Reservation Based Admission Control of Multimedia System

by

Md. Mujibur Rahman

A Thesis Submitted to the Department of Computer Science and Engineering in Partial Fulfillment of the Requirement for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

(Computer Science and Engineering)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DHAKA-1000, BANGLADESH
NOVEMBER, 2005

List of Figures

List of Tables

Chapter 1
Introduction

Rapid advances in communication, high speed network, media compression, and processor and memory design made multimedia computing infrastructure feasible. Exciting interactive applications such as multimedia mail, orchestrated presentations, high quality on-demand audio and video, collaborative multimedia document editing, large digital libraries or multimedia archives, and virtual reality became possible in this environment. A user may want to access these applications from a variety of end systems such as personal computer at home, a Personal Digital Assistant (PDA), a set top box with a television set, or a workstation in the workplace. Now-a-days users are ready to pay for these multimedia services like video on demand (VoD) if the Quality of Service (QoS) is guaranteed. Such guarantees can be provided by performing admission control and resource reservation.

Multimedia service providers can be able to make good business if some issues related to the services are resolved that are required to satisfy the customers with guaranteed Quality of Service (QoS). These issues are as follows -

- What are the resources to be reserved for a particular multimedia session?
- How the resources will be reserved?
- How does conventional reservation scheme be applied?
- What will be the main policy of admission controller?

Our research addresses these issues for a single multimedia server by presenting a reservation based admission controller for a server system.

1.1 Motivation

Commercially attractive multimedia services like Video on Demand (VoD), high quality video conferencing, e-learning, WebTV, large digital libraries or multimedia archives, and virtual reality environments etc require guaranteed QoS to satisfy the users who are willing to pay certain amount of money for a particular multimedia service. Multimedia services deal with voluminous data (e.g., audio, video, images, etc) that require a high-end server to serve the customers. The Multimedia server needs *resources* (i.e., CPU cycles, memory, bandwidth etc.) for their operation in order to provide services for the multimedia applications [1]. But the resources or multimedia server are finite. It is difficult to ensure guaranteed QoS of these multimedia applications with the finite resource in different service providers such as server, network and client. Therefore some form of admission control is necessary to reject some of the customers sessions when insufficient resources are available to serve all of them. Maximization of revenue by admitting profitable sessions is an important objective of the admission controller.

Delivery of multimedia streams with absolute guarantees of QoS from a single multimedia server has been proposed by Khan [2, 3]. When a customer requests a new multimedia session, the customer first specifics the QoS requirement (QoS specification). The user's specification is then translated to system and network level QoS parameters (QoS translation). The resource managers at the end-systems and network nodes determine whether sufficient resources are available to support the specified quality. If sufficient resources are available, the session is accepted, and the system reserves the required end-system and network resources to provide service according to the QoS specification. If the available resources cannot support the specified quality, the system may either reject the request or suggest a compromise of quality.

Admission control and QoS adaptation is done on batches of sessions in a regular time interval which is called an epoch [4]. In each epoch some of the old sessions leave and some new session requests are batched for admission. Thus there is a change of resource usage in the system.

In the conventional admission control of multimedia server there is no provision of reserving resource for the future. But when the resources of Multimedia server are over-subscribed especially in the peak load hours, there is a chance of rejection of some requests if a user submits a request to start her movie session instantly. In that case busy users may be interested to make reservations days or even weeks before the starting time of the session. The session might start in future. But the users can reserve the resource in order to get admitted in the system at a particular time in future.

1.2 Previous Works and Problem Definition

Admission controlling is done in order to ensure the Guaranteed Quality of Service for multimedia services. Guaranteed Quality of Service for multimedia application requires end-to-end guarantees, covering the server, network and clients. More precisely,

- The multimedia server must be able to deliver multimedia stream to all admitted customers with guaranteed QoS. Sufficient server resources (i.e. CPU cycles, memory, bandwidth etc.) should be reserved for each multimedia session.
- The customer's machine must be able to play the multimedia stream with QoS.
- The connection of sufficiently high bandwidth is required from server to customer.

The admission controller admits the customers into the system depending upon the availability of resources; otherwise the user will face delay and jitter while enjoying multimedia services due to the contention of resources.

The Utility Model, proposed by Khan [2, 3] demonstrates the admission control and QoS adaptation principles of Adaptive Multimedia System (AMS) where the utility (revenue earned) is maximized for a single server multimedia service provider. This problem can be mapped to a Multi-dimensional Multiple-choice Knapsack Problem (MMKP) which is an NP-hard problem [5]. I-HEU, an incremental heuristic to solve the MMKP, is used to do real time admission control [3].

The above mentioned admission controlling strategies do not address reservation issues. Our research addresses the resource reservation scheme and profit (revenue) maximization by the system. The customers submit the session requests for admission. The admitted session might be started right away or later at a particular time. The requests are batched over an epoch. Batches of requests are submitted to the admission controller for consideration. The Admission controller keeps the record of available resources and it runs the admission controlling algorithms for determining the profitable sessions to be admitted. It also keeps the record of resources that will be available in future and revenue earned from the accepted sessions.

1.3 Scope and Focus

The main focus of this thesis is to present a reservation based admission control of a multimedia system. The multimedia services become feasible due to the availability of a number of enabling technologies, such as MPEG (Motion Picture Expert Group), ATM (Asynchronous Transfer Mode), ADSL (Asymmetric Digital Subscribers Line), HFC (Hybrid Fiber Coax) with the maturity of two physical and medium access control protocol standards, i.e IEEE 802.14 and MCNS (Multimedia Cable Network System) [6].

The research will be applicable in order to support the system that will provide the multimedia services. In our thesis we consider Video-on-demand (VoD) service as an example of multimedia service. We describe a practical system for reservation based admission control. But the implementation detail is not described as implementation is not our goal. We present algorithms, the complexities of the algorithms. The exact algorithm is out of scope because our smart focus is real time admission control.

To analyze the performance of the reservation based admission controller of a multimedia server, we developed a discrete event simulation of the admission controller. We simulated the proposed system in Visual C++.

1.4 Outline

This thesis is organized in 5 chapters. Chapter 1 presents the introduction and objectives of the problem. Chapter 2 of this thesis presents the preliminary details and literature review. Chapter 3 presents the mathematical formulation of the problem. The heuristic to do admission control will be presented in Chapter 4. In Chapter 5 we mapped our problem into the MMKP (Multiple choices Multidimensional Knapsack Problem) and we present a solution of that MMKP.

Chapter 6 presents the simulation of the proposed admission controller with experimental results. Obtained results are presented in graphs and analyzed.

Chapter 7 concludes the paper with future research plan and major contributions.

Chapter 2

Preliminary Details and Literature Review

2.1 Multimedia

The term "**Multimedia**" is used to indicate that the information/data being transferred over the network may be composed of one or more of text, images or graphics, audio, video etc. media types [6]. A multimedia system is characterized by computer-controlled, integrated production, manipulation, presentation, storage and communication of independent information, which is encoded at least through a continuous (time-dependent) and a discrete (time-independent) media [7].

Discrete media are composed of time-independent information units. That is, there is no temporal relationship between successive units of information from the same media. *Text*, *Graphics* and *images* are discrete media. Continuous media, such as *audio* and *video*, requires continuous play out as time passes. In other words, *time* dependency between information units is part of information itself in continuous media.

2.2 Multimedia Server System (MSS)

In a multimedia system, capturing even a small amount of multimedia content requires a large amount of data. Typically, this data is stored in a server or storage system from which the data is streamed continuously to the user's (client) display device at a given playback rate. Therefore, delivery of multimedia data is time-sensitive; that is, users (clients) will notice glitches if audio or video data is not delivered on time. This implies that management of every component of Multimedia Server System must consider the time criticality of the data. Moreover, the communication network that interconnects the server(s) to the users (clients) requires more Bandwidths compare to other conventional networks.

Figure 2-1 illustrates a Multimedia Server Systems (MSS) that has the following three main components such as -

- Server(s)
- Users (Clients)
- Communication Network.

The server(s) is an active element in the Multimedia Server which provides a predefined set of services that are accessible through the communication network. The main function of a multimedia server is to provide the continuous delivery of data (multimedia streams) in real time to the users (clients) [8]. The users (clients) enjoy the multimedia streams from the multimedia server(s) through the communication network. Users pay money for the services that they enjoy. Hence, the guranteed Quality of Services (QoS) of multimedia must be ensured. The basic function of the network infrastructure is to connect the many different users (clients) to the servers directly or indirectly.

