node along the channel, network resources will be reserved to guarantee a throughput within the acceptable range, or the request is rejected. A user can also request a desired delay, but the network cannot guarantee it.

To monitor transmission rates, every channel has a variable called the VirtualClock at every node along the channel. When the first packet on a channel arrives, the VirtualClock is set to the actual time. During an average interval, whenever a packet arrives, the VirtualClock is increased by the amount of time it would have taken for that amount of data to arrive, if the source were transmitting at its average rate. Thus, if a source transmits faster than its agreed throughput, its VirtualClock value will be larger than the actual time. If a source sends more slowly, the VirtualClock will be behind the actual time.

At the end of the average interval, the VirtualClock value is compared with the actual time. If the value of VirtualClock is larger than the actual time by more than a certain amount, the source is advised to slow down. If the source keeps on sending
packets too fast, the traffic from the source will receive a lower priority.

An auxiliary VirtualClock is also used at every node to assign timestamps to packets to schedule their transmission. When a packet arrives, the auxiliary VirtualClock is set to the larger of the actual time and its current value. In addition, at the end of the average interval, the auxiliary VirtualClock is set in the way outlined above (for the VirtualClock), and the packets are stamped with the auxiliary VirtualClock and scheduled for transmission according to their timestamps. The reason for setting the auxiliary VirtualClock to the actual time if the latter is larger is to prevent bursty flows from interfering with other flows. Otherwise, a sender can maintain a period of silence and then send a burst of data to achieve higher priority. However, it is not easy for this approach to provide a guaranteed bounded delay.

Real-time Channels

A real-time channel [12] is a simplex connection between a sender and a receiver, and has traffic characteristics and performance requirements associated with it. The traffic characteristics consist of the minimum and average interpacket intervals. The performance requirements include maximum end-to-end delays, end-to-end delay variation and maximum packet loss rate. This is appealing, since other models mentioned above cannot completely support bounded delays, which is a very important issue in real-time communications. On the other hand, because of this support, the real-time channel model suffers from large overhead during the channel establishing phase. Because of the special characteristics of the delay requirements in a multimedia on-demand system, it is possible to simplify the establishing phase.

Teleputer
A Teleputer was developed by Silicon Graphics Inc. (SGI) [7, 22]. The head-end hardware is a cluster of 14 SGI servers on an Fiber Distributed Data Interface (FDDI) ring. Two of the machines in the cluster will function as session managers and transaction processors, and the other will deliver Motion Picture Experts Group (MPEG) standard compressed video. Small video frames will be transmitted from the video server to an ATM network node. The network will terminate in set-top boxes designed by SGI. The initial prototype of a set-top box is a modified Indy workstation with a 100MHz MIPS R4000 processor and a Scientific-Atlanta add-in board for MPEG decompression and analog signal processing. Bootup and client software management will be handled at the head-end, which reduces system cost.

Gemmell's Work

Gemmell's work establishes some fundamental principles for the retrieval and storage for delay-sensitive multimedia data [13]. His theoretical framework is developed for the real-time requirements of digital audio playback. By showing how to describe these requirements in terms of the consumption rate and the data-retrieval rate, bounds are derived on buffer space for certain common retrieval scenarios. Storage placement strategies for multimedia synchronized data are also examined in his work. Gemmell's work provides us with a basic idea for calculating of the range of the transfer rate, even though his work focuses on the buffer space requirement when the transfer rate is fixed. However, his work cannot be applied to the buffer overflow and underflow problems for variable-size media, which will be discussed in Chapter 4. We will point out differences between his model and ours later at appropriate places.

UC San Diego's Multimedia On-demand System
Work at UC San Diego [24] focuses on one important issue of multimedia on-demand: synchronization among several media streams which are played back simultaneously. In their work, they assume that clients’ equipments have minimum capacity for playback but cannot run elaborate time synchronization protocols. They use a rate-based feedback strategy in which a multimedia file server uses feedback units transmitted periodically by clients’ equipments, to estimate the playback instants of frames. According to these estimates, the server detects the discontinuities of playback owing to buffer underflow or overflow and adjusts transmission to avoid either of these problems. We do not address synchronization problem. In our model, we assume that data of different media streams are stored together and have to be transmitted from the same file server at the same time (like convention video programs). [24] discusses the relationship between buffer size and transmission rate, in which the buffer size determines minimum feedback ratio, the minimum rate at which feedback units must be transmitted to change the transfer rate if buffer underflow or overflow occurs. But it does not mention how to support transmission with bounded delay for every packet; instead it assumes implicitly that network resources are always available, thus the transfer rate can always be changed according to the consumption rate and synchronization requirement. Under their model, network resources is wasted when a higher transfer rate can be supported. Moreover, when network resources are not enough, buffer underflow can happen continually, and the transmission of feedback units can only waste more network bandwidth.

XTP

*Xpress Transfer Protocol* (XTP) [25] is one of the best documented lightweight
transport protocols. It combines the functionality of transport and internet protocols to produce a streamlined protocol, which is specially designed with high-speed networks in mind. It uses switch-like routing for routing packets quickly. XTP also allows for various degrees of error checking and is designed to be implemented using a parallel processing chipset. Flow control, which is basically a sliding-window strategy, is applied on a logical connection basis. It may make it difficult for this approach to support bounded delays.

**Warehouse Problem**

*Warehouse Problem* [1] is an operations research problem, which share some similarities to our buffer space problem. In a warehouse problem, a man is engaged in buying and selling identical items. He operates from a warehouse that can hold $N$ items. Each month, he can sell any quantity up to the stock at the beginning of the month. For each month, he can also buy as much as he wishes, as long as his total stock does not exceed $N$ items. The man also knows all the sale and purchase prices for every month of the whole year. The goal of the problem is to select a buy-and-sell policy, thus the man can make the maximum amount of money.

In a buffer space problem, we can treat the size of a buffer as the size of a warehouse (i.e., $N$), and in the buffer space problem, a transfer rate is only bounded by the size of the buffer, which is similar to the buy function of the warehouse problem. However, the consumption rate in buffer space problem is more regular, i.e., after playback starts, the data consumed is determined by the playback time (for fixed-size media). Thus if we assume that the consumption rate (corresponds to sell prices in the warehouse problem) is fixed, while the transfer rates (corresponds to purchase prices in the warehouse problem) can be determined according to the traffic situation,
we are able to determine the optimal transfer rates according to the traffic situation. (i.e., According to the traffic situation, transfer rates can be changed during the transmission phase.) In general, a buffer space problem is a much more regular problem than a warehouse problem, i.e., a buffer space problem can be treated a special case of a warehouse problem. In this thesis, we will not discuss the changes of a transfer rate, instead the transfer rate is determined at the channel set-up phase and will not be changed later.

1.4 Thesis Overview

The thesis proposes a reservation-based channel allocation scheme and an associated control mechanism for multimedia on-demand applications. Generally, there are two important QOS requirements for multimedia on-demand applications: playback performance and initial wait time. We select playback performance as the major requirement, and according to this requirement and the available buffer space at the destination, we calculate the minimum and maximum transfer rates. Based on this transfer rate range, our channel allocation scheme reserves bandwidth in the form of uniformly distributed time slots. This approach not only satisfies bandwidth and real-time requirements of multimedia on-demand applications, but also simplifies the channel establishing procedure. Thus, our scheme can satisfy QOS requirements for on-demand applications without large overhead. The associated traffic controller and scheduler are also described, followed by a performance analysis of this model and a discussion of other related problems.
The remaining chapters are organized as follows. We will give the general architecture of our model and discuss major QOS requirements for multimedia on-demand applications in Chapter 2. In Chapter 3, the calculation of the minimum and maximum transfer rates will be presented first, followed by a general rule for selecting a transfer rate. The major component of this chapter will present a new channel allocation scheme and the associated control mechanism used to fulfill QOS requirements. We will also discuss the impact of this channel allocation scheme on QOS requirements and other related problems. Chapter 4 will describe the characteristics of the buffer underflow and overflow problems for variable-size media and also present the formulae for the minimum and maximum transfer rates of such media. In Chapter 5, we will review the major results of this thesis and discuss future research problems.
CHAPTER 2

The Model

In this chapter, we first describe the general architecture of our model, followed by a discussion of major QOS requirements for multimedia on-demand applications.

2.1 General Architecture

In general, the major functional components of a multimedia on-demand system include multimedia file servers, the network and clients' equipment, which can be connected in many different ways. Multimedia file servers store and provide multimedia information to clients and clients' equipment decodes, decompresses and displays multimedia information. The network can be treated as a black box which transmits information from the file servers to the clients' equipment. Since multimedia on-demand systems are still not commercially available, there is no widely accepted standard for their general architecture.
CHAPTER 2. THE MODEL

Figure 2.1 illustrates an example of the architecture used in our model. In this example, several file servers are connected to high-resolution playback units, such as videophones, through a B-ISDN network. Playback units, which belong to clients of the system, are analogous to VCRs, but are connected through subscriber telephone lines or cable lines. In the file servers, multimedia files, such as video programs, are stored digitally on a large array of high capacity storage devices, such as optical disks. The multimedia information can be randomly accessed with a short seek time. A client can select multimedia files according to a variety of indices such as video titles, and request them to be transmitted in real-time and played back on his/her videophone.

A service for a multimedia on-demand application can be decomposed into three phases: set-up, data transmission and clear-up phases. To simplify their description, we treat the multimedia file server and a client’s equipment as a single user of network resources, so a network user will mean a set of application processes outside of the network. In this section, we describe a general scenario for these three phases, emphasizing the interaction between the user and the network.

Set-up Phase

A client wishing service from a multimedia on-demand system can send a request to the nearest network host. The host will calculate the possible transfer rate range and then forward the request to a multimedia file server at the source site.

In order to achieve guaranteed performance, the network and its users should carry out the following sequence of operations:

1. Before the actual communication begins, the network service and the user should
Figure 2.1: A sample multimedia system architecture.
agree upon:

- **Transfer parameters**: which include the packet size, the transfer rate and the length of the multimedia file. It is the user’s responsibility to ensure that the traffic characteristics will be maintained.

- **Performance specification**: network performance such as throughput, delay bounds, and loss rate required by the application. It is the network’s responsibility to keep this part of the agreement. In this thesis, throughput and bounded delay requirements will be achieved by our channel allocation scheme.

2. The network then uses a routing algorithm to find a logical channel; that is, a sequence of links from the source to the destination. By a **global logical channel** or simply a **channel**, we mean a connection from the server to the client, each with its own traffic and performance attributes. This connection may travel through several network links and, thus, every link has a **local logical channel** corresponding to this global logical channel. To find a suitable channel, we need to consider the state of the network system, the currently allocated channels and any special requirements for real-time communications.

3. The network may need to translate the global user-network agreement into a local requirement for every network node along the channel. The agreement mentioned in (1) is a global or end-to-end agreement, which does not rely on the topology or state of the network. From the user’s point of view, the most important goal is to ensure that the data will arrive at the destination node of the network before the deadline.

The end-to-end agreement only states the transfer parameters at the source host where the data enter the network. In our model, every link along the channel
CHAPTER 2. THE MODEL

will experience the same transfer parameters for a given user, and a channel’s performance parameters at every node are also the same. Thus, unlike some other models, such as Univ. of Calif. at Berkeley’s *real-time channel* [10], our global-to-local translation is straightforward. This is one of the advantages of our approach.

4. For every network link in a planned logical channel, an **admission test** is performed. The test should include:

   (a) Processing capacity test, to ensure that there is enough bandwidth in the link and processing capacity for all packets at entry node to the link along a channel.

   (b) Buffer space requirement test, to ensure that packets will not be dropped because of buffer overflow at the entry node to the link.

   (c) Deadline requirement test, to ensure packets belonging to different channels will not miss their deadlines.

If the admission test succeeds at a network link, the required resources can be reserved.