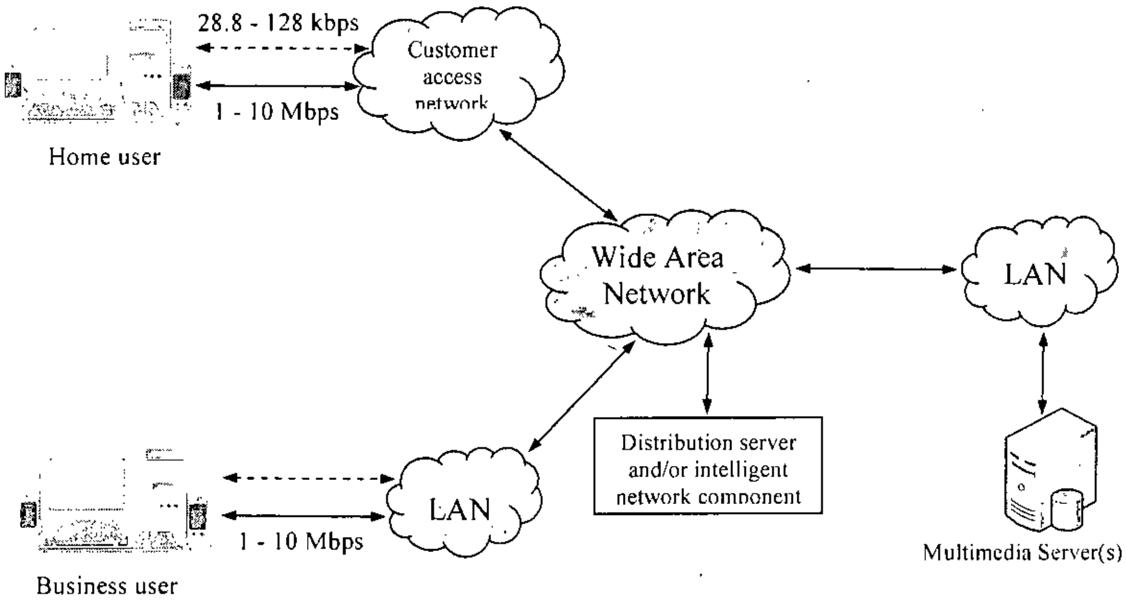


Figure 2-1: Multimedia Communication Network

After the basic requirement of connectivity is satisfied, clients must be able to retrieve the multimedia streams from the server(s) with satisfactory performance (i.e. QoS) [9]. The requirements of multimedia communications and networks are described in the later section.

2.3 Quality of Service (QoS)

The QoS may be expressed using a set of qualitative or quantitive parameters, where each parameter relates to one property of service [10]. For example, the QoS perceived by a user of a session depends on the user's subjective evaluation. The user may express her perception using expressions such as good, ok, bad, choppy but understandable. Technically QoS may be expressed using parameters such as network Bandwidth, end-to-end delay, jitter, synchronization, video resolution, audio quality, percentage loss of packets, etc.

2.3.1 End to End Delay

A disturbance in synchronization of multimedia streams is simply a delay suffered by one (or more) of the streams during transmission. The end-to-end delay is the total delay for a data unit to be transmitted from the source to its destination. For example, the source of a video telephone is the camera; the destination is the video window on the screen of the partner.

2.3.2 Delay Jitter

The delay may not be a constant for a given stream across the multimedia communication network because of the highly dynamic nature of the network (and the traffic handled by it). That means the receiver will experience not only a delay but also a variation of the delay. The variation of the delay is called jitter.

2.3.3 Resolution of Video

The image sharpness of a display screen is called its resolution; the more pixels there are per square inch, the finer the level of detail attained. Resolution is expressed in terms of the formula horizontal pixel × vertical pixels. Each pixel can be assigned a color or a particular shade of gray. Thus a screen with 640×480 pixels multiplied together equals 307,200 pixels. Some display screens are 800×600 or 1024×768.

2.3.4 Frame Rate

Refresh rate is the number of times per second that the pixels are recharged so that their glow remains bright. Refreshing is necessary because the phosphors hold their glow for just a fraction of a second. The higher the refresh rate, the more solid the image looks on the screen, that is, the image doesn't flicker. Refresh rate is called frame rate. Generally frame rate is 30 frames per second, but it may vary depending upon the system that is used.

2.4 Audio Quality

Sound is a continuous wave that travels through the air. Human ears can hear in the range of 20 Hz to about 20 KHz. This range of sound is called the audio. Audio in the natural world is analog. But audio stored on a computer is digital. The process of converting natural analog audio into discrete digital audio is digitization. Digitization of analog audio is composed of two phases: sampling; and quantization.

Audio quality can be defined as the degree of accuracy with which a device records or emits the original audio waves. This accuracy depends on the digitization (sampling and quantization) and compression technique which are described in later sections.

2.4.1 Sampling

To sample a signal means to examine it at some point in time, as shown in Figure 2-2. Sampling usually happens at equally separated intervals; this interval is called the sampling interval. The reciprocal of sampling interval is called the sampling frequency or sampling rate. The unit of sampling interval is second. The unit of sampling rate is Hz, which means cycles per second.

In Figure 2-2, the sampling interval is 1 μsec (10^{-6} sec), or in other words the sampling frequency is 1 MHz (10^6 Hz). This means that the ADC samples this sine wave every 1 μsec. If the analog signal has the highest frequency of *f*; to reconstruct the original analog signal faithfully; the sampling rate must be at least 2*f*. This is also called the sampling theorem or Nyquist sampling theorem. In Figure 2-2, the frequency of this sine wave is 2×10^4 Hz, and the sampling frequency is 10^6 Hz, which is much greater than $2\times2\times10^4$ Hz. So this sine wave will be faithfully reconstructed back to an analog signal.

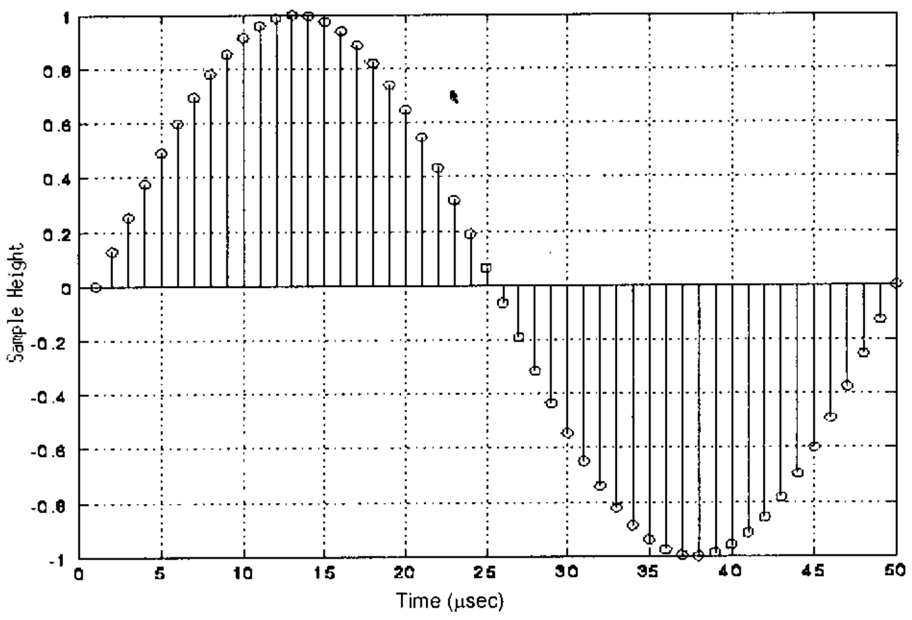


Figure 2-2 : Sampled waveform.

2.4.2 Quantization

The process of converting a sampled sound into a digital value is termed as quantization. The number of distinct sound levels that can be represented is determined by the number of bytes used to store the quantization value. CD audio, the most common quantization strategy uses 2 bytes (16 bits); capable of representing 65,536 discrete levels. In simple terms, quantization can be viewed as converting real (continuous sound), values into integer (discrete sound) values. This process involves dealing with the error between the sampled discrete values and the actual continuous sound, termed *quantization error*.

2.4.3 Audio Compression

Computers can use different schemes to eliminate repetitive information when storing data. In case of recorded audio, compression schemes are applied to reduce the storage size and eliminate redundant data. Compression can be lossy, meaning the sound quality will be negatively affected by compression, or lossless, meaning there will be no change in sound quality. Common techniques, such as MPEG, MACE, and ADPCM compression, are lossy, as they provide a great deal of space savings at a reasonable cost in quality [7].

Audio data does not easily yield high compression rates when standard compression methods are employed. The CD Audio standard (***P***luse ***C***ode ***M***odulation-PCM) is commonly termed linear PCM since it performs no compression, storing each value as a separate 16-bit value. Uncompressed audio gives the user the highest audio quality.

2.4.4 Audio Data Transfer Rate

Audio data transfer rate is affected by sampling rate, sampling resolution, mono or stereo, audio compression. There is a relation between audio quality and data transfer rate. This is shown in the following table [6].