5. Finally, if all the links along the channel pass the admission tests, the request is accepted, and if required, a confirmation will be sent back to every node along the channel. Otherwise, the request is either rejected or the routing algorithm is invoked again to find an alternate channel. If the channel is accepted, we proceed to the next phase, i.e., data transmission.

**Data Transmission Phase**
Mechanisms are needed to ensure the agreed-upon transfer parameters and performance requirements. These mechanisms include:

1. A traffic controller at every network node to ensure that the file server or the previous network node along the channel does not send packets to the next node at a rate higher than what was promised. The rate controller verifies and forces the packets to wait at nodes along the channel until they are expected to arrive.

2. A scheduler at every network link along the channel to order packets’ transmission. In doing so, it ensures that all packets will be transmitted without missing their deadlines if the user obeys the user-network agreement.

In our model, (1) and (2) are combined into a single mechanism which ensures that only when a packet reaches its scheduled time, is it transmitted. Figure 2.2 illustrates the general structure of a network node in our model.
CHAPTER 2. THE MODEL

Clear-Up Phase

When the whole transmission is over, the channel is cleared and the reserved buffer space is freed.

2.2 Quality of Service (QOS)

Because of the characteristics of real-time multimedia communications, protocols should follow a predictive scheme, whereby a new request for a channel is rejected when the QOS guarantees made to the existing users could be violated. To fulfill QOS requirements, the network system has to reserve suitable resources, including processing capacity and buffer space at the appropriate nodes. However, unlike interactive multimedia applications, data are generated beforehand in multimedia on-demand applications, and, therefore, it is easy to determine the transfer parameters and QOS requirements, and it is also possible to adjust the transfer parameters according to the available network and buffer resources. In summary, transmission can only start after a user and the network agree on the transfer parameters and performance requirements. Furthermore if the users obey the transfer parameters agreed to, then their QOS requirements will be guaranteed by the network.

In this section, we will identify the major performance requirements of multimedia on-demand applications and the associated transfer parameters and performance parameters.
2.2.1 QOS Specification

Transfer parameters and performance requirements should be carefully selected, since the allocation of resources will depend solely on them. Before a discussion of the major QOS requirements for multimedia on-demand applications, we give some desirable properties of their specification:

- The network should be able to verify whether a user obeys the promised transfer parameters. Otherwise, a misbehaving user can degrade the performance of the whole system.

- The specification must allow ease of resource allocation and admission tests without large overhead. Because of the characteristics of real-time communications, large overhead is always a drawback.

In general, the QOS requirements can be provided in one of two ways:

- The user specifies all the performance requirements, and the network checks the available resources to decide whether the request can be granted.

- The user only specifies the major requirements and the other performance parameters are provided by the network. The user can accept or reject the service, depending on whether he/she is satisfied with the network-supplied parameters.

In this thesis, we assume the second approach.
2.2.2 QOS for Multimedia On-demand Applications

There are two important performance requirements for a multimedia on-demand application:

1. **Playback performance**, which is the quality of playback. There are several factors which can affect playback performance, the most important one being the delivery rate, which is the rate at which data are output from the network into a client’s buffer. Other factors include the reliability of communications, sequencing and duplication.

2. **Initial wait time**, which is the period of time between the receipt of a request to the beginning of the playback.

In this thesis, we select playback performance as the only major requirement because:

1. From a client’s point of view, the quality of playback is most important; as long as it is not disrupted, the client’s basic requirement is satisfied.

2. If we only focus on the playback performance, our channel allocation and control mechanism will be simpler, and thus the total channel establishing time will be shorter.

3. It also frees users from having to decide upon other parameters. (This can be a drawback in terms of flexibility.)

So, we will focus on the playback performance and reserve network resources accordingly. In the remaining part of this chapter, we will study playback performance
and discuss the associated transfer parameters and performance parameters. The initial wait time will be discussed in Section 3.4.

### 2.2.3 Playback Performance

The digitization of multimedia information is usually achieved by sampling an analog signal at fixed intervals and storing the value of each sample in binary form. In this thesis, a set of samples which forms a unit of playback is called a **frame**. Generally, the packet size may be larger or smaller than the frame size; that is, a packet may carry more than one frame, or one frame may be transmitted by several packets.

In general, there are two kinds of media:

1. **Fixed-size media**, such as uncompressed audio and text files, in which each frame consists of a constant number of bytes. This type of media also includes linearly compressed media (see Figure 2.3).

2. **Variable-size media**, such as video or nonlinearly-compressed audio, in which frames consist of variable numbers of bytes. For example, a constant number of frames may be displayed every second for video, but the number of bytes of data displayed every second may be variable (see Figure 2.4).

We will first consider fixed-size media. Then, in Chapter 4, we will consider the real-time requirements for variable-size media.

During playback, frames are consumed at a constant rate, called the **consumption rate**. Failure to provide a frame at the correct time will result in playback disruption.
CHAPTER 2. THE MODEL

Figure 2.3: Fixed-size media.

Figure 2.4: Variable-size media.
Of course, we can allocate buffer space at the destination to prefetch some data. However, this introduces two problems.

If the delivery rate is lower than the consumption rate, frames must first be prefetched, and then passed to the playback unit at the correct time. After playback begins, if we can ensure that the buffer always holds the next frame, then the real-time deadlines for providing the playback unit with frames can be met. Of course, we can always achieve this by prefetching all the data into the client's buffer before playback begins. However, this will require an unreasonably large buffer, as well as a long wait before playback can begin. The problem then becomes one of how to minimize the required buffer size and prefetching delay. We call this the buffer underflow problem.

On the other hand, if the delivery rate is higher than the consumption rate, we need buffer space to store the data until they are displayed. We call this the buffer overflow problem.

Gemmell [13] has studied the real-time requirements of retrieving uncompressed audio from secondary storage. He points out that the solution of using the minimum start time for playback also requires the least buffer space. So, we only need to solve either the buffer space requirement or the least wait time problem.

The delivery rate was defined before as the number of packets per second which are output from the network into a buffer at a destination. The (end-to-end) transfer rate is defined as the number of packets per second transmitted from a source to the associated destination. We use a scheduler at every node to ensure that packets are transmitted in the assigned time slots, and the interval time between two successive
packets is the same, thus achieving the required transfer rates. Since the (end-to-end) transfer rate and the delivery rate are the same, (assuming that no packets are dropped), we will refer to them as the **transfer rate**. Note that since the interval time between two successive packets is the same, to simplify analysis, we assume that the buffer fills with arriving packets as a linear function of time.

In this thesis, we will focus on the buffer size and the possible transfer rate and calculate the **minimum transfer rate** $R_{t,\text{min}}$, (when all available buffer space is used to store prefetched data) and the **maximum transfer rate** $R_{t,\text{max}}$, (when all available buffer space is used to buffer early arrived data). These two parameters provide a range from which a transfer rate for the request can be selected. However, the transfer rate is constrained not only by the size of the buffer space at the destination, but also by other factors such as the compression rate and processing capacity of the destination, which could be easily added to our model.

To simplify the problem of selecting a suitable transfer rate, we divide the time on each link into slots, each slot having a length equal to the time for a single packet to be output from an output queue into the link. In an ATM-based network, all packets have the same size, so the slot size is fixed. We number the slots $0, 1, 2, \ldots$ and use them as units of bandwidth allocation.

Consider a link $\ell$ with bandwidth $p$ (packets/sec). If a request requires a transfer rate of $p/2$, then we divide the bandwidth of link $\ell$ into two logical channels, $C_{\ell}^0$ and $C_{\ell}^1$, and use one of them. $C_{\ell}^0$ ( $C_{\ell}^1$ ) consists of the even-(odd-)numbered time slots of link $\ell$. If the request requires a transfer rate $p/4$, then we divide $C_{\ell}^0$, for example, into two subchannels, $C_{\ell}^{00}$ and $C_{\ell}^{01}$, and use one of them. $C_{\ell}^{00}$ ( $C_{\ell}^{01}$ ) consists of the even-(odd-)numbered slots of $C_{\ell}^0$, and so on.
CHAPTER 2. THE MODEL

In general, the time slots of link $\ell$ can be decomposed into different local channels, $C_{\alpha_1}^\ell$, $C_{\alpha_2}^\ell$, $C_{\alpha_3}^\ell$, ..., such that local channel $C_{\alpha_j}^\ell$ has a speed of $2^{-k_j} \cdot p$ (packets/sec), where integer $k_j$ is the length of the binary string $\alpha_j$ (e.g. $C_{011}^\ell$ has speed $p/8$ since $\alpha_j = 011$ and $k_j = 3$.)

When an unallocated local channel, $C_{\alpha_{k-1}}^\ell$, with a speed of $2^{-(k-1)} \cdot p$, is too fast for the request under consideration, we divide it into two local channels, each with the transmission speed of $2^{-k} \cdot p$, and allocate one of them to the request. If they are still too fast, this division can be applied repeatedly. If two adjacent links $\ell$ and $\ell'$ have transmission speeds of $p$ and $p'$ respectively such that $p = 2^h \cdot p'$ for some integer $h$, then for an arbitrarily chosen local channel for $\ell'$, one can always find a local channel for $\ell$ with the same speed as what of the channel for link $\ell'$.

As mentioned above, besides the transfer rate, other factors, such as loss and error rates, sequencing and duplication, can also affect the playback performance. Loss can occur because of buffer overflow, packets missing their deadlines and noise in the links. Since noise in the links is not predictable, only the first two factors are determinate. To simplify the problem, in this thesis, we will not consider the loss rate. Instead, buffer space requirements are tested at every node including the destination, when a channel is established, to ensure loss will not occur because of buffer overflow. As mentioned in Section 1.2, we assume management information is transmitted in a separate channel. By using separate logical channels for management information and data, we can achieve:

1. Reliability, since special error checking and recovery techniques can be used for the management information channel and a data channel, according to the QOS requirements of different media.
2. Efficiency, by allowing several users to share one channel for transmission of management information.

3. Shorter establishing time, since the management channel usually exists before the data channel has been set up, so it can be used to save time.

In real-time communications, because of the delay requirements of channels, it is not easy and also not reasonable to use a "time-out-and-retransmit" mechanism to achieve reliability. On the other hand, most multimedia applications are fault-insensitive to some extent. Thus we can use error recovery mechanisms and/or interpolation algorithms at destinations to compensate for the corrupted data. In this thesis, however, we will not consider such errors. We assume that either users can tolerate the corrupted data or there are some suitable mechanisms to compensate for the corrupted data. Also, since, in our model, packets are transmitted from the beginning of a file to the end via the same route and there is no retransmission, sequencing and absence of duplication can be guaranteed.

In the next chapter, we discuss transfer rates and present a new channel allocation scheme.
CHAPTER 3

Channel Allocation

As mentioned in the previous chapter, the minimum and maximum transfer rates provide a range from which a transfer rate for a new request can be selected. In the first section of this chapter, we calculate this range from the consumption rate and the available buffer space, and then give a rule for selecting a transfer rate. The main objective of this chapter is to present a new channel allocation scheme (Section 3.2). In Section 3.3, a control mechanism used to fulfill QOS requirements is presented. We discuss the impact of our channel allocation scheme on QOS requirements in Section 3.4. Finally, in Section 3.5, we discuss other problems related to our channel allocation scheme.
3.1 Transfer Rate

In this section, we derive the formulae for the minimum transfer rate, $R_{t,\min}$, and maximum transfer rate, $R_{t,\max}$, for fixed-size media. We first make the following assumptions:

1. The network’s maximum error rate is acceptable. (Therefore, we do not discuss error control issues.)

2. The buffer fills with the arriving packets linearly with time.

3. The time for a packet to be output from the network to the client’s buffer depends only on its size.

Let $f$ (packets) be the frame size. Let $R_c$ (packets/second) and $B$ (packets) be the given consumption rate and the size of the client’s buffer, respectively, and let $F$ (packets) be the size of a multimedia file to be played back. We assume that, while playback is in progress, a frame in the buffer is used exclusively for playback and no part of it should be modified. Immediately after a frame is played back, the next frame should be available. This implies that

$$B \geq 2 \cdot f. \quad (3.1)$$

Note that our assumptions are different from Gemmell’s [13], who assumes that as a frame is being played back, the buffer space for the frame can be used to buffer new data.
CHAPTER 3. CHANNEL ALLOCATION

We calculate $R_{t,\text{min}}$ and $R_{t,\text{max}}$, considering two cases.

1. $R_t < R_c$ (Underflow Problem):

Before playback begins, we buffer as much data as possible. Figure 3.1 shows the number of packets in the buffer as a function of time. A displayed frame is shown as being instantaneously consumed. Playback should start when there is only $f \cdot R_t / R_c$ (packets) of free space left in the buffer. Thus when the first frame has been played back, which takes $f / R_c$ seconds, the buffer will be full except for one frame worth of free space, which has just been emptied. At this point, $F - B$ packets of the remaining file are yet to be transmitted. The transmission of these packets must complete before the playback of the last frame of the file can start. Thus

$$\frac{F - 2 \cdot f}{R_c} \geq \frac{F - B}{R_t}.$$  \hspace{1cm} (3.2)

From Inequality (3.2), we can derive the minimum transfer rate as:

$$R_{t,\text{min}} = R_c \cdot \frac{F - B}{F - 2 \cdot f}, \text{ if } F > B.$$  \hspace{1cm} (3.3)

From Inequality (3.2), we know that both $(F - 2 \cdot f) / R_c$ and $(F - B) / R_t$ should be positive, thus it requires $F > B$. If $F \leq B$, we can prefetch the whole file before playback begins, and this would drastically reduce the minimum transfer rate. In this case, the minimum transfer rate depends on other factors, such as the initial wait time.

2. $R_t \geq R_c$ (Overflow Problem):
CHAPTER 3. CHANNEL ALLOCATION

Figure 3.1: Calculation of minimum transfer rate, $R_{t,\text{min}}$, when $R_t < R_c$. 

Number of packets in client's buffer

\[ B - f \frac{R_t}{R_c} \]

slope = $R_t$

\[ \frac{B - f \frac{R_t}{R_c}}{f} = \frac{(F-B)R_t}{(F-2f)R_c} \]

end of playback

start of transmission

end of transmission and is also start of the playback for the last frame
CHAPTER 3. CHANNEL ALLOCATION

Figure 3.2: Calculation of maximum transfer rate when $R_t > R_c$ (the case where $f/R_c < (F-f)/R_t$ and $2 > F/f - [(F/f - 1) \cdot R_c/R_t] \cdot R_t/R_c$).

Since data arrive faster than they can be played back, we start playback as soon as a frame is available in the buffer. See Figures 3.2 and 3.3.

We consider two cases:

1. Transmission of the remaining file $F-f$ takes longer than playback of the first frame; that is, $f/R_c < (F-f)/R_t$. (Figures 3.2 and 3.3).

2. The transmission of the remaining file completes before playback of the first frame finishes; that is, $f/R_c \geq (F-f)/R_t$. (Figure 3.4).

In the first case, transmission begins at time $A$ (Figure 3.2) while playback starts at time $S$. The line segment connecting $Q$ and $S$ corresponds to the amount of initial data (buffered data at the time playback starts), which is equal to $f$. $MD$
Figure 3.3: Calculation of maximum transfer rate when \( R_t > R_c \) (the case where \( f/R_c < (F - f)/R_t \) and \( 2 \leq F/f - [(F/f - 1) \cdot R_c/R_t] \cdot R_t/R_c \)).

corresponds to the amount of data in the buffer when the transmission completes at time \( D \). \( OC \) corresponds to the amount of data in the buffer when frame \( l \) finishes (at time \( C \)), where frame \( l \) is the last frame which finishes its playback before or at the time when the transmission finishes. If \( MD \) is shorter than \( OC \), the maximum buffer space requirement appears at time \( C \), instead of time \( D \) (Figure 3.2). However, if \( MD \) is longer than \( OC \) (Figure 3.3), the maximum buffer space requirement appears at time \( D \), instead of time \( C \). By checking whether the amount of data buffered, after frame \( l \) finishes its playback, is more than \( f \), we can determine at which point the maximum buffer space requirement occurs.

In general, we calculate the required buffer space \( B \) for this case as:

\[
B \geq \text{space for initial data}
\]
CHAPTER 3. CHANNEL ALLOCATION

Figure 3.4: Calculation of maximum transfer rate when \( R_t > R_c \) (the case where \( f/R_c > (F - f)/R_t \)).

\[ + \text{space for data buffered between time } S \text{ and time } C \]
\[- \text{space for data consumed between time } S \text{ and time } C \]
\[ \text{if the data in } l \text{ is less than the amount of data buffered after the playback of } l \text{ finishes,} \]
\[ + \text{space for data buffered after the playback of } l \text{ finishes} \]
\[ \text{else} \]
\[ + \text{space for the last frame } l \text{ (i.e., } f) \]. \hspace{1cm} (3.4) \]

Note that \[ [(F - f)/R_t] \cdot R_c/f \] is the number of frames which can finish playback during the transmission time, and \[ [(F - f)/R_t] \cdot (R_c/f) \cdot f \cdot (R_t/R_c) \] is the data (packets) buffered between time \( S \) and time \( C \). Thus we get the buffer space requirement \( B \) as:
We now rewrite Inequality (3.5) as:

\[ B \geq f + \left[ \frac{F - f}{R_t} \cdot R_c \right] \cdot f \cdot \left( \frac{R_t}{R_c} - 1 \right) \]
\[ + \max \left\{ f, F - f - \left[ \frac{F - f}{R_t} \cdot \frac{R_c}{f} \right] \cdot f \cdot \frac{R_t}{R_c} \right\}, \text{ if } F > B. \]  

(3.5)

We now rewrite Inequality (3.5) as:

\[ \frac{B}{f} \geq 1 + \left[ \left( \frac{f}{f} - 1 \right) \cdot \frac{R_c}{R_t} \right] \cdot \left( \frac{R_t}{R_c} - 1 \right) + \max \left\{ 1, \frac{F}{f} - 1 - \left[ \left( \frac{f}{f} - 1 \right) \cdot \frac{R_c}{R_t} \right] \right\} \frac{R_t}{R_c}. \]

(3.6)

Note that Inequality (3.6) involves only “normalized” parameters, \( B/f, F/f \) and \( R_t/R_c \).

In the second case (see Figure 3.4), the whole file has been retrieved before even the first frame finishes its playback. The maximum buffer space requirement appears at time \( G \) when the transmission finishes. Note that, in the second case, the buffer space cannot be smaller than the file size \( F \); that is \( B \geq F \). Also the transfer rate can be arbitrarily high and would be bounded by other factors such as the decoding rate of the client’s equipment.

Using these formulae, we can also calculate the maximum buffer space used, \( U_{\text{max}} \), when a transfer rate is fixed. If the transfer rate \( R_t \) is higher than the consumption rate \( R_c \) (buffer overflow), \( U_{\text{max}} \) is calculated according to Inequality (3.6):

\[ \frac{U_{\text{max}}}{f} = 1 + \left[ \left( \frac{f}{f} - 1 \right) \cdot \frac{R_c}{R_t} \right] \cdot \left( \frac{R_t}{R_c} - 1 \right) + \max \left\{ 1, \frac{F}{f} - 1 - \left[ \left( \frac{f}{f} - 1 \right) \cdot \frac{R_c}{R_t} \right] \right\} \frac{R_t}{R_c}. \]

(3.7)
Figure 3.5: $U_{\text{max}}/f$ vs. $R_t/R_c$. (Maple was used to draw this figure.)
CHAPTER 3. CHANNEL ALLOCATION

If the transfer rate $R_t$ is smaller than the consumption rate $R_c$ (buffer underflow), $U_{\text{max}}$ is computed according to Inequality (3.2):

$$\frac{U_{\text{max}}}{f} = \frac{F}{f} - \left(\frac{F}{f} - 2\right) \cdot \frac{R_t}{R_c}. \quad (3.8)$$

Examples are provided below which illustrate the relationship between $U_{\text{max}}$, $F$ and $R_t/R_c$.

**Example 1.a:** Figure 3.5 illustrates the relationship between the required buffer space $U_{\text{max}}/f$ and the ratio of the transfer rate over the consumption rate, $R_t/R_c$, for $F/f = 25$. The horizontal and vertical axes represent $R_t/R_c$ and $U_{\text{max}}/f$, respectively. When $R_t/R_c \leq 1$, the relationship is linear corresponding to Inequality (3.2). When $R_t/R_c \geq 1$, there is a curve, from which we see that in some ranges, the buffer space requirement increases while the ratio $R_t/R_c$ increases, while in others, the buffer space requirement remains constant. It is easy to understand that the buffer space requirement does not decrease as the ratio increases, since, as the ratio increases, less data is consumed during the transmission period. The flat line segments correspond to the case illustrated in Figure 3.3. From Equation (3.7), it is clear that the value of $U_{\text{max}}/f$ remains the same as $R_t/R_c$ changes within small ranges. For a larger ratio $R_t/R_c$, the range of $R_t/R_c$, which generates the same $\lfloor (F/f - 1) \cdot R_c/R_t \rfloor$ and thus requires the same buffer space size, is also bigger.

**Example 1.b:** Figure 3.6 illustrates the relationship between $U_{\text{max}}/f$ and $F/f$, for $R_t/R_c = 4$. The horizontal and vertical axes represent $U_{\text{max}}/f$ and $F/f$, respectively. Naturally, we know that when the file size increases, we need more buffer space, provided other parameters are fixed. As in Figure 3.5, there are some ranges in which
Figure 3.6: $U_{max}/f$ vs. $F/f$ when $R_t/R_c > 1$. (*Maple* was used to draw this figure.)
the buffer space requirement remains constant as the file size increases. Note that these ranges have the same length. This can be explained using Figure 3.2. When the transmission finishes at time $D$, the required buffer size is the same as the case where the transmission finishes at time $C$, namely, $QC$. Since the size of a frame is constant, these ranges have the same length. Note that if $R_t/R_c < 1$, then the relationship between $U_{\text{max}}/f$ and $F/f$ would no longer look like Figure 3.6. The precise relationship between them is given by Equation (3.8).