Table 1 : Audio quality and data transfer rates

Quality	Sample Rate (KHz)	Bits per Sample	Mono/ Stereo	Data Rate (Uncompressed)
Telephone	8	8	Mono	8 KBytes/sec
AM Radio	11.025	8	Mono	11.0 KBytes/sec
FM Radio	22.050	16	Stereo	88.2 KBytes/sec
CD	44.1	16	Stereo	176.4 KBytes/sec
DAT	48	16	Stereo	192.0 KBytes/sec
DVD Audio	192	24	Stereo	1,152.0 KBytes/sec

2.5 QoS Over IP Networks

Admission control is a useful technique in telecommunication networks, Internet and multimedia system to achieve the quality of the service. Different strategies are adopted to do admission control in different types of network. In circuit switching networks, a circuit is not established if there is not enough free bandwidth to create that circuit. On the other hand, the Internet, which is basically a packet switched network, uses traffic shaping for QoS adaptation during congestion.

The telephone network is a circuit switched network, which carries audio between a source (the caller terminal) and a destination (the called terminal). When a user dials a number, the telephone switch finds a free end-to-end voice circuit to carry the audio transmission. If such a path is available then the call is set up, otherwise the user gets the busy tone. All users are guaranteed to receive uninterrupted service if a call is set up.

Throughput maximization in a telephone network depends on the algorithm used for selecting particular paths for new circuits [6].

Currently, the Internet is based on a best-effort datagram service model: this model does not require (and generally does not permit) resource reservation prior to data transmission. When a packet arrives at a router, and sufficient resources are available, the packet is forwarded to the next router. However, if the necessary resources are not available, the incoming packet may be delayed, or even dropped. It is therefore difficult to predict, let alone guarantee, the bandwidth or latency experienced by a stream of packets under best-effort datagram services. And since each packet of a session is forwarded through the network independently, packets may experience variable and unpredictable delays, and may arrive at the destination out of order. This service has many advantages, but it is unworkable for real-time multimedia applications requiring absolute standards of performance such as continuous bandwidth during a Video on Demand session with maximum delay and jitter constraints [6]. Hence a best-effort datagram service model is not considered suitable for the Internet2 [11], which proposes to offer the end-to-end quality-of-service guarantees similar to that of the telephone network.

Class-based forwarding proposals such as DiffServ [11], where the packets of applications requiring guaranteed QoS are assigned higher priority classes than the best-effort traffic, are at best partial solutions: they provide superior service to the QoS-sensitive application in the relative sense, i.e., relative to the best-effort traffic, but they cannot guarantee absolute standards of QoS to applications, including telephony and interactive video communication, which require such standards.

RSVP [11] is a protocol used to reserve resources i.e., link bandwidth over the Internet. It requires reservation in each switch from the source to the destination, which clearly requires the determination of a fixed path on which all datagram of the flow are carried. This protocol works for real-time audio and video transmission, and in that sense could provide a basis for guaranteed QoS, but scalability is a problem.

The Multi Protocol Lebel Switch (MPLS) [11] provides a mechanism for sending data independent of the IP routing tables in the routers. In this mechanism each packet is routed through a predefined path, which is determined before data transmission. A label is added to the packet, and this label is used for table look up in the router for forwarding packets to the next router with another label. The label forwarding table is created at the time of fixing the path and it contains additional information such as *Class-of-Service* (CoS) values that can be used to prioritize packet forwarding. As MPLS is a lower layer protocol than IP and UDP, real-time multimedia transmission using UDP over MPLS is considered plausible.

2.6 Scheduling of Real Time Jobs

Reserving resources for a particular period of time in future is a problem of scheduling. More specifically, the problem of scheduling is to find a feasible solution that schedules all time-critical continuous media tasks in a way that each of them can meet its deadlines. For scheduling of multimedia two conflicting goals must be considered. Firstly, a non-critical process should not suffer from starvation and secondly a time critical process must never be subject to priority inversion. Note carefully that in contrast to the traditional scheduling on time sharing computers, the main goal of real-time scheduling is to provide a schedule that allows as many time-critical processes possible, to be processed in time, according to the deadline. There are several attempts to solve real-time scheduling problems. Many of them are just variations of two basic algorithms namely Earliest Deadline First (EDF) algorithm and Rate Monotonic Scheduling algorithm (RM).

In EDF algorithm, as the name indicates, at every new ready state, the scheduler selects the task with the earliest deadline among the tasks to be scheduled. EDF is an optimal, dynamic algorithm having the complexity of $O(n^2)$ with n tasks. An extension of EDF is the Time-Driven Scheduler (TDS) which presents a scheme to tackle with the overload situations. In [12] another priority–driven EDF scheduling algorithm is introduced, where every task is divided into a mandatory and optional part. Here the tasks are scheduled according to the deadline of the mandatory part and the optional parts are processed if the resource capacity is not fully utilized.

Rate monotonic scheduling [13] is an optimal, static, priority-driven algorithm for preemptive periodic jobs. Several extensions exist to this algorithm. One of them, like an extension to EDF, divides a task into a mandatory and optional part. The mandatory part is scheduled according to the rate monotonic algorithm whereas for the scheduling of the optional part different policies are suggested [12,14].

Apart from EDF and rate monotonic scheduling algorithm, there are other scheduling algorithms like Least Laxity First (LLF), Deadline Monotone Algorithm, and Shortest Job First (SJF) etc. In LLF, the task with the shortest remaining laxity is scheduled first [15,16]. The laxity is the time between the actual time and the deadline minus the remaining processing time. LLF is an optimal dynamic algorithm for exclusive resources. Furthermore it is an optimal algorithm for multiple resources if the ready-times of the real time tasks are the same. A fixed priority assignment according to the deadlines of the tasks are done in the deadline monotonic algorithm. In SJF the task with the shortest remaining computation time is chosen for execution [15]. SJF guarantees that as many task as possible meet their deadlines under an overload situation if all of them have the same deadline.

The above mentioned priority-driven algorithms for real time scheduling of periodic tasks are not optimal for scheduling non-preemptable or sporadic jobs [17]. Moreover these algorithms are not optimal when the system is overloaded. During an overloaded situation when it is not possible to schedule all the jobs to meet their deadlines, it may make sense to reject the less critical jobs so that the more critical jobs can meet their deadlines. Our problem deals with non-preemptable and sporadic jobs that may be overloaded in the system. For these reasons we did not use the above mentioned algorithms and we propose a technique of admission controlling.

2.7 Necessity for Admission Controller in MSS

In order to ensure the guaranteed Quality of Service (QoS) in a multimedia system, admission control is necessary [10]. The system has to ensure that enough resources are available at run time to meet the minimum quality guarantee. Actually, admission control is the process of making decision to accept or reject a multimedia session based on the availability of server resources (I/O bandwidth, CPU cycles, memory, etc), link Bandwidth of network and the revenue earned by that session. The objective of this admission controlling is to maximize the revenue by providing the guaranteed service to the users.

2.8 Simple Multimedia Server Architecture

The server is an active element in the Multimedia Server System which provides a predefined set of services that are accessible through the communication network. The main function of a multimedia server is to provide the continuous delivery of data (multimedia streams) in real time to the users (clients).

Figure 2.3 illustrates a typical architecture for a multimedia server [1]. The storage subsystem stores the multimedia data. It may contain devices of different types (e.g. disks, CDs, DVDs, MO etc.) with different characteristics. An efficient and cost effictive server design exploits the difference in device characteristics as well as in costs. Similarly, the design process also considers network subsystems that can be used for the transmission of multimedia data to the clients.

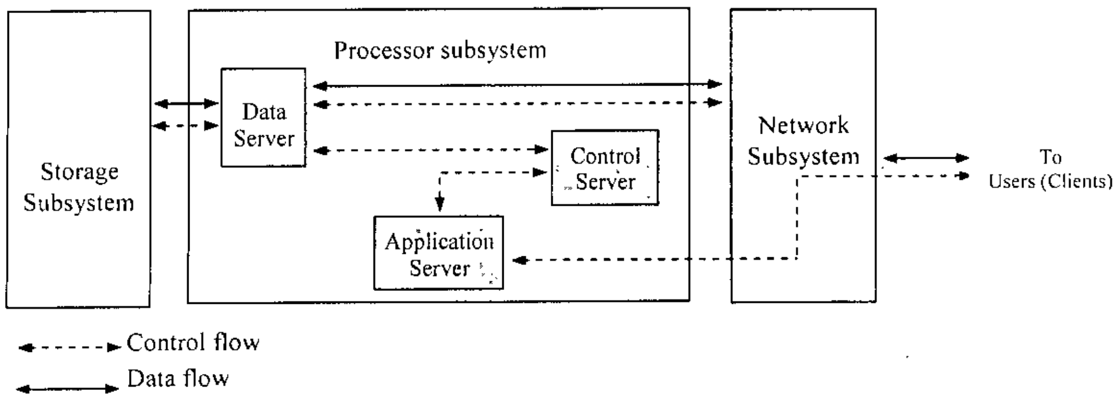


Figure 2-3 : Simple multimedia server

The software processes executing in the processor subsystem are responsible for the management and operation of the multimedia server. The software architecture of multimedia server consists of three primary elements [1] i.e. *Application server* interacts with a user and selects a specific media manager segment to be retrieved. *Control server* performs admission control and resource reservation as well as resource optimization and the *Data server* delivers the data to the network.