**Example 1.c:** Figure 3.7 illustrates the relationship between $F/f$ and $R_t/R_c$. When $R_t/R_c \geq 1$ (buffer overflow), the surface represents $U_{\text{max}}/f$ of Equation (3.7) as a function of $F/f$ and $R_t/R_c$, and the plane represents the equation $U_{\text{max}}/f = 5$. Therefore, the intersection of the plane with the surface gives us the maximum transfer rate $R_{t,\text{max}}/R_c$ for any given $F/f$. When $0 < R_t/R_c \leq 1$ (buffer underflow), the relationship between the minimum transfer rate $R_{t,\text{min}}/R_c$ and $F/f$ can be derived from Equation (3.3). The intersection of the surface with the plane gives us the minimum transfer rate $R_{t,\text{min}}/R_c$ for any given $F/f$. It can be observed in Figure 3.7 that if $F < B$, (which is unlikely to happen for a huge multimedia file), then $R_{t,\text{min}} = 0$ and $R_{t,\text{max}} = \infty$. As $F/f$ increases, the difference between $R_{t,\text{min}}$ and $R_{t,\text{max}}$ becomes smaller and smaller. Note that Figure 3.7 (3-dimensions) gives us more information than Figures 3.5 and 3.6 (2-dimensions). For example, when $F/f$ is fixed, (imagine there is a plane parallel to the $B/f$ and $R_t/R_c$ axes, which represents the fixed $F/f$), we get the same relationship of $B/f$ and $R_t/R_c$ as that in Figure 3.5 (the intersection of the surface with the imagined plane). Similarly, we can also get the relationship between $F/f$ and $B/f$ when $R_t/R_c$ is fixed.
Figure 3.7: $F/f$ vs. $R_t/R_c$. (*Maple* was used to draw this figure.)
Figure 3.8: $U_{max}$ vs. $R_t$ (in frames/sec) for PCM voice. (*Maple* was used to draw this figure.)
Example 2: Figure 3.8 was generated based on the parameters of a real example [21]. In this example, PCM voice is transmitted to a client, in which each frame (sample) consists of 8 bits (uncompressed), 8000 frames are played back per second and the total playback time is 90 minutes. From Figure 3.8, we see that when $R_t = R_c$, $U_{\text{max}}$ is the smallest (i.e., 2 bytes) and $U_{\text{max}}$ increases as $R_t$ deviates from $R_c$. \(\square\)

The minimum buffer space required at the client’s site is $B = 2 \cdot f$ (Inequality (3.1)). In this case, $R_{t,\text{min}} = R_c$ (Equation (3.3)). According to Inequality (3.5), $B = 2 \cdot f$ implies $[(F/f - 1) \cdot R_c/R_t] \cdot (R_t/R_c - 1) = 0$ (since it cannot be negative). However, if $[(F/f - 1) \cdot R_c/R_t] = 0$, then $2 \cdot f \geq F$, which is unlikely for most real multimedia files. (As mentioned before, when $B \geq F$, the maximum transfer rate depends on other factors.) Thus, $R_t/R_c - 1 = 0$, which implies $R_{t,\text{max}} = R_c$. In summary, when $B = 2 \cdot f$, $R_{t,\text{min}} = R_{t,\text{max}} = R_c$.

To minimize a client’s buffer space, we could let $R_t = R_c$. In this case, we only need a buffer of size $2 \cdot f$; one buffer space of size $f$ for buffering new data and another for playback. However, since there are probably peak and non-peak hours for multimedia on-demand services, this approach may limit the number of users that could be supported. The selection of a transfer rate should be based on the current traffic pattern and network configuration, within the calculated transfer rate range.

In the following section, we present algorithms for allocating a logical channel according to the selected transfer rate.
3.2 Channel Allocation Scheme

3.2.1 Local Channel Allocation

In this section, we present a rule for selecting the transfer rate, followed by a local channel allocation algorithm.

As mentioned before, the minimum and maximum transfer rates provide a range from which to choose a suitable transfer rate. The maximum transfer rate \( R_{t,\text{max}} \) is the highest \( R_t \) which still satisfies Inequality (3.5). Due to the truncation operation in Inequality (3.5), it is not easy to calculate the maximum transfer rate, \( R_{t,\text{max}} \) directly. But since our channel allocation scheme, introduced in Section 2.2.3, provides a channel with a transmission speed of \( p \cdot 2^{-k} \) (packets/sec), in which \( k \) is an integer and \( p \) (packets/sec) is the entire bandwidth of a link, we can still use Inequality (3.5) to compute the range of a transfer rate as follows. First, we calculate the minimum transfer rate \( R_{t,\text{min}} \) from Equation (3.3) and the largest \( k \) such that \( p \cdot 2^{-k} \geq R_{t,\text{min}} \) and let \( R_t = p \cdot 2^{-k} \). If \( R_t \) violates (3.5), it means \( 2^{-k} \cdot p > R_{t,\text{max}} \). In this case, some slots in the allocated logical channel will go unused. Otherwise, keep doubling \( R_t \), \( (R_t = 2 \cdot R_t) \) as long as \( R_t \) does not violate Inequality (3.5). Note that when we select a transfer rate, we need only calculate \( k \) instead of a particular channel.

With the selected transfer rate \( R_t \), we use the following local channel allocation algorithm. At every node, there is a local channel manager, which is responsible for allocating local logical channels connecting it to its neighbouring nodes, and collecting released local channels. Let \( \ell \) be a link with bandwidth \( p \) (packets/sec), and let \( C_{\alpha}^{\ell} \) represent a logical channel, where the subscript \( \alpha \) is a binary string of length \( k \). Given
the method of channel division described in Section 2.2.3, a binary tree is a natural
data structure for maintaining and managing allocated local channels.

Initially, this binary tree has the root as the only leaf (whose level is 0), which
 corresponds to the entire bandwidth of the link. To allocate a channel with bandwidth
 $2^{-k} \cdot p$ to a new request, we search the tree to find an unallocated leaf whose level is
 $k$. If we can find such an unallocated leaf, then we have found a path corresponding
to a channel with bandwidth $2^{-k} \cdot p$, thus we allocate the corresponding channel to
the request and mark the leaf as allocated. If there is no such leaf, we go back to
the level $k - 1$ to find an unallocated leaf. This procedure proceeds until we find an
unallocated leaf at a level which is smaller than $k$. If unsuccessful, then we have to
reject the request or multiplex several channels to support the request. If successful,
then we split it into two unallocated leaves. If these leaves are still not at level $k$, one
of them is split into two unallocated leaves again. This procedure continues until the
level of the new leaves is $k$. We mark one of them as allocated and allocate it to the
request.

To release a local channel, we traverse the tree according to the subscript of the
channel. That is, for every bit of the subscript, 0 for the left child and 1 for the right
child, we travel down the tree (see Figure 3.9) until we arrive at the leaf, then we mark
the leaf as unallocated and go back to its parent node. If the parent node has two
unallocated leaves, we free these leaves and change the parent node to an unallocated
leaf. This procedure proceeds until either the parent has only one unallocated leaf or
we reach the root.

The algorithms are simple and, through the binary tree, it is quite easy to deter-
mine which slots are being allocated. (See Figure 3.9, in which channel $C_{000}$, $C_{01}$ and
Figure 3.9: Binary tree for channel allocation scheme.
CHAPTER 3. CHANNEL ALLOCATION

48

$C_{11}$ are allocated while channel $C_{001}$ and $C_{10}$ are not). But the search procedure can be quite time-consuming, so we can use the binary tree structure to illustrate unallocated and available channels as network management information and use other data structures to implement the channel allocation scheme.

One possibility is the use of linked lists. The associated algorithm is as follows: for each $k = 0, 1, 2, \ldots$, we maintain all the unallocated local channels with bandwidth $2^{-k} \cdot p$ in a linked list. At the beginning, only the first list with bandwidth $p$ (i.e., $k = 0$) is not empty, and all other lists are empty. Channels $C^t_\alpha$ and $C^t_\beta$ are called buddy channels, if $k_i = k_j$ (the lengths of the subscripts are the same) and all but the last digit of $\alpha_i$ and $\alpha_j$ agree. For example, in Figure 3.9, channels $C_{10}$ and $C_{11}$ are buddy channels.

To allocate a channel with bandwidth $2^{-k} \cdot p$, we check the linked list corresponding to $k$. If this list is empty, the list for $k - 1$ is searched, and so forth. If a local channel is found in the list for $k' < k$, we separate the channel into several subchannels, of which we select one local channel with bandwidth $2^{-k} \cdot p$ to the request, inserting the remaining subchannels into the appropriate lists. The remaining subchannels include one with bandwidth $2^{-k'-1} \cdot p$, a second with $2^{-k'-2} \cdot p$, \ldots, and a final one with $2^{-k} \cdot p$. If the list for $k'$ is empty for all $k' \leq k$, then the local channel allocation fails and thus the request may be rejected, or several local logical channels from lists $k'' > k$ are allocated for the request.

When a local channel $C^t_\alpha$, whose bandwidth is $2^{-k} \cdot p$, is released, it is placed in the list for $k$. If its buddy channel is also in the list, we merge these two local channels to form a channel with bandwidth $2^{-k+1} \cdot p$ whose subscript is the same as the first $k - 1$ digits of $\alpha$, and put it in the linked list for $k - 1$. Again, if two buddy channels
3.2.2 Global Channel Allocation

In the above section, we discussed the local channel allocation scheme for one link. However, since the bandwidths of different links can vary, a global channel whose bandwidth can be satisfied at one link may not be satisfied at the next link. In the beginning, it may not be an optimal choice to reserve a particular local channel for a potential end-to-end channel. Such a local channel should be allocated to a global channel which has been determined to be feasible. To avoid the unnecessary division of a local logical channel into subchannels, instead of reserving a real local channel, we temporarily reserve bandwidth of a link until the admission test has been passed at every link along the route. Then we allocate a real local channel at every link along the route. In this approach, only the available bandwidth needs to be checked. If there is enough bandwidth, we reserve the required bandwidth at this link and proceed to the next one, otherwise a reject signal is sent to the user and temporary reservations are cancelled. When the destination is reached, a success signal is sent to all the links along the route with the bandwidth that every link along the route can provide. Thus a local channel allocation algorithm can be run in parallel at each node along the route, which decreases the channel establishing time.

Figure 3.10 illustrates our channel allocation scheme. For example, at link B-V, which connects network node B to client C, we allocate the local channel $C_{0}^{B,V}$, which consists of slots 0, 2, 4, 6, ..., to a request. Assuming that the bandwidth of link A-B is twice that of link B-V, we need only allocate the local channel $C_{01}^{A,B}$, which consists
of slots 2, 6, 10, …

After a global channel is allocated to a new request, the data transmission phase is entered. In the following section, we present a control mechanism which satisfies QOS requirements during this phase.
3.3 Node Controller and Transmission Scheduler

A packet may spend a variable amount of time at a network node and link (this variability of transmission time is called delay jitter), and a file server may intentionally or unintentionally send packets faster than the rate agreed to in the channel set-up phase. Both of them can cause the traffic pattern at subsequent nodes to be different from what was expected, thus causing the QOS requirements of some “well-behaved” channels to be violated if no appropriate actions are taken. It is necessary for every output queue to change unregulated input traffic into regulated output traffic; we call this traffic reconstruction. Otherwise, performance of subsequent network nodes can be disrupted. As mentioned before, in order to satisfy local real-time requirements, two mechanisms are needed: a controller and a scheduler. (Figure 3.11).

A traffic controller is needed to reconstruct the traffic from the previous network node along a channel or a local file server. Even if other nodes or local file servers do not intentionally violate the agreements reached during the channel establishing phase, it is still possible for local problems (such as a file server unintentionally sending packets too fast because of hardware problems) at these nodes to occur, which may not be apparent immediately. In order to prevent such a local problem from disrupting the performance of the entire system, a controller is used to check whether a packet arrives too early.

Delay jitters can also cause traffic irregularity in subsequent nodes along a channel, so it is necessary for the controller to calculate or estimate the expected arrival times of packets along every logical channel through the node. For example, in our model, a complete packet should not arrive before the previous packet of the same channel has
been sent. The arrival times could be calculated precisely in our model, but, since for every logical channel there is always one packet-sized buffer available at every intermediate node, it is not necessary to be precise. In general, if a packet arrives earlier than its expected time, it should be held until its expected time, otherwise, this packet might affect the performance guarantees of other packets at subsequent nodes.

**Clock drifts**, when clocks used by two successive nodes on a global channel go out of synchronization, may also cause traffic patterns at subsequent nodes to differ from what was expected. Fortunately, most of the time we can assume that the difference in the clocks is fixed during a transmission period, thus we can control the impact of clock drifts when we schedule the transmission time of packets.

A **scheduler** is used to select the next packet from the output queue to be sent to the subsequent node. In our model, the scheduler can be combined quite easily with the controller.

At every node, a mechanism is used as a controller and scheduler, providing several functions:

1. Calculating slot numbers for incoming packets.
2. Holding packets arriving too early.
3. Sending packets to the output port at the appropriate times. (Note that the mechanism can be shared by several output links.)

We now present a method for calculating slot numbers belonging to a particular channel. From Section 2.2.3, we know that in our model, bandwidth is divided into
Figure 3.11: Controller and scheduler.

slots. Recall that, if we number the time slots of link \( \ell, 0,1,2, \ldots \), then \( C_0^\ell \) consists of the even-numbered slots, 0, 2, 4, \ldots. So, for slots \( i = 0, 1, 2, \ldots \), the \( i \)th slot belonging to \( C_0 \) is given by:\(^1\)

\[
C_0(i) = 2i.
\]

Similarly, \( C_1 \) consists of the odd-numbered slots, i.e.,

\[
C_1(i) = 2i + 1.
\]

Let \( k \) be the length of the subscript \( \alpha \) for channel \( C_\alpha \), i.e., \( \alpha = d_1d_2\ldots d_k \). We define \( C_\alpha \) recursively as follows. Given \( C_{d_1d_2\ldots d_{k-1}} \),

\[
C_{d_1d_2\ldots d_{k-1}}(i) = \text{the } i\text{th even slot in } C_{d_1d_2\ldots d_{k-1}} = C_{d_1d_2\ldots d_{k-1}}(2i), \quad (3.9)
\]

and

\[
C_{d_1d_2\ldots d_{k-1}}(i) = \text{the } i\text{th odd slot in } C_{d_1d_2\ldots d_{k-1}} = C_{d_1d_2\ldots d_{k-1}}(2i + 1). \quad (3.10)
\]

\(^1\)We omit the superscript \( \ell \), since it is clear from the context.
We use $C_{d_1 d_2 \cdots d_{k-1}}(2i + d_k)$ to represent both Equations (3.9) and (3.10), so we get:

$$C_{d_1 d_2 \cdots d_{k-2}}(i) = C_{d_1 d_2 \cdots d_{k-2}}(2 \cdot (2i + d_k)) = C_{d_1 d_2 \cdots d_{k-2}}(2^2 \cdot i + 2 \cdot d_k),$$

and

$$C_{d_1 d_2 \cdots d_{k-2}}(i) = C_{d_1 d_2 \cdots d_{k-2}}(2 \cdot (2i + d_k) + 1) = C_{d_1 d_2 \cdots d_{k-2}}(2^2 \cdot i + 2 \cdot d_k + 1).$$

We repeat this procedure and derive the following formula:

$$C_{d_1 d_2 \cdots d_k}(i) = 2^k \cdot i + d_1 + d_2 \cdot 2 + \ldots + d_k \cdot 2^{k-1} \quad (3.11)$$

At every node, the slot number of the first packet is recorded for every local channel passing through this node. Other information such as a channel’s label (i.e., its subscript) is also stored for every channel. When a packet arrives, the associated slot number is calculated using its sequence number and channel label (Equation (3.11)). Then there are several operations which need to be executed:

1. If the packet’s calculated slot number is larger than the current slot number, the packet must wait.

2. If the packet’s calculated slot number is smaller than the current slot number, which means the packet has missed its local deadline, then we need to check the current slot. If it is empty, we send the packet immediately with the hope that it can catch its slot at the subsequent node. This approach is reasonable, since the current slot might be wasted anyway. If the current slot is occupied, the packet is dropped.
Note that this approach does not introduce any problems to the resource allocation algorithm, even though playback performance may be affected if many packets are dropped.

3.4 Performance Analysis

In this section, we analyze the impact of our channel allocation scheme on performance requirements: initial wait time, playback performance, unreserved bandwidth and fragmentation.

3.4.1 Initial Wait time

As mentioned in Section 2.2.2, the initial wait time is another important performance parameter for multimedia on-demand applications. Generally it is composed of:

1. The **channel establishing time**, which is the time required to establish a global channel.
2. Packet preparation time at the source.
3. The **maximum transmission time**, which is the maximum amount of time a packet can possibly spend within the network before arriving at the destination.
4. The **prefetch time**, which is the delivery time of all the data that should be prefetched before playback can start.
Below, we analyze the impact of our channel allocation scheme on the initial wait time.

Channel Establishing Time

For each new request, the required resources are calculated according to the performance requirements, then admission tests are used to ensure that, if the new request is granted, the performance of the new request and the existing ones will not be violated. These tests check processing capacity, deadlines, and buffer space, and can be very time-consuming in models such as UCB’s real-time channel.

In our model, the bandwidth of existing channels has been reserved, so when several packets from different channels arrive at the same time, there will be no conflict. Thus, bandwidth tests are not necessary. Also, the unused bandwidth is always in the form of slots which are uniformly distributed over time. When a logical channel is allocated to a request, packets will be sent regularly. In other words, the interval time between any two successive packets of a channel is the same, and each packet can reach its destination within a bounded delay. This eliminates the need for deadline tests. Our approach also avoids buffer space tests, since at every link, packets for the same request are sent at the same speed, so at the intermediate nodes, only one packet-sized buffer is needed to store a packet that must wait for its transmission slot.

In conclusion, if we can find and allocate a local logical channel at every link along the route, then we do not need to perform complex admission tests. This results in large savings in establishing time, and thus, packets can be sent as soon as the global logical channel is found.
Maximum Transmission Time

The maximum transmission time of a global channel is the sum of (1) the propagation time of every link along the global channel, and (2) the maximum relay time of a packet at every node along the global channel.

The relay time of a packet at a node is the period of time between when the first bit of the packet is about to be received from the preceding node and when it is about to be sent to the next node. Thus, \(\text{relay time} = \text{queueing delay} + \text{packet output time} + \text{packet handling time}\), where

- **Queueing delay** is the amount of time a packet spends in the output queue until its first bit is about to be output. Under our channel allocation scheme, when a packet belonging to a channel with bandwidth \(p/2^k\) arrives at a node, it has to wait for the next slot of the channel, and the maximum queueing delay at the link is \(2^k/p\). On average, its wait time is \(2^{k-1}/p\).

- **Packet output time** is the interval between when a packet’s first bit is about to be output to a link until its last bit has just been output. It depends solely on the hardware speed.

- **Packet handling time** consists of all the rest of the time, such as scheduling time, etc. Due to the simplicity of our traffic control and scheduling algorithm, it is assumed to be negligible. This may not be the case in other models.

Prefetch Time
If \( R_t \geq R_c \) then the prefetch time \( T_p \) is the delivery time of one frame, that is,

\[
T_p = \frac{f}{R_t}.
\]

If \( R_t < R_c \), the prefetch time is the time required to buffer the prefetched data \( D \), where

\[
D = U_{\text{max}} - f \cdot \frac{R_t}{R_c} = F - (F - 2 \cdot f) \cdot \frac{R_t}{R_c} - f \cdot \frac{R_t}{R_c} = F - (F - f) \cdot \frac{R_t}{R_c}. \tag{3.12}
\]

(See Equation (3.8).)

Therefore, the prefetch time \( T_p \) is:

\[
T_p = \frac{D}{R_t} = \frac{F}{R_t} - \frac{F - f}{R_c}.
\]

### 3.4.2 Playback Performance

Recall that, in our model, to support reasonable playback performance, a suitable transfer rate is the most important requirement. It requires enough bandwidth and that packets be transmitted within a bounded delay. In conventional networks, packets are sent in First-Come-First-Served (FCFS) order, and it is difficult to achieve a bounded delay. To support real-time requirements, Earliest-Due-Date-First (EDDF) schedulers can be used to order the transmission of packets according to their deadlines. But a complex deadline requirement test is needed during the channel establishing phase to ensure that all packets can arrive at their destinations before their deadlines, which can be quite time consuming. During the transmission phase, an EDDF scheduler can be used to select the next packet to be transmitted, which is
also time-consuming. Thus an EDDF scheduler is not an attractive solution for real-time communications.

In our model, the transmission slots of all packets for every channel are reserved at the time a logical channel is established. This not only ensures enough bandwidth for the transmission, but also provides a bounded delay without a complex admission test and complex scheduling algorithm. Thus, our channel allocation scheme provides good playback performance without large overhead compared to other models.

3.4.3 Unreserved Bandwidth

As in all other networks, the bandwidth in our scheme may not be fully utilized. It is possible that for some users, there is an excess of buffer space at their destinations, so the respective transfer rates can be higher than the assigned ones.

One way to make use of the unreserved bandwidth is to send more than one packet for every channel during the usual interval time between two successive packets. Of course, then it may be necessary to allocate more buffer space at the subsequent nodes, since it may not be possible to transmit these packets immediately at these nodes. The required buffer space calculation is as follows. Consider two successive nodes, \( N_1 \) and \( N_2 \), on a logical channel \( C \). Assume that the \( i \)th packet for \( C \) must have left node \( N_1 \) by time \( a_1 + b_1 i \), and node \( N_2 \) by \( a_2 + b_2 i \), (most likely, \( b_1 = b_2 = \) the required transfer rate), and let \( t \) be the current time, \( s \) the size of the packet, \( B_2 \) the buffer size reserved for \( C \) at node \( N_2 \), and \( T_t \) transmission time for a packet \( j + 1 \) from \( N_1 \) to \( N_2 \). Also assume that, at time \( t \), the first \( j \) packets have already left node \( N_1 \) (this information is maintained at node \( N_1 \)). The question we must answer
is: can we send the $j+1$st packet now without overflowing the buffer $B_2$ at node $N_2$?

First, observe that the current number of packets at node $N_2$ is at most $j - i_2$, where $i_2$ is the maximum integer satisfying $a_2 + b_2i_2 < t + T_t$. Then if $(j + 1 - i_2) \cdot s \leq B_2$, packet $j + 1$ can be transmitted, otherwise, the packet has to be held at $N_1$.

If we utilize unreserved bandwidth and buffer space at subsequent nodes, then the transmission time for the entire file is decreased.

### 3.4.4 Fragmentation

If a destination only has limited buffer space, the required transfer rate may be too low to utilize the entire bandwidth of a channel with bandwidth $p/2^k$. However, another channel, whose bandwidth is $p/2^{k+1}$, cannot satisfy the bandwidth requirement. In this case, if the channel with bandwidth $p/2^k$ is used, we may have to skip some slots, which wastes bandwidth. In the worst case, roughly $p/2^{k+1}$ bandwidth can be wasted when the required transfer rate is slightly higher than $p/2^{k+1}$.

One solution to this problem is to multiplex several logical channels to support the required transfer rate. Of course, to simplify channel management, we can specify a threshold to limit the number of multiplexed channels. Multiplexing several logical channels to support one request can also be used when a channel with the required transfer rate cannot be found but the bandwidth of the associated link is available.
3.5 Related Problems

In this section, we discuss other related problems for multimedia on-demand applications, which include request conflicts and routing algorithms.

3.5.1 Conflicting Requests

It is possible that several requests may want to reserve the same resources simultaneously, which could result in a deadlock or unnecessary rejections to requests. For example, in Figure 3.12, a request $r_1$ from Node C, which has reserved bandwidth of link B-C, wants to reserve bandwidth of link A-B. At the same time, another request $r_2$ from node A, which has reserved bandwidth of link A-B, wants to reserve bandwidth of link B-C. If the bandwidth of both links A-B and B-C can only support one request but not both, then there is a conflict. According to our original scheme, both requests will be rejected, even though one could be satisfied.

There are several ways to avoid this situation. Since most likely, multimedia on-demand systems will be supported by telephone or cable companies, the network
architecture of these companies is usually a chain or a tree-based, it is natural to modify the tree locking protocol as a solution. The tree locking protocol has advantages over some other locking protocols such as the two-phase locking protocol, in that it is deadlock-free, so that rollback is not needed. Also it may need less wait time and thus concurrency is increased.

The general rules of the tree locking protocol are:

- The first lock by a request $r_i$ can be on any node in the tree.
- Then, a node $N_1$ can be locked by $r_i$ only if the parent of $N_1$ is currently locked by $r_i$.
- A node may be unlocked at any time.
- The request $r_i$ cannot relock a node if it has been locked and unlocked by $r_i$.