Generally there will be one *Application Server* in processor subsystem which will receive the application command from the user (client). The application server converts these application commands for displaying specific images and video files.

The multimedia server commands are next received by the *Control Server*. The control server has three major functions. They are as follows -

1. *Making control decisions regarding multimedia data flow.*

   (i.e. Admission controlling)

2. *Performing various optimizations that increase overall server efficiency.*
3. *Hiding the complexity of the configuration from the application server.*

Finally, the *Data Server* is responsible for the actual retrival and delivery of the multimedia data. The major distinguishing characteristic of multimedia applications is their Quality of Service (QoS) requirments. These may include end-to-end delay, delay jitter, resolution, frame rate, audio quality etc. Providing the data delivery while satisfying the QoS requirments is perhaps the most important function of the data server.

2.9 Resources of Multimedia Server

Multimedia server needs resources for their operation in order to provide services for multimedia applications. A resource is a system entity required by processes for manipulating data. A resource can be used exclusively (e.g. loudspeaker), or shared among various processes (e.g. Bandwidth). Again a resource can be active or passive. An active resource is the CPU or a network adapter for protocol processing; it provides a service. A passive resource is the main memory, communication Bandwidth; it denotes some systems capabilities required by active resources. Each resource has a capacity which results from the ability of certain task to perform using the resource in a given time-span. In this context, capacity refers to CPU capacity, frequency range or, for example, the amount of storage.

Following resources are very much required for successfully multimedia streaming.

- Disk space in the hard disk,

- I/O bandwidth to carry multimedia stream from hard disk to the memory of the computer,
- Memory for temporary storage of multimedia stream,
- CPU cycles to execute necessary instructions for fetching multimedia from hard disk to computer memory and memory to Network Interface Card (NIC),
- Network Bandwidth to transmit multimedia at sufficient bit rate,
- Power of NIC to inject packet to the network at sufficient speed.

Our special interest is the resources which are *shared* among applications, system and network, such as ***CPU cycles***, ***memory***, and ***I/O bandwidth*** and ***network Bandwidth***.

2.9.1 CPU Cycles

Every conventional processor has a clock with a fixed *cycle time* or *clock rate*. Clock rates are generally measured in ***MHz*** (millions of cycles/second) and clock times are often measured in ***ns*** (nanoseconds). But at present different company is producing ***GHz*** clock rate processor (i.e. clock rate of Intel Xenon processor is 3.0 GHz etc). In order to maximize the utilization of CPU, the CPU cycles are shared. CPU cycles = Instructions executed * CPI (average clock ***C***ycles ***P***er ***I***nstruction). Utilization of CPU cycles is measured in percentage.

2.9.2 Memory

Computer memory is organized into a hierarchy. At the highest level (closest to the processor) are the processor registers. Next comes one or more levels of cache. When multiple levels are used, they are denoted L1, L2 etc. Next comes main memory, which is usually made out of dynamic random access memory (DRAM). All of these are considered internal to the computer system. Main memory is a shareable passive resource of a server. Currently addressable main memory of an entry level server is ranging from hundreds megabytes to some gigabytes.

2.9.3 I/O Bandwidth

I/O Bandwidth is the maximum amount of data that can travel in between I/O devices (i.e. hard disk) and memory in a given time. In a VoD server, movies are generally stored in hard disk drive that will be retrieved by the hard disk controller. During multimedia streaming a block of data will be read into the memory of the server. The data from the memory will be transmitted through the Network Interface Card (NIC) to the users (workstations). In order to ensure the guaranteed service the I/O controller (i.e. hard disk controller) must read the data block in a particular time duration. As lots of sessions are served simultaneously by the server, the I/O controller must have the capacity to retrieve necessary data for all the sessions. The data retrieving capacity of I/O controller is defined as I/O bandwidth which is measured by bits/sec.

The resources mentioned above are of different types based on the availability and cost. Although storing of movies requires large disk space, it is not very much costly and thus addition of more hard disks is not a very difficult job to do. The requirement of memory and CPU cycles for multimedia streaming is not as high as I/O and Network bandwidth. Besides this, the required amount of memory and CPU cycle is proportional to the amount of I/O bandwidth or the rate of multimedia playback. Thus I/O bandwidth in an MSS could create bottleneck. On the other hand the total I/O bandwidth of the storage subsystems is almost fixed for a particular type of hard disk, (e.g., the *RPM* of the hard disk are 7200, 5800, etc.). That is why the capacity of the multimedia server is denoted by the available I/O bandwidth of the hard disks. To avoid the bottleneck of the system, the other resources such as memory and CPU cycles are configured as proportional as or sufficiently larger than the required one. This will ensure enough resources for the multimedia streams served by the I/O bandwidth of the system. Only the I/O bandwidth is considered in this calculation of admission controller.

2.10 Storage Subsystem of Multimedia Server

Multimedia data is stored on the various storage devices, which are connected to the processor through device interfaces or controllers. Device interfaces provide two important functions. First, the storage device may contain mechanical components, and therefore may be much slower than the processor. The storage interfaces buffer the data and provide matching of speed required for data transfer. Second, a sequence of commands may be needed to retrieve a single block from a storage device. The controller allows the processor to send the sequence of commands to the controller, relieving the processor of this burden. Because multimedia data travels through the controllers, the performance of the I/O subsystem is heavily influenced by the performance of the controllers, not just the storage devices.

The controllers are connected, in turn, to the processor. The internal architecture of the processor determines the precise manner in which the controllers are connected. Therefore, the processor architecture also has a strong impact on the performance of the I/O subsystem.

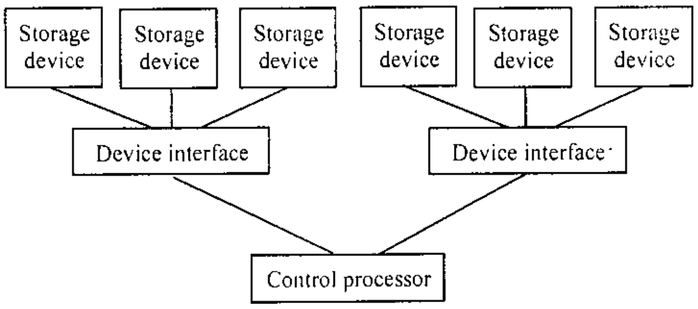


Figure 2-4 : Storage subsystem

In discussing the characteristics of the various storage devices, we concentrate on three primary measures that are storage capacity, access time, and Bandwidth [1]. The storage capacity of the device is important because multimedia files are large. The access time, the delay between the time the device receive the command to read or store multimedia data and the time the device actually begins to read or store the data, is important for two reasons. First, device with large access times may not be suitable for interactive applications where quick response time is needed. Second, large access times also reduce effective data transfer rates. Consider the retrieval of successive blocks from the same device. Because no data is transferred during the access time, the larger the access time, the smaller the effective data retrieval rate.

The next measure we consider is the Bandwidth of the device- the maximum data rate the device can deliver. The Bandwidth has an important influence on QoS support. For example, if a device has a Bandwidth of 10 MB/s and an application requires a Bandwidth of 0.5 MB/s, the device will not be able to support more than 20 instances of the application.

2.10.1 Comparative Summary of Different Storage Device

The differing constructions and physical properties of the various devices result in different storage and access characteristics. Hard disks provide much higher transfer rate ranging from 13.3 MB/s to 160 MB/s [18] or more. Hence disks are most suitable for serving multiple users concurrently. Their storage capacities (ranging from 20 GB to 180 GB or more), as high as those of tapes (20 GB to 110 GB), are comparable to and even higher than those of DVDs (8.5 GB to 17 GB). Therefore, disks are an ideal storage device for frequently accessed files.

Tapes are sequential-access device that provide large storage capacity but are not suitable for concurrent or interactive access (57-100 seconds). Therefore, less frequently accessed files can be archived on tapes and brought to disks for playback. Tapes may be used as a tertiary storage for Multimedia servers. Both DVDs and CDs are designed for single-stream delivery (due to their higher access time). DVDs provide both higher Bandwidth and storage capacity than sequential access device, making them suitable for video storage and playback. CDs are also suitable for video storage and playback.

2.10.2 RAID

The basic idea of RAID [19] was to combine multiple small, inexpensive disk drives into an array of disk drives which yields performance exceeding that of a Single Large Expensive Drive (SLED). Additionally, this array of drives appears to the computer as a single logical storage unit or drive.

Five types of array architectures, RAID-1 through RAID-5, were defined in [19], each providing disk fault-tolerance and each offering different trade-offs in features and performance. In addition to these five redundant array architectures, there are some more RAID concepts including RAID 0 are available now-a-days.

The large capacity, high-speed drives available today might seem to make the need for RAID irrelevant. However, the need for data storage capacity and high speed data access is growing faster than the capacity and transfer rates of individual drives. RAID is still the best solution for providing large amounts of data storage at a reasonable cost, with the added benefit of data protection.