In our model, all locks are exclusive. All file servers and network links are treated as one node in our locking tree, even though we may reserve different types of resources; i.e., on a file server, we reserve retrieval capacity and the associated buffer and on a link, we reserve bandwidth. A user always sends a request to the root of the tree, which is a multimedia file server (if there are several file servers, we select one of them as the root), and sets a lock on the node until either the required resources are reserved or the request is rejected. If the request is rejected, it frees all its locks. Otherwise, the request is sent to the subsequent node. A lock is then set on that node, and the lock on the parent node is released. If no enough resources are available at this node, the request is rejected or other routes could be tried. Otherwise, an allocation request is sent to the subsequent node. This proceeds until the request reaches the
destination or it is rejected. If a node is already locked by another request, the new request has to wait until the lock is freed. Thus, all requests are ordered and will no longer be unnecessarily rejected because of a deadlock.

This approach can also be used in an arbitrary network, provided the following condition holds: for any pair of nodes, $N_1$ and $N_2$, if a route is allowed which visits $N_1$ first then $N_2$, then all routes that involve both $N_1$ and $N_2$ visit $N_1$ first then $N_2$. Under this condition, locking nodes along a contemplated route will not cause a deadlock.

### 3.5.2 Routing Algorithm

In this section, we modify the shortest path routing algorithm, to account for the allocated channels and the transfer rate requirement of a request.

#### 3.5.2.1 Objective

To make better use of high-speed real-time networks and reduce congestion, an appropriate real-time routing algorithm is very important. A good routing algorithm should increase throughput and balance the traffic load. For real-time multimedia communications, other desired requirements include:

1. Obtaining routes sufficiently fast, which will affect the response time to the request.

2. Minimizing the new real-time channel’s effect on the links’ ability to guarantee QOS requirements for future channels.
3. Increase the possibility for the route to pass all the admission tests.

Besides many well-known routing techniques in conventional communications, a good real-time routing algorithm should also maximize the whole throughput of the on-demand system by minimizing the number of intermediate links through which the channel traverses. There are several reasons for this:

1. A large number of links involved in allocating a channel will increase the time required to establish the channel. And, to avoid request conflicts, associated links may have to be locked during channel allocation, which prevents other requests from requesting logical channels.

2. More links will also increase the probability that at some links, the bandwidth is not enough to support the required transfer rate, which makes it impossible for the route to pass the admission tests. This results in increased usage of the routing algorithm to find another route.

3. Non-real-time channels will also suffer worse performance if there are many links involved in the real-time channels, since such channels usually have higher priorities.

3.5.2.2 Shortest Path Routing

The shortest path routing algorithm can be used for selecting a route for a potential global channel. A directed graph can be created in which a node corresponds to a network node, and an edge to a network link. The cost function for an edge, in general, should be an decreasing function of its excess bandwidth. Namely, if the
excess capacity of a link is $p_e$ (packets/sec), then the cost for this link should be $d(p_e)$. The function should take the transfer rate requirement into consideration. For example, let $R_{t,\text{min}}$ (packets/sec) be the minimum transfer rate required. Then we define $d()$ as

$$d(p_e) = \infty, \text{ if } p_e < R_{t,\text{min}}$$

As mentioned before, the number of links involved in a route is also an important factor, which can be accomplished by having a parameter as a measure of the impact of the number of links in $d()$, and trying to minimize the cost of a route. We leave the exact form of $d()$ to the designer of the actual system.

Like the original Bellman-Ford algorithm [8], the shortest route can be found by finding the shortest path which consists of at most one link, then two links and so on. The algorithm proceeds as follows. Let $s$ be the source node, $h$ be the maximum number of links in a path at the current stage of the algorithm and

$$d_{ij}(p_e) = \text{link cost from node } i \text{ to node } j, \text{ in which } p_e \text{ is the excess capacity}$$

and $d_{ii} = 0$;

$$d_{ij}(p_e) = \infty, \text{ if the nodes are not directly connected or the available bandwidth cannot support the transfer rate requirement;}$$
$D^h_{st} = \text{cost of the least-cost path from node } s \text{ to } t \text{ with no more than } h \text{ links;}$

The following steps are repeated until the costs do not change:

1. Initialization:

   $$D^0_{st} = \infty, \text{ for all } t \neq s;$$

2. For every successive $h \geq 0$,

   $$D^{h+1}_{st} = \min_{i \neq t} \left\{ D^h_{si} + d_{it}(p_c) \right\}, \text{ for all } t \neq s,$$

where $D^{h+1}_{st}$ is the least-cost for a path that terminates with the link from node $i$ to $t$.

In this chapter, we have presented the channel allocation scheme and other related problems for fixed-size media. The next chapter focuses on the problems of variable-size media.
CHAPTER 4

Variable-Size Media

In this chapter, we will discuss the buffer underflow and overflow problems for variable-size media, from which we can derive bounds on the transfer rate.

As mentioned before, for variable-size media, we can no longer use a constant consumption rate \( R_c \) (packets/sec), since each frame may consist of different number of packets.

In this thesis, we assume that for every file, the minimum and maximum frame sizes, \( f_{\text{min}} \) and \( f_{\text{max}} \) (packets), where \( f_{\text{max}} > f_{\text{min}} \) are known. Let \( R^f_c \) (frames/sec) be the number of frames a playback unit consumes per second, which is assumed to be constant. Thus, the maximum consumption rate is given by \( R_{c,\text{max}} \) (packets/sec) = \( f_{\text{max}} \cdot R^f_c \), and the minimum consumption rate is given by \( R_{c,\text{min}} = f_{\text{min}} \cdot R^f_c \). We also assume that we know the number of frames, \( N \), a file consists of, in addition to the file length \( F \) (packets) and the buffer size \( B \) (packets). To avoid a discussion of trivial cases, we assume that \( F > B \).
CHAPTER 4. VARIABLE-SIZE MEDIA

4.1 Buffer Underflow and Overflow

For fixed-size media, once a transfer rate is fixed, we only need to deal with either the buffer underflow or overflow problem, depending on $R_t < R_c$ or $R_t > R_c$. For variable-size media, however, this may not be true.

We discuss the prefetch requirement first. Generally, there are two cases:

1. When $R_t \leq R_{c,\text{min}}$, the order of different sizes of frames appearing in the file will not affect the prefetch requirement. Since the playback of every frame requires some prefetched data, regardless of the frame sizes appearing we have in a file, we have to prefetch enough data to prepare for the worst case. Thus the buffer underflow problem of variable-size media can be translated to that of fixed-size media.

2. When $R_{c,\text{min}} < R_t < R_{c,\text{max}}$, we have to deal with both buffer underflow and overflow problems at the same time.

Below, we present the worst prefetch requirement when both buffer underflow and overflow problems exist.

**Theorem 1** Let $R_{c,\text{min}} < R_t < R_{c,\text{max}}$. If all, except possibly one, frames in a file have either size $f_{\text{min}}$ or $f_{\text{max}}$, all frames of size $f_{\text{max}}$ appear at the beginning of the file, and all frames of size $f_{\text{min}}$ appear at the end of the file, then we need the maximum amount of prefetched data to prevent buffer underflow.
Figure 4.1: Playback of frame $a$ followed by frame $b$ ($f_a \cdot R_c^f < R_t < f_b \cdot R_c^f$)

**Proof:** Figure 4.1 shows the amount of data in a client's buffer as a function of time. For the buffer underflow problem, the required buffer space is determined by the amount of required prefetched data $D$, which in turn depends on the difference between the transfer rate and the consumption rate and changes in a zigzag pattern.

Assume that, in Figure 4.1, frames $a$ and $b$ appear one after the other in the file. Time 1 is the start time of frame $a$'s playback, time 2 is when its playback finishes and is also the start time for frame $b$'s playback, and time 3 is when the playback of frame $b$ finishes. Since the same amount of data is buffered per second, (i.e., $R_t$ is constant), the height of 2-1' is the same as that of 3-2', which corresponds to the amount of data buffered during the playback of one frame. But a playback unit can consume a different amount of data per second, so the heights of 2-2' and 3-3', which correspond to the amount of data constituting for frames $a$ and $b$, respectively, are not the same. We can see that if the size of frame $a$, $f_a$ satisfies $f_a \cdot R_c^f < R_t$, then playback of frame $a$ consumes less data than was fetched. At the end of playback,
CHAPTER 4. VARIABLE-SIZE MEDIA

Figure 4.2: Playback of frame $b$ followed by frame $a$ ($f_a \cdot R_i^f < R_t < f_b \cdot R_i^f$)

the net amount of prefetched data has increased. Similarly, if the size of frame $b$, $f_b$ satisfies $f_b \cdot R_i^f > R_t$, at the end of playback, the net amount of prefetched data has decreased.

In Figure 4.1, frame $b$ requires prefetched data and frame $a$ does not. If we play back frame $a$ first, then the required prefetched data which is present will have increased when the playback of frame $a$ finishes. This will alleviate some of the prefetch requirement of frame $b$. (Note that if $f_a \cdot R_i^f \geq R_t$, then the change in the order will not affect the amount of the required prefetched data in the worst case.)

Figure 4.2 shows the effect when frame $b$ is played back before frame $a$. With less prefetch time prior to playback and a higher consumption rate, the amount of prefetched data remaining in the buffer is reduced. To avoid underflow, the amount of data which is prefetched prior to the start of playback must be increased.

Consider replacing frames $b$ and $a$ in Figure 4.2 with frames $b_1$ and $a_1$, respectively.
Figure 4.3: Frames $b$ and $a$ are replaced by frames $b_1$ and $a_1$, respectively. ($f_{a_1} \cdot R_c^f < R_t < f_{b_1} \cdot R_c^f$)

(Figure 4.3). If $f_a + f_b < f_{\text{max}} + f_{\text{min}}$, let $a_1$ have size $f_{\text{min}}$ and $b_1$ have size $f_{b_1} = f_a + f_b - f_{\text{min}}$. If $f_a + f_b \geq f_{\text{max}} + f_{\text{min}}$, let $b_1$ have size $f_{\text{max}}$ and $a_1$ have size $f_{a_1} = f_a + f_b - f_{\text{max}}$. In both cases, frame $b_1$ appears before frame $a_1$. Since frame $b_1$ needs more prefetched data than frame $b$ in both cases, it can only make the prefetch requirement worse.

With the above observations, we can pick two successive frames $a$ and $b$ in the file and change their order and/or replace them with frames $a_1$ and $b_1$ in the manner outlined to get a worse or the same prefetch requirement, until the condition of the theorem is satisfied. That is when all (except possibly one) frames in the file have either size $f_{\text{min}}$ or $f_{\text{max}}$, and all frames of size $f_{\text{max}}$ appear at the beginning of the file, all frames of size $f_{\text{min}}$ appear at the end of the file.

Now we discuss the worst case of the buffer overflow problem. Like the buffer underflow problem, we have two cases:
1. When $R_t \geq R_{c,\text{max}}$, unlike the prefetch requirement discussed before, the order of different frame sizes appearing in the file will affect the buffer space requirement. For example, if frames of size $f_{\text{max}}$ appear at the beginning of the file, we need a smaller buffer, since more data is consumed while the remaining packets are buffered.

2. When $R_{c,\text{min}} < R_t < R_{c,\text{max}}$, we have to deal with both buffer underflow and overflow problems at the same time.

However, in both cases, the condition of the worst requirement of a buffer overflow problem is the same. Using a method similar to that of Theorem 1, we can prove that, for both cases, if frames $a$ and $b$ appear one after the other in a file, and if we replace $a$ and $b$ with $a_1$ and $b_1$ in a similar way and let the frame $a_1$ (smaller size) appear before $b_1$, we have a worse or the same buffer requirement. Formally we have,

**Theorem 2** If all, except possibly one, frames in a file have either size $f_{\text{max}}$ or $f_{\text{min}}$, all frames of size $f_{\text{min}}$ appear at the beginning of the file, and all frames of size $f_{\text{max}}$ appear at the end of the file, then we need the largest buffer space to prevent the buffer overflow problem.