In a Multimedia Server where large records, such as video, audio, and images, are stored, RAID can be used in order to reliable and efficient retrieval of the data for delivering to users.

2.11 Literature Review of Admission Control

Admission control generally requires two components: knowing the *load* that a particular job will generate on a system, and knowing the *capacity* of that system. By keeping the maximum amount of load just below the system capacity, overload is prevented and peak throughput is achieved. There are various algorithms that focus the different issues of admission control. A brief literature review of different researches of admission control is presented below.

Domjan [20] proposed an admission controller using the Simplex Linear Programming method for resource reservation of adaptive applications. There is a starting and an end time for each application and the admission control for new requests are done to maximize the utilization of resources.

In [21] an observation-based admission control algorithm was proposed, in which a client is admitted for service by a multimedia server only if the predicted extrapolation from the status quo measurements of the storage server utilization indicate that the services requirements of all the clients can be met satisfactorily. The performance of admission control algorithm and hence, the number of clients admitted and served simultaneously are maximized by employing a disk scheduling algorithm that minimizes both the seek time and rotational latency incurred while accessing a sequence of media blocks from disk.

Chen [22] proposed an admission control algorithm that makes acceptance/rejection decisions not only to satisfy the hardware requirements of clients' requests but also to optimize the reward of the system based on a performance criterion as it serves the clients of different priority classes. In [23] Chen proposed a cost based approach to address the issue of how much resources should be allocated to serve multimedia requests.

Admission control procedures, which consider requests for multiple streams from multiple destinations and resolve contention when users' requests exceed available network resources, are presented in [24]. In this research real-time multicast communication is considered in which each destination makes an individual bid for delivery of a subset of real-time hierarchically encoded streams that are offered to the session by the source. The objective is to customize stream delivery to destinations, based on their requests and network constraints.

In [25] two new types of admission control schemes for the VoD services are proposed. They are the Enhanced Strict Admission Control (ESAC) and the Probabilistic Admission Control (PAC). In the ESAC scheme, it is proposed to use more statistics than the peak frame size of the stored video information to strictly guarantee the QoS requirement and to achieve potentially much higher throughput. In the PAC scheme, it is proposed to use similar statistics as used in the ESAC scheme to achieve even higher throughput. Compared with the predictive admission control scheme, the PAC scheme does not require real-time traffic measurement collection and analysis, and can guarantee that the maximum allowable data overdue probability be not exceeded.

A measurement-based Admission Control Algorithm (MBAC) is presented in [26] that consist of two logically distinct pieces, the *criteria* and the *estimator*. The admission control criteria are based on an equivalent token bucket filter model, where each predictive class aggregate traffic is modeled as conforming to a single token bucket filter. The estimator produces measured values (i.e., experienced delay and utilization of network) that are used in the admission control criteria. Later Moore [27] presented an implementation based comparison of the measurement-based Admission Control Algorithm.

A QoS aware job scheduling is proposed in [28] for a cluster-based web server where a few computing nodes are separately reserved for high-performance computing applications. As an example application, a multimedia server was considered that dynamically generates video unites to satisfy the bit rate and bandwidth requirements of a variety of clients. To perform QoS aware scheduling of multiple multimedia jobs on the computing servers, a two-step algorithm is proposed. The first step is to fairly schedule multimedia streams to satisfy each stream's QoS requirement; and the second step is to balance the workload among heterogeneous computing nodes in the cluster. A new Quota-based Adaptive CoScheduling (QACS) algorithm that greatly reduces delay jitter by eliminating the out-of-order departure for outgoing streams, as well as achieves high throughput in a heterogeneous cluster is proposed for this purpose.

- Admission control and request scheduling for multiply-tiered E-commerce Web sites are presented in [29]. By externally measuring service costs online and distinguishing between different types of requests, the proposed approach can achieve both stable behaviors during overload and dramatically improved response times. The proposed method is embodied in a transparent proxy called Gatekeeper. Gatekeeper intercepts requests from the application server to the database, allowing interoperability with standard software components. Gatekeeper identifies different request types and maintains online estimates of their expected service times, based on measurements of recent executions. The estimated request service time indicates the load that a request imposes on the system. The current load on the system is calculated as the sum of the estimated services times of all executing requests. The system capacity is measured offline and it limits the number of requests admitted to the system; excess requests are queued. The Gatekeeper proxy uses request scheduling to reduce the average response time of dynamic Web site interactions, in the form of a shortest job first (SJF) policy.

2.12 Multimedia Communications

Multimedia communications [6] encompasses the delivery of multiple media content, such as text, graphics, voice, video, still images, and audio, over communications network to users. Several of these media types may be part of a particular interaction between or among the users.

The media to be transmitted, often called sources, are represented in digital form, and the networks used to transmit the digital source representations may be classified as digital communications networks, even though analog modulation is often used for free-space propagation or for multiplexing advantages. In addition to the media sources and the networks, the user terminals, such as computers, telephones and personal digital assistants (PDAs) also have a large impact on multimedia communications.

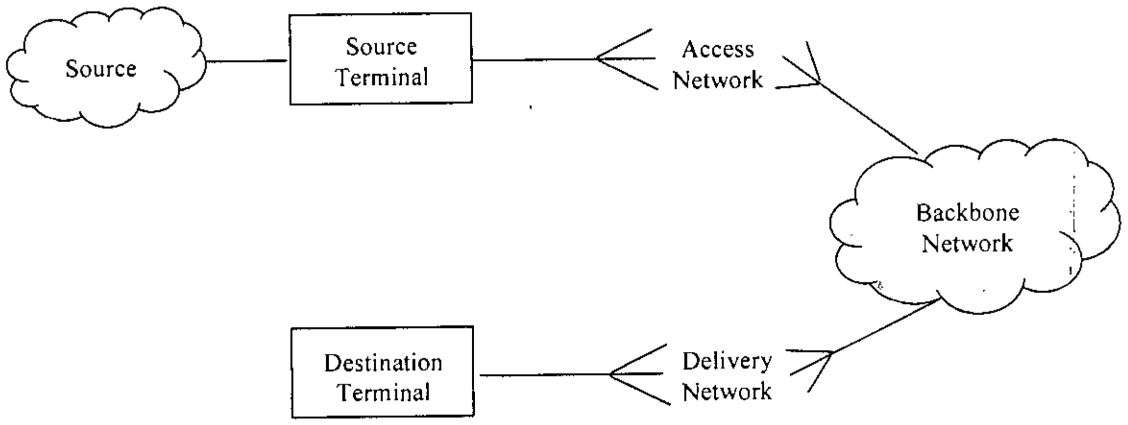


Figure 2-5 : Components of a multimedia communications network

The components of multimedia communications shown in the Figure 2-5 are the source, the source terminal, the access network, the backbone network, the delivery network, and the destination terminal. This categorization allows us to consider two-way, peer-to-peer communications connections, such as videoconferencing or telephony, as well asymmetric communications situations, such as broadcasting or video streaming. In Figure 2-5, the source consists of any one or more of the multimedia sources, and the job of the source terminal is to compress the source such that the bit rate delivered to the network connection between the source terminal and the destination terminal is at least approximately appropriate. Other factors may be considered by the source terminal as well. For example, the Source Terminal may be a battery- power-limited device or may be aware that the Destination Terminal is limited in signal processing power or display capability. Further, the Source Terminal may packet the data in a special way to guard against packet loss and aid error concealment at the destination terminal. All such factors impinge on the design of the source terminal.

The access network may be reasonably modeled by a single line connection, such as a 28.8 Kbit/s modem, a 56 Kbit/s modem, a 1.5 Mbit/s Asymmetric Digital Subscriber Line

(ADSL) line, and so on, or it may actually be a network that has shared capacity, and hence have packet loss and delay characteristics in addition to certain rate constraints.

The backbone network may consist of a physical circuit switched connection, a dedicated virtual path through a packet- switched network, or a standard best-effort Transmission Control Protocol/Internet Protocol (TCP/IP) connection, among other possibilities. Thus, this network has characteristics such as Bandwidth, latency, jitter, and packet loss, and may or may not have the possibility of Quality of Service (QoS) guarantees.

The delivery network may have the same general set of characteristics as the access network, or one may envision that in a one-to-many transmission that the delivery network might be a corporate intranet. Finally, the destination terminal may have varying power, mobility, display, or audio capabilities.

The source compression methods and the network protocols of interest are greatly determined by international standards, and how these standards can be adapted to produce the needed connectivity is a challenge. The terminals are specified less by standards and more by what users have available now and are likely to have available in the near future. The goal is delivery of multimedia streams via seamless network connectivity.

2.12.1 Considerations in Multimedia Network

A multimedia networking system allows for the data exchange of discrete and continuous media among computers. Multimedia data specially the continuous media (audio, video, etc.) have the following characteristics:

- voluminous
- real-time (synchronization, esp. between audio and video)
- interactive (e.g., in video conferencing, interactive TV).