For variable-size media, we need to calculate the bounds on the transfer rate according to buffer underflow and/or overflow problems. Table 4.1 summarizes the worst case for buffer space requirement. In the next section, we discuss the bounds on the transfer rate.
CHAPTER 4. VARIABLE-SIZE MEDIA

<table>
<thead>
<tr>
<th>Value of $R_t$</th>
<th>Problem encountered</th>
<th>Worst case</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_t &lt; R_c$</td>
<td>Buffer underflow</td>
<td>largest frames first</td>
</tr>
<tr>
<td>$R_t &gt; R_c$</td>
<td>Buffer overflow</td>
<td>smallest frames first</td>
</tr>
</tbody>
</table>

Table 4.1: Problems for variable-size media

4.2 Bounds on the Transfer Rates

In this section, we calculate the maximum numbers of frames with size $f_{\text{min}}$ and $f_{\text{max}}$, respectively and then present the bounds of the transfer rate.

We divide the frames of a multimedia file into three sets, one consisting of frames with size $f_{\text{max}}$, a second consisting of frames with size $f_{\text{min}}$, and the last consisting of the remaining frames if any. So the maximum number $N_{\text{max}}$ of frames with size $f_{\text{max}}$ in a file containing $N$ frames should satisfy:

$$N_{\text{max}} \cdot f_{\text{max}} + (N - N_{\text{max}}) \cdot f_{\text{min}} \leq F,$$

That is,

$$N_{\text{max}} = \left\lfloor \frac{F - f_{\text{min}} \cdot N}{f_{\text{max}} - f_{\text{min}}} \right\rfloor.$$

Since when we maximize $N_{\text{max}}$, we also maximize the maximum number, $N_{\text{min}}$, of frames with size $f_{\text{min}}$. $N_{\text{min}}$ should satisfy:

$$N_{\text{min}} \cdot f_{\text{min}} + (N - N_{\text{min}}) \cdot f_{\text{max}} \leq F,$$

That is,

$$N_{\text{min}} = \left\lfloor \frac{N \cdot f_{\text{max}} - F}{f_{\text{max}} - f_{\text{min}}} \right\rfloor.$$
According to Equations (4.1) and (4.2), we get

\[
\frac{F - f_{\text{min}} \cdot N}{f_{\text{max}} - f_{\text{min}}} - 1 < N_{\text{max}} \leq \frac{F - f_{\text{min}} \cdot N}{f_{\text{max}} - f_{\text{min}}}.
\]

(4.1)

\[
\frac{f_{\text{max}} \cdot N - F}{f_{\text{max}} - f_{\text{min}}} - 1 < N_{\text{min}} \leq \frac{f_{\text{max}} \cdot N - F}{f_{\text{max}} - f_{\text{min}}}.
\]

(4.2)

According to Equations (4.1) and (4.2), we get

\[N - 2 < N_{\text{max}} + N_{\text{min}} \leq N.\]

Thus we have

\[N - 1 \leq N_{\text{max}} + N_{\text{min}} \leq N.\]

The frame, if any, whose size is neither \(f_{\text{min}}\) nor \(f_{\text{max}}\) has size:

\[F - f_{\text{min}} \cdot N_{\text{min}} - f_{\text{max}} \cdot N_{\text{max}}\]

To simplify the following formulae, we assume there is no such frame and \(N > 2\).

As for fixed-size media, immediately after a frame is played back, the next frame should be available. Thus, we require that

\[B \geq 2 \cdot f_{\text{max}}.\]

We divide the range of \(R_t\) into three parts:
1. Region A: \( R_t \leq R_{c,\text{min}} \) (only buffer underflow is possible).

2. Region B: \( R_{e,\text{min}} < R_t < R_{c,\text{max}} \) (both buffer underflow and overflow are possible).

3. Region C: \( R_t \geq R_{c,\text{max}} \) (only buffer overflow is possible).

Below we calculate the bounds on \( R_t \) in these three regions.

(1) Region A \( (R_t \leq R_{e,\text{min}}) \):

Since only buffer underflow is possible in this region, we only need to calculate the minimum transfer rate \( R_{t,\text{min}} \). The method used for fixed-size media is applicable here. Note that \( (N - 2)/R_c^f \) is the consumption time of all frames except the first and the last. In place of Equation (3.2), we have:

\[
\frac{N - 2}{R_c^f} \geq \frac{F - B}{R_t}.
\]

We thus have:

\[
R_{t,\text{min}} = R_c^f \cdot \frac{F - B}{N - 2}.
\]

The above inequality can be rewritten as:

\[
B \geq F - \frac{N - 2}{R_c^f} \cdot R_t. \tag{4.3}
\]

If \( R_{t,\text{min}} > R_{c,\text{min}} \) (i.e., \( (F - B)/(N - 2) \geq f_{\text{min}} \)), we cannot select a transfer rate in region A. Otherwise, we can select any \( R_t \) such that \( R_{t,\text{min}} \leq R_t \leq R_{c,\text{min}} \). (See Region A in Figure 4.5 for an example which illustrates Inequality (4.3).)
(2) Region C ($R_t \geq R_c,ma$):

When $R_t \geq R_c,ma$, we only need to deal with the buffer overflow problem. As in Section 3.2.1, we can calculate the required buffer space $B$ as:

$$B \geq \text{space for the initial data}$$

+ space for data buffered between time $S$ and time $C$

− space for data consumed between time $S$ and time $C$

if the amount of data in $l$ is less than the amount of data buffered after the playback of $l$ finishes,

+ space for data buffered after the playback of $l$ finishes

else

+ space for the last frame $l$,

$$B \geq \text{space for the initial data} + \text{space for data buffered between time } S \text{ and time } C - \text{space for data consumed between time } S \text{ and time } C$$

if the amount of data in $l$ is less than the amount of data buffered after the playback of $l$ finishes,

+ space for data buffered after the playback of $l$ finishes

else

+ space for the last frame $l$, \hspace{1cm} (4.4)

where frame $l$ is the last frame that finishes its playback before or at when the transmission finishes. (Refer to Figures 3.2 and 3.3 for the meaning of time $S$ and $C$.)

Since the initial data may not have the size of one frame and frames may have different sizes, we get from Inequality (4.4):

$$B \geq f_{init} + \left[ \frac{F - f_{init}}{R_t} \cdot R_c^f \right] \cdot \left( \frac{R_t}{R_c^f} - f' \right) + \max \left\{ f_l, F - f_{init} - \left[ \frac{F - f_{init}}{R_t} \cdot R_c^f \right] \cdot \frac{R_t}{R_c^f} \right\}$$

\hspace{1cm} (4.5)

where $f_{init}$ (packets) is the number of packets in the initial data, $f_l$ is the size of frame $l$, $f'$ (packets) is the average number of packets in a frame consumed between time $S$ and time $C$, and $\left\lfloor ((F - f_{init})/R_t) \cdot R_c^f \right\rfloor$ is number of frames which can finish
their playback during the transmission period. So \[ \left\lfloor \left( \frac{F - f_{\text{init}}}{R_t} \right) \cdot R_c^f \right\rfloor \cdot R_t / R_c^f \] is the amount of data (packets) buffered between time \( S \) and time \( C \).

According to Theorem 2, to prevent buffer overflow, we need to assume that all frames of size \( f_{\text{min}} \) appear at the beginning of the file (we call it first part) and frames of size \( f_{\text{max}} \) appear at the end (we call it second part). Since we have no idea whether the size of the first frame is \( f_{\text{max}} \) or not in a real file, we need to buffer \( f_{\text{max}} \) packets of data before playback can begin. Depending on the transfer rate, there are two cases to consider:

1. Transmission finishes before the first frame of the second part of the file finishes its playback.

2. Transmission finishes after the first frame of the second part finishes its playback.

\[ \frac{F - f_{\text{max}}}{R_t} \] is the entire transmission time after playback starts and \( \frac{(N_{\text{min}} + 1)}{R_c^f} \) is the entire playback time for the first \( N + 1 \) frames. Therefore, in the first case, \( R_t \) should satisfy:

\[
\frac{F - f_{\text{max}}}{R_t} \leq \frac{N_{\text{min}} + 1}{R_c^f},
\]

So let

\[
\bar{R}_t = \frac{F - f_{\text{max}}}{N_{\text{min}} + 1} \cdot R_c^f.
\]

Since we are in Region C, when \( R_t \geq \max \{ \bar{R}_t, R_{c,\text{max}} \} \), we have the first case, and when \( R_{c,\text{max}} \leq R_t \leq \bar{R}_t \), we have the second case. Note that if \( \bar{R}_t < R_{c,\text{max}} \), the second case is impossible.
In the first case, according to Inequality (4.5), \( R_t \) should also satisfy:

\[
B \geq f_{\text{max}} + \left[ \frac{F - f_{\text{max}}}{R_t} \cdot R_c^f \right] \cdot \left( \frac{R_t}{R_c^f} - f_{\text{min}} \right) + \max \left\{ f_{\text{min}}, F - f_{\text{max}} - \left[ \frac{F - f_{\text{max}}}{R_t} \cdot R_c^f \right] \cdot \frac{R_t}{R_c^f} \right\}.
\] (4.6)

\( R_{t,\text{max}} \) is the maximum transfer rate obtained from Inequality (4.6). (See Section 3.2.1). If \( R_{t,\text{max}} \geq \max \{ \bar{R}_t, R_{c,\text{max}} \} \), \( R_t \) can be any value between \( R_{t,\text{max}} \) and \( \max \{ \bar{R}_t, R_{c,\text{max}} \} \).

In the second case, the transmission finishes when all the frames of size \( f_{\text{min}} \) and some frames of size \( f_{\text{max}} \) finish their playback. The size of the initial data is \( f_{\text{max}} \), \( N_{\text{min}} \cdot \frac{R_t}{R_c^f} \) is the amount of the data buffered during the playback period of the first part of the file, while \( N_{\text{min}} \cdot f_{\text{min}} \) is the amount of data for the first part of the file. \( \left( \frac{(F - f_{\text{max}})}{R_t} \right) R_c^f \) is the number of frames played back during the whole transmission time, and thus \( \left| \frac{(F - f_{\text{max}})}{R_t} \cdot R_c^f - N_{\text{min}} \right| \) is the number of frames with size \( f_{\text{max}} \) which have finished their playback when the transmission is completed.

Using Inequality (4.5), again, we have:

\[
B \geq f_{\text{max}} + N_{\text{min}} \cdot \left( \frac{R_t}{R_c^f} - f_{\text{min}} \right) + \left[ \frac{F - f_{\text{max}}}{R_t} \cdot R_c^f - N_{\text{min}} \right] \cdot \left( \frac{R_t}{R_c^f} - f_{\text{max}} \right) + \max \left\{ f_{\text{max}}, F - f_{\text{max}} - N_{\text{min}} \cdot \left( \frac{R_t}{R_c^f} - f_{\text{min}} \right) - \left[ \frac{F - f_{\text{max}}}{R_t} \cdot R_c^f - N_{\text{min}} \right] \cdot \frac{R_t}{R_c^f} \right\} = f_{\text{max}} + N_{\text{min}} \cdot (f_{\text{max}} - f_{\text{min}}) + \left[ \frac{F - f_{\text{max}}}{R_t} \cdot R_c^f \right] \cdot \left( \frac{R_t}{R_c^f} \right) + \max \left\{ f_{\text{max}}, F - f_{\text{max}} + N_{\text{min}} \cdot f_{\text{min}} - \left[ \frac{F - f_{\text{max}}}{R_t} \cdot R_c^f \right] \cdot \frac{R_t}{R_c^f} \right\}.
\] (4.7)

\( R_{t,\text{max}} \) is the maximum transfer rate obtained from Inequality (4.7). (See Section 3.2.1.) If \( R_{c,\text{max}} \leq R_{t,\text{max}} \leq \bar{R}_t \), \( R_t \) can be any value between \( R_{c,\text{max}} \) and \( R_{t,\text{max}} \).
(3) Region B ($R_{c,min} < R_t < R_{c,max}$):

In this region, we need to consider both buffer underflow and overflow problems. Since we have no idea where the consumption rate exceeds the transfer rate in a real file, we need to prefetch enough data, $D$ (packets), before the playback begins.