Considering the above mentioned parameters of multimedia data, multimedia networks also should have the following parameters:

- Bandwidth Requirements - Constant bit rate (CBR) / Variable bit rate (VBR)

- Quality of Service (QoS)
  - Delay,
  - Delay jitter
  - Packet loss
  - Error resilience
- Real-time constraints
- Synchronization of video, audio, data, applications ...
- Cost.

2.12.2 Controlled Network

A controlled Network is owned by an organization/Institution in which admission controlling of users is done. QoS, Traffic, congestion, delay, delay jitter etc are controlled by administrator in a controlled network. The Internet is not a controlled network.

2.12.3 Enterprise Network

An Enterprise network (EN) is a private network, usually with fewer than 100 nodes (switches), and administered by a single organization [4]. This network can be used by the company for data transmission. For example, a company can set up an Enterprise Network among all of its offices in Europe. Teleconferencing or videoconferencing may be the applications over this Enterprise Network. Similarly, a telecommunication company can set up an Enterprise Network especially for multimedia stream transmission. A multimedia user will submit her request to the owner of the EN for Bandwidth from a source node to a destination node of the EN and the admission controller will decide whether a request can be entertained or not after doing computations.

2.12.4 A Practical Network

A practical network may be composed of different network services like Plain old telephone service (POTS), Integrated Services Digital Network (ISDN), Asymmetric Digital Subscriber Line (ADSL), and Ethernet etc. Hence the different networks service have different data transfer rate in a practical network. For MPEG1 multimedia stream transmission the minimum data transfer rate should be 1.5 Mbits/s [8] in the network.

Two critical characteristics of networks are *transmission rate* and *transmission reliability*. The desire to communicate using multimedia information affects both of these parameters profoundly. Transmission rate must be pushed as high as possible, and in the process, transmission reliability may suffer. Different networks and network services according to rate are shown in following table [8].

Table 2 : Network services and data rates

Service/Network	Rate
POTS	28.8-56 Kbits/s
ISDN	64-128 Kbits/s
ADSL	1.544-8.448 Mbits/s (downstream) 16-640 Kbits/s (upstream)
VSDL	12.96-55.2 Mbits/s
CATV	20-40 Mbits/s
OC-N/STS-N	NX51.84 Mbits/s
Ethernet	10 Mbits/s
Fast Ethernet	100 Mbits/s
Gigabit Ethernet	1000 Mbits/s
FDDI	100 Mbits/s
802.11 (Wireless)	1, 2, 5.5, and 11 Mbits/s in 2.4 GHz band
802.11a (Wireless)	6- 54 Mbits/s in 5 GHz band.

A practical network is illustrated below.

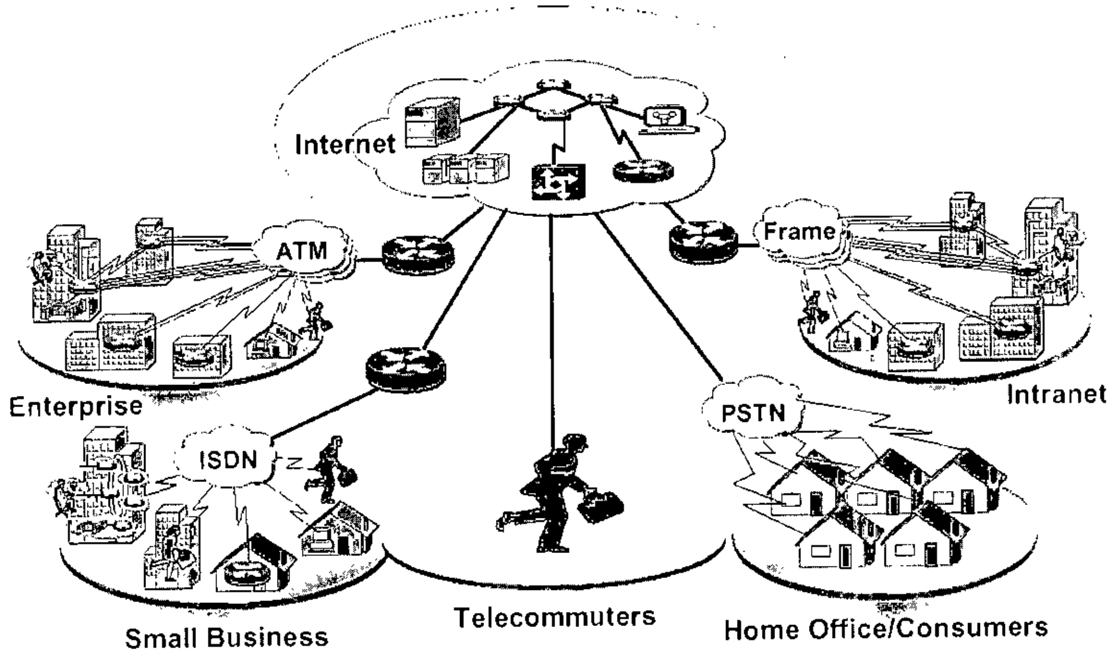


Figure 2-6 : A practical network

Certainly, most people today connect to the Internet through the plain old telephone system (POTS) using a modem that operates at 28.8 Kbits/s up to 56 Kbits/s.

2.12.5 ISDN

Integrated Services Digital Network (ISDN) is a type of circuit switched telephone network system, designed to allow digital (as opposed to analog) transmission of voice and data over ordinary telephone copper wires, resulting in better quality and higher speeds, than available with analog systems. More broadly, ISDN is a set of protocols for establishing and breaking circuit switched connections, and for advanced call features for the end user. In ISDN, there are two types of channels, *B* and *D*:

- *Bearer* or *B channels* are used for data -- this refers both to voice and data information, and
- *Delta* or *D channels* are intended for signaling and control (but can also be used for data).

There are two kinds of access to ISDN [6]:

- **Basic rate interface (BRI)** also referred to as **Basic rate access (BRA)** - consisting of two B channels, each with bandwidth of 64 kbit/s, and one D channel with a bandwidth of 16 kbit/s. These three channels can be designated as 2B+D.
- **Primary rate interface (PRI)** also referred to as **Primary rate access (PRA)** - containing a greater number of B channels and a D channel with a bandwidth of 64 kbit/s.

The number of B channels varies based on the country [6]:

- North America and Japan: 23B+1D, aggregate bit rate of 1.544 Mbit/s (T1)
- Europe, Australia: 30B+1D, aggregate bit rate of 2.048 Mbit/s (E1).

2.12.6 Asymmetric Digital Subscriber Line (ADSL)

Asymmetric Digital Subscriber Line (ADSL), a modem technology, converts existing twisted-pair telephone lines into access paths for multimedia and high-speed data communications. ADSL can transmit up to 9 Mbps to a subscriber, and as much as 832 kbps or more in both directions. Such rates expand existing access capacity by a factor of 50 or more without new cabling. ADSL is literally transforming the existing public information network from one limited to voice, text and low resolution graphics to a powerful system capable of bringing multimedia, including full motion video, to everyone's home in this century.

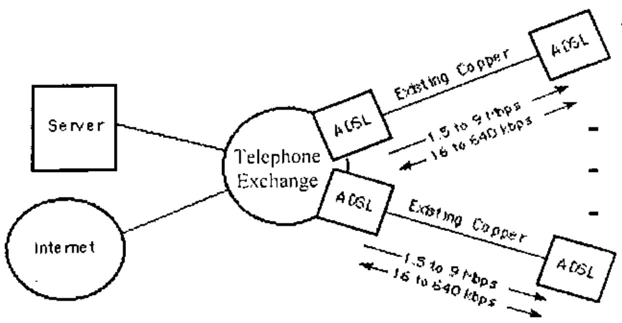


Figure 2-7 : ADSL connection

An ADSL circuit connects an ADSL modem on each end of a twisted-pair telephone line, creating three information channels – (i) a high speed downstream channel, (ii) a medium speed duplex channel, depending on the implementation of the ADSL architecture, (iii) and a POTS (Plain Old Telephone Service) or an ISDN channel.