According to Theorem 1, we assume that the file consists of two parts: all the frames of the first part have size $f_{max}$ and all the frames of the second have size $f_{min}$. Since we are in Region B, only the first part needs prefetched data, where $f_{max} \cdot N_{max}$ is the size of the first part and the consumption rate is $R_{c,max}$. Thus the required prefetched data is:

$$D = f_{max} \cdot N_{max} - (f_{max} \cdot N_{max} - f_{max}) \cdot \frac{R_t}{R_{c,max}} = f_{max} \cdot N_{max} - (N_{max} - 1) \cdot \frac{R_t}{R_c}.$$  

(See Equation (3.12).)

If $B < D$, then the client does not have enough buffer space and a transfer rate cannot be selected in this region. Otherwise, playback should begin when the required prefetched data $D$ is buffered.

Now we can calculate $R_{c,max}$, in which there is a buffer overflow problem. According to Theorem 2, to calculate $R_{t,max}$, we assume that the file consists of two parts: all the frames of the first part have size $f_{min}$ and all the frame of the second have size $f_{max}$. Note that since $R_{c,max} \geq R_t$ (i.e., $R_t/R_c \leq f_{max}$), we have $D \geq f_{max}$. Playback should begin when the required prefetched data $D$ is buffered.

Though there is a buffer overflow problem for the first part of the file, we cannot use the same method as used for fixed-size media to calculate the maximum transfer
rate. That is because transmission finishes only after the entire file has been buffered, not just the first part. In general, there are two possible cases:

1. Transmission finishes before the first frame of the second part of the file is completely played back.

2. Transmission finishes after the first frame of the second part of the file is completely played back.

\((F - D)/R_t\) is the entire transmission time after the playback starts and \((N_{\text{min}} + 1)/R_c^f\) is the entire playback time for the first \(N_{\text{min}} + 1\) frames. Therefore, in the first case, \(R_t\) should satisfy:

\[
\frac{F - D}{R_t} \leq \frac{N_{\text{min}} + 1}{R_c^f},
\]

or

\[
\frac{F - f_{\text{max}} \cdot N_{\text{max}}}{R_t} \leq \frac{N_{\text{min}} - N_{\text{max}} + 2}{R_c^f}.
\]

Let

\[
\tilde{R}_t = \frac{N_{\text{min}}}{N_{\text{min}} - N_{\text{max}} + 2} \cdot R_{c,\text{min}}, \quad \text{if } N_{\text{min}} - N_{\text{max}} + 2 > 0,
\]

which is the minimum value of \(R_t\) obtained from the above inequality.

When \(N_{\text{min}} - N_{\text{max}} + 2 > 0\) and \(\max \{\tilde{R}_t, R_{c,\text{min}}\} \leq R_t \leq R_{c,\text{max}}\), we have the first case. When \(N_{\text{min}} - N_{\text{max}} + 2 \leq 0\) and \(R_{c,\text{min}} < R_t < R_{c,\text{max}}\), or when \(N_{\text{min}} - N_{\text{max}} + 2 > 0\) and \(R_{c,\text{min}} \leq R_t \leq \min \{R_{c,\text{max}}, \tilde{R}_t\}\), we have the second case.
In the first case, we buffer the whole file during the prefetch time and all or part of the playback time of the first part of the file. By Inequality (4.4), \( R_t \) should also satisfy:

\[
B \geq D + \left[ \frac{F - D}{R_t} \cdot \frac{f^I_c}{R^I_e} \right] \cdot \left( \frac{R_t}{R^I_e} - f_{\text{min}} \right) + \max \left\{ f_{\text{min}}, F - D - \left[ \frac{F - D}{R_t} \cdot \frac{f^I_c}{R^I_e} \right] \cdot \frac{R_t}{R^I_e} \right\}.
\]

(4.8)

Let \( R_{t,\text{max}} \) be the maximum transfer rate obtained from Inequality (4.8). (See Section 3.2.1). If \( \max \{ R_t, R_{c,\text{min}} \} \leq R_{t,\text{max}} \leq R_{c,\text{max}} \), \( R_t \) can be any value between \( \max \{ R_{c,\text{min}}, \tilde{R}_t \} \) and \( R_{c,\text{max}} \).

In the second case, we know that during the playback period of the first part of the file, the buffer space requirement always increases until the last frame of \( f_{\text{min}} \) finishes its playback. After that the buffer space requirement always decreases. This is illustrated in Figure 4.4. Transmission starts at time \( A \), playback starts at time \( S \), playback of the first part of the file finishes at time \( C \), playback of the first frame with size \( f_{\text{max}} \) finishes at time \( H \), transmission completes at time \( L \) and playback ends at time \( F \). \( QS \) corresponds to the buffered packets for the initial data, which is \( D \). The maximum buffer space is required at the time just before playback of the first frame with size \( f_{\text{max}} \) completes. The number of packets consumed between time \( S \) and time \( C \) is \( f_{\text{min}} \cdot N_{\text{min}} \), and the number of packets buffered during that period is \( (N_{\text{min}}/R^I_e) \cdot R_t \). The packets buffered between time \( C \) to time \( H \) is \( R_t/R^I_e \). Thus, the buffer space requirement can be calculated as:

\[
B \geq D + N_{\text{min}} \cdot \left( \frac{R_t}{R^I_e} - f_{\text{min}} \right) + \frac{R_t}{R^I_e}.
\]

(4.9)
Figure 4.4: Calculation of maximum transfer rate (Case 2: when transmission finishes after the first frame of the second part of the file finishes its playback).
Let $R_{t,max}2$ be the maximum transfer rate obtained from Inequality (4.9). If $N_{min} - N_{max} + 2 > 0$ and $R_{c,min} \leq R_{t,max}2 \leq \min \{R_{c,max}, \tilde{R}_t\}$, or if $N_{min} - N_{max} + 2 \leq 0$ and $R_{c,min} \leq R_{t,max}2 \leq R_{c,max}$, then $R_t$ can be any value between $R_{c,min}$ and $R_{c,max}$.

**Example 3:** Figure 4.5 illustrates the relationship between $U_{max}/f_{min}$ and $R_t/R_{c,min}$. (Since $R_c$ is variable for variable-size media, we normalize $R_t$ with respect to $R_{c,min}$.) The parameters used are $F = 250$ (packets), $N_{max} = 6$ and $N_{min} = 13$, $f_{max} = 20$, and $f_{min} = 10$ (packets). For these data, we have $\tilde{R}_t = 13/9$ and $\tilde{R}_t = 23/13$. From Figure 4.5, we can see that when $R_t \leq R_{c,min}$ (region A), there is a decreasing linear region, which corresponds to Inequality (4.3). When $R_{c,min} \leq R_t \leq R_{t,max}$ (region B), there are two subregions. An increasing linear subregion corresponds to Inequality (4.9), which is the second case in region B. The other subregion corresponds to Inequality (4.8), which is the first case in region B. These two subregions meet when $R_t = \tilde{R}_t$. Since $\tilde{R}_t < R_{c,max}$ for the specific values used in this example, only the first case is possible in region C, which corresponds to Inequality (4.6). If $\tilde{R}_t > R_{c,max}$, there would be two subregions which corresponded to Inequalities (4.6) and (4.7), respectively. We can also see that, the minimum buffer space is required, when $R_t = R_{c,min}$. For any fixed $U_{max}$, which is bigger than the minimum buffer space required, there are two transfer rates, $R_{t,min}$ and $R_{t,max}$.

**Example 4:** Figure 4.6 was generated based on the parameters of a simplified real video program [21]. We assume that (1) every screen of the video has $320 \cdot 240$ pixels, each consisting of 24 bits, (2) 30 frames are played back per second with an “average” compression ratio 5000, (3) $N_{max}/N_{min} = 2/15$ and $f_{max}/f_{min} = 20/1$, and (4) the total playback time is 60 minutes. Figure 4.6 shows that the minimum buffer space requirement is about 10k when $R_t = R_{c,min}$. And it will increase as $R_t$ deviates
Figure 4.5: $U_{\text{max}}/f_{\text{min}}$ vs. $R_t/R_{c,\text{min}}$. (Maple was used to draw this figure.)
Figure 4.6: $U_{max}$ vs. $R_t$. (*Maple* was used to draw this figure.)
CHAPTER 4. VARIABLE-SIZE MEDIA

from $R_{c,\text{min}}$. 
CHAPTER 5

Summary and Conclusions

In this chapter, we present a brief review of this thesis and discuss future research.

5.1 Review

In this thesis, we have developed a reservation-based channel allocation scheme and the associated control mechanism to support multimedia on-demand applications.

The major contributions of this thesis are:

1. The development of a logical channel allocation scheme, which satisfies different real-time bandwidth requirements of multimedia on-demand applications. Because of its simplicity, our channel allocation scheme avoids complex admission tests and thus decreases the channel establishment time. And since neither any special connectivity of networks nor any special types of media is assumed in
CHAPTER 5. SUMMARY AND CONCLUSIONS

our model, our approach is quite general. It can be used for different types of multimedia on-demand applications, such as education, entertainment, real estate, etc., through arbitrary connected networks, such as tree-like and ring-based networks with different kinds of media, such as audio, video, text, etc.

2. The development of the associated control mechanism as a traffic controller and transmission scheduler. Our mechanism is very simple and thus satisfies the real-time requirement for high-speed network systems.

3. We have also studied many related problems. The principle of our analysis on initial wait time and playback performance could be used for other types of multimedia applications. Our routing algorithm can take the hop-count of a route into consideration, which makes it more suitable for real-time communications.

4. We have studied the buffer underflow and overflow problems for both fixed-size and variable-size media and have presented the formulae for calculating the minimum and maximum transfer rates as well. Our results on the buffer underflow and overflow problems are not constrained to multimedia on-demand applications. The basic principles for calculating the minimum and maximum transfer rates and retrieving delay-sensitive data should be useful for designing any kind of multimedia systems. Since our results cover both fixed-size and variable-size media, they could be used for many kinds of media.
5.2 Future research

Since the area of multimedia on-demand applications is still relatively new, there are many problems which need further work.

A common problem for predictive schemes is that this type of resource management is bound to use the network resources less than an adaptive approach. According to monitored results [30], an adaptive scheme can achieve almost full utilization of network resources. But a predictive scheme is based on conservative estimates and has to prepare for the worst case, which may not happen in an actual application. To make better use of resources, we need to study how to use unreserved bandwidth and the fragmentation problem.

Despite the advantages of our logical channel allocation scheme, whether this model can work efficiently in a real system is still a question. To precisely compare the performance of our scheme with that of other models mentioned in Section 1.3, simulations and experiments need to be done. Since our model is designed for multimedia on-demand systems, its modification to support other types of multimedia applications, such as interactive multimedia communications, or to co-exist with other types of network protocols such as TCP/IP, is a very challenging and important task.

Another interesting problem is how to use the relationship between file size, buffer space and transfer rates in an actual on-demand system. This relationship can be used by system designers to select and evaluate design strategies. More lessons need to be learned from real on-demand systems before we can completely understand this relationship and develop a cost-effective system accordingly.
Routing is another interesting and important topic for multimedia communications. As mentioned in Section 3.5.2.2, more study is needed to find a suitable cost function \( d() \) according to the special requirements of real-time multimedia communications. Dynamic routing and fault tolerance issues should also be added to our model to increase the reliability and availability of multimedia on-demand systems. Multicast and broadcast are also very important for multimedia communications, which should be added to our model in the future.

Our model may serve as a good platform for these and other research problems.
REFERENCES


REFERENCES


REFERENCES