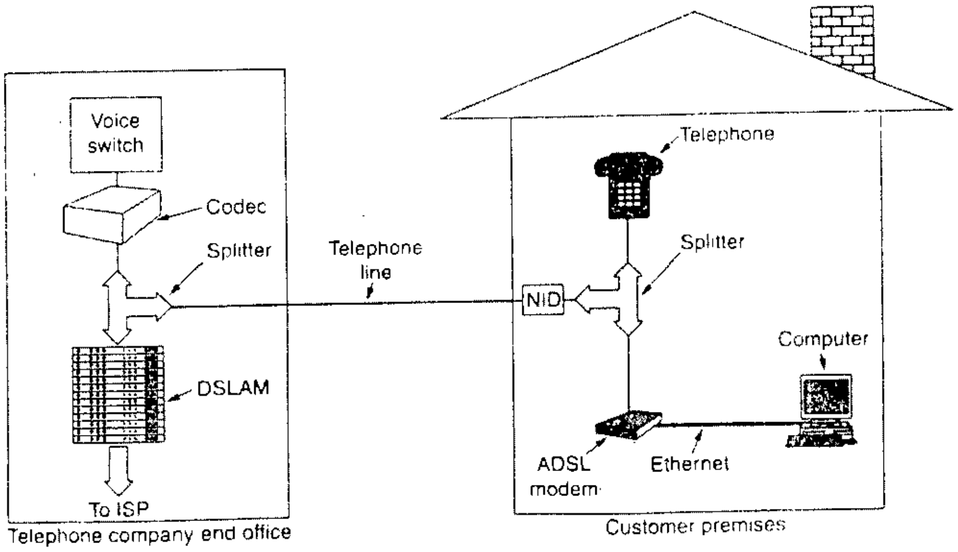


Figure 2-8 : A typical ADSL equipment configuration

(Courtesy: Computer Network by Tanenmbum, 4th Edition)

The POTS/ISDN channel is split off from the digital modem by splitter (filters), thus guaranteeing uninterrupted POTS/ISDN, even if ADSL fails. The high speed channel ranges from 1.5 to 8 Mbps, while duplex rates range from 16 to 832 kbps. Each channel can be sub-multiplexed to form multiple, lower rate channels, depending on the system.

The initial ADSL offering was from AT&T and worked by dividing the spectrum available on the local loop [30], which is about 1.1 MHz, into three frequency bands: POTS (Plain Old Telephone Services) upstream (user to end office) and downstream (end office to user).

The alternative approach, called DMT (Discrete Multitone), has 1.1 MHz spectrum available on the local loop that is divided into 256 independent channels of 4312.5 Hz each. Channel 0 is used for POTS. Channel 1-5 are not used, to keep the voice signal and data signals from interfering with each other. Of the remaining 250 channels, one is used for upstream control and one is used for downstream control. The rest are available for user data.

ADSL modems provide data rates consistent with North American and European digital hierarchies and can be purchased with various speed ranges and capabilities. The minimum configuration provides 1.5 or 2.0 Mbps downstream and a 16 kbps duplex channel; others provide rates of 8 Mbps and 64 kbps duplex. Products with downstream rates up to 8 Mbps and duplex rates up to 640 kbps are available today.

Downstream data rates depend on a number of factors, including the length of the copper line, its wire gauge, presence of bridged taps, and cross-coupled interference. Line attenuation increases with line length and frequency, and decreases as wire diameter increases. While the measure varies from provider to provider, these capabilities can cover up to 95% of a loop plant depending on the desired data rate. Customers beyond these distances can be reached with fiber-based digital loop carrier systems.

Many applications enabled by ADSL involve digital compressed video. As a real time signal, digital video cannot use link or network level error control procedures commonly found in data communications systems. ADSL modems therefore incorporate forward error correction that dramatically reduces errors caused by impulse noise. Error correction on a symbol-by-symbol basis also reduces errors caused by continuous noise coupled into a line.

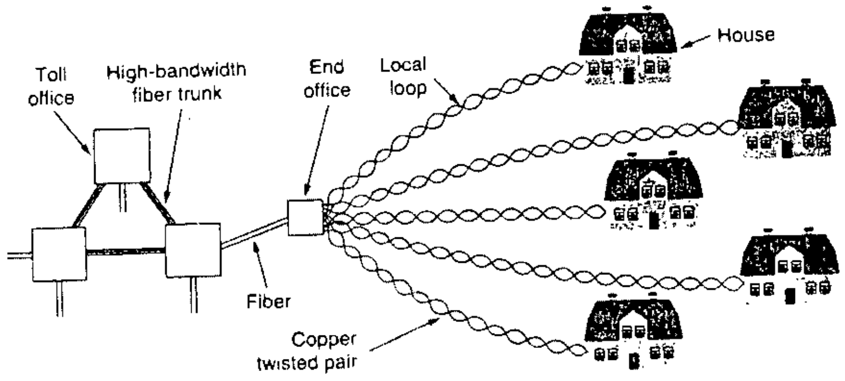


Figure 2-9 : ADSL connection of home and end office

(Courtesy: Computer Network by Andrew S. Tanenbum, 4th Edition)

2.12.7 Cable TV Network

Essentially, a basic cable TV network is a television distribution facility. A set of TV programs are received from satellite and/or terrestrial broadcasts and the cable network is then used to distribute this set of programs to subscriber premises. Early cable television networks – of which there are still many in existence – are based entirely on coaxial cable. These were designed to distribute broadcast television (and radio) programs that are received at a central site to customer premises geographically distributed around an area such as a town or city. Such networks are known as Community Antenna Television (CATV) networks. The data transmission rate of CATV is 20 Mbits/s or more.

2.12.8 Ethernet

The most widely used high speed LANs today are based on Ethernet and were developed by IEEE 802.3 standards committee. A family of 100 Mbps LANs known as Fast Ethernet currently dominates the high-speed LAN market. A more recent entry is Gigabit Ethernet and 10 Gigabit Ethernet [31].

Classical Ethernet operates at 10 Mbps over a bus topology LAN using CSMA/CD (carrier sense multiple access with collision detection) medium access control protocol.

Fast Ethernet refers to a set of specifications developed by the IEEE 802.3 committee to provide a low-cost, Ethernet compatible LAN operating at 100 Mbps. The comprehensive designation for these standards is 100 BASE-T. The committee defined a number of alternatives to be used with different transmission media.

In late 1995, the IEEE 802.3 committee formed a High-Speed Study Group to investigate means for conveying packets in Ethernet format at speeds in the gigabit-per-second range [30, 31]. The strategy for Gigabit Ethernet is the same as that for Fast Ethernet. While defining a new medium and transmission specification, Gigabit Ethernet retains the CSMA/CD protocol and frame format of its 10 Mbps and 100 Mbps predecessors. It is compatible with 100BASE-T and 10BASE-T, preserving a smooth migration path. With gigabit products still fairly new, attention has turned in the past several years to a 10 Gbps Ethernet capability. The principal driving requirement for 10 Gigabit Ethernet is the increase in Internet and intranet traffic carrying multimedia streams.

2.13 Summary

In this chapter we discussed the various terms that are used in our thesis. We define Multimedia, Multimedia Server System (MSS), QoS and the other parameters for MSS. We also discussed component of the multimedia communication network. A comparison of data transfer rate of different network was depicted. Finally the background knowledge and a literature review relating to our problem were presented.

Chapter 3

Reservation Based Multimedia Server System

3.1 Introduction

In this chapter we will introduce reservation based admission control of a Multimedia Server System, a new scheme in admission control of a multimedia server. The feasibility of the problem with respect to the network bandwidth (i.e., whether the proposed system is feasible with the existing network infrastructure) will be also discussed. This chapter also presents the mathematical formulation and hardness of the proposed reservation scheme in admission control problem.

3.2 Proposed Multimedia Server System

Multimedia Server System (MSS) has a server that can provide multimedia services to the clients (users) i.e. Video on Demand (VoD) or Movie on Demand service. A Video-on-Demand (VoD) system provides a service, which enables a user of the system to request for real time transmission of live video from a collection of available video data. The customers (clients) can get movie streams from the media server(s) through the networks.

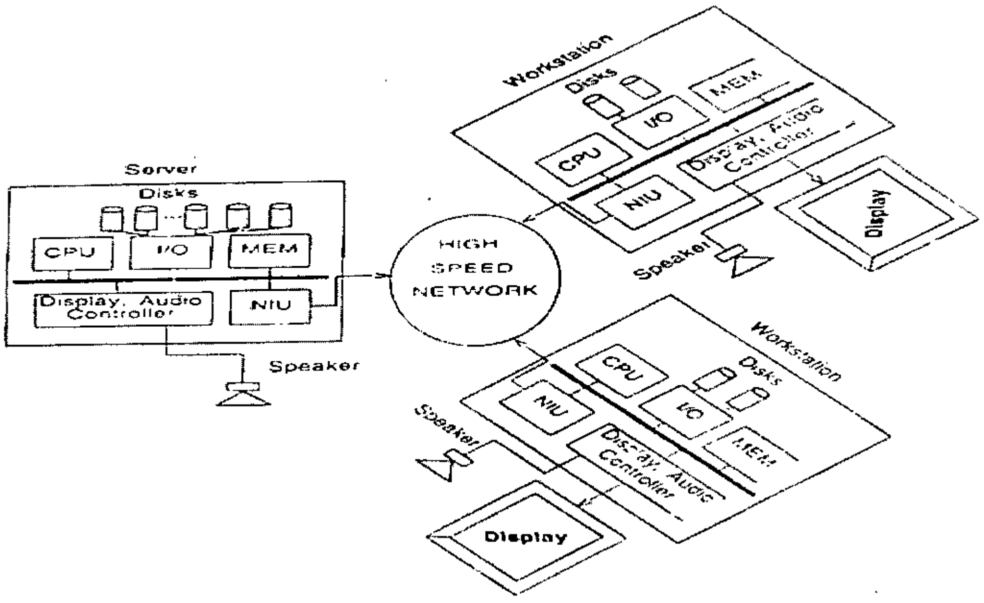


Figure 3-1: Multimedia Server System

Admission control is done when the user submits multimedia session reservation request to the server in order to start a multimedia session (i.e., to get multimedia streams) in future. The admission controller gets multimedia session reservation request and depending upon the availability of resources, the session request may be accepted or rejected. Admission control is a decision making process whether to admit the user or not. In this process the resources is allocated for the sessions. The objective of admission control is to maximize the revenue of the service provider. In the multimedia session request the user specifies the ready time, deadline, duration, and bandwidth requirement of the session to be enjoyed in future. The admission controller notifies the users whether the requests are accepted or rejected. If the requests are accepted then the starting time of the session are also notified to the users by the admission controller.

3.3 Practicability of the Proposed System

One of the important issues of the practicability of the proposed Video-on-Demand (VoD) system is the stream rates; i.e., the existing network system must be capable of transferring the MPEG1 movie stream. Different stream rates of MPEG movie are given below [6].

Table 3: Different multimedia stream rates

Stream coding	Stream rate (Mbits/s)	2 hour film size (GB)	100 hour online capacity (GB)
MPEG1	1.5	1.3	70
MPEG2 (Variation 1)	4	3.5	180
MPEG2 (Variation 2)	5	4.4	220
MPEG2 (Variation 3)	8	7	350

Hence the media server can be located in the telephone exchange and the users may be connected to the exchange through ADSL lines [30]. Any short range and high speed communication media is desirable from the servers to the switch. These lines are capable to carry 1.5 Mbps MPEG-1 streams with considerable delay and jitter. So if we consider VoD service for the users from the servers of the same exchange then there will be no chance of congestion in the network. Thus admission control is not required for the network resources.

3.3.1 Protocols for Multimedia Communications

The most commonly used transport layer protocol in the Internet is the Transmission Control Protocol (TCP). TCP provides reliable, connection-oriented service and is thus well suited for data traffic as originally envisioned for the Internet. Unfortunately, one of the ways reliable delivery is achieved is by retransmission of lost packets. Since this incurs delay, TCP can be problematical for the timely delivery of delay-sensitive multimedia traffic.

Therefore, for multimedia applications, many users often employ another transport layer protocol called User Datagram Protocol (UDP). Unlike TCP, UDP simply offers connectionless, best-effort service over the Internet, thus avoiding the delays associated with retransmission, but not guaranteeing anything about whether data will be reliably delivered. In case of our Multimedia Server System (MSS), mainly the following types of communications were found.

- Admission controlling
- Transmitting continuous media
- Transmitting discrete media.

Admission controlling:

TCP/IP protocol is used for admission controlling because it requires error free delivery of messages. The controlling information are small amount compared to voluminous multimedia information, so the retransmission overhead could be neglected.

Transmitting continuous media:

UDP protocol is used for transmitting continuous media because this protocol provide guarantees for delay and jitter. TCP is not good for transmitting continuous media because of retransmission delay.

Transmitting discrete media:

TCP/IP protocol is used for transmitting discrete media.

The network contains many routers that will not be congested as admission controller does not allow more sessions to admit into the system. Hence packet loss will be minimum.

3.4 Mathematical Formulation of the Problem

Let there be *n* session requests $s_1, s_2, s_3 \ldots\ldots s_n$ from *n* users in a Multimedia Server System. Each session requires b_i resources (Bandwidth) from the server. A multimedia session request can be specified by the following terms -

Ready time (r_i): *Ready time* (r_i) is the earliest time when a session can be started. Actually it indicates that the user is ready to receive multimedia streams from that time.

Deadline (d_i): The admission controller must ensure starting time of session between r_i and the r_i +*deadline* (d_i). It indicates the latest time when the session can start.

Duration of the session (l_i) : Any session of multimedia which is enjoyed by the user must have a particular *duration* (l_i) i.e., length of the session.

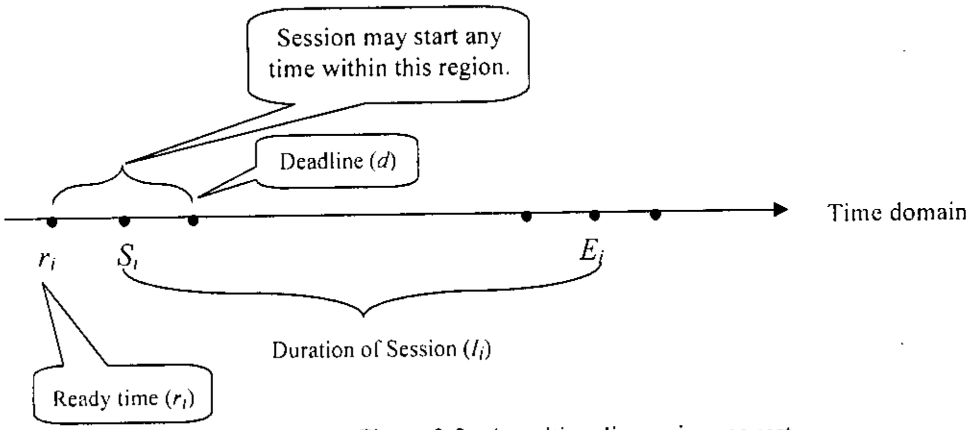

Session may start any time within this region.
Deadline (d)
Time domain
r_i
S_i
E_i
Duration of Session (l_i)
Ready time (r_i)

Figure 3-2 : A multimedia session request

Required resource (b_i): The required resources for the session may be I/O Bandwidth, processor cycle, main memory, and network Bandwidth. Here we only consider Bandwidth (b_i) as required resource. All the resources in a standard multimedia server are always proportional to the I/O Bandwidth as mentioned earlier in Section 2.9.

Utility (u_i): *Utility* (u_i) is defined as the revenue earned by the server from a user enjoying a session. The term utility can also be defined by different kinds of satisfaction of the user (response time, quality of the service etc.), but here we are not considering those parameter. The price (*utility*) offered for a session is considered only.

Mathematically a multimedia session reservation request can be described by s_i

$s_i = (r_i, d_i, l_i, b_i, u_i)$

Where,

r_i = Ready time	d_i = Deadline
l_i = Duration of the session	b_i = Required resource
u_i = Utility	

3.4.1 Inputs of the System

Customers place their session request to the server of MSS (i.e. Video on Demand) through the network in order to book their sessions, so that they can enjoy multimedia streams in future at their desired time. Hence the input of the problem is the session requests, which are defined by ready time, deadline, and duration of the session, required resources and utility of the session.

Multimedia session admission requests are submitted on batches of sessions in a regular time interval which is called an *epoch*. In each epoch some of the old sessions leave and some new session requests are batched for admission.

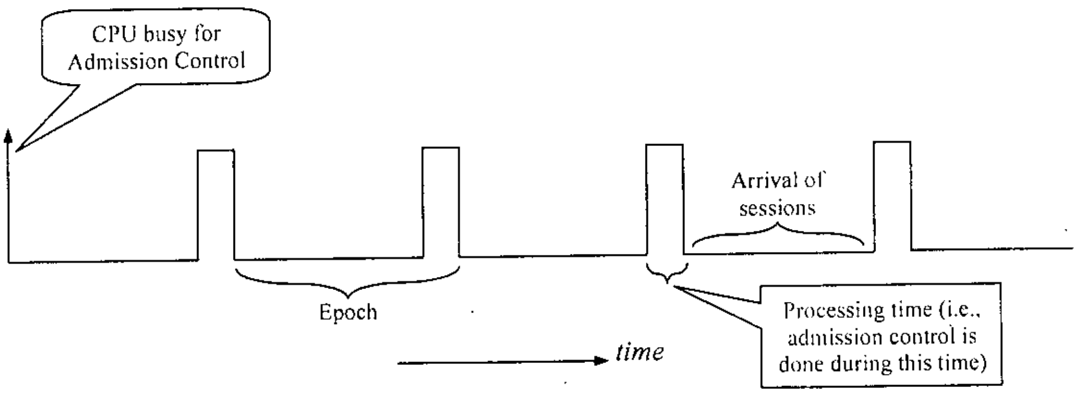

CPU busy for Admission Control
Arrival of sessions
Epoch
Processing time (i.e., admission control is done during this time)
time

Figure 3-3 : Epoch (arrival of sessions)

When some old sessions leave, some resources get free and when some sessions get admitted, some resources are allocated. Admission controller keeps records of all the resources of the system. Admission controller dynamically allocate resources or to re-allocate newly-released resources. Thus there may be a change of resource usage in the system in each epoch.

3.4.2 Constraints of the System

At any moment, the system state has to obey the following constraints i.e., boundary of session duration constraints and resource constraints.

3.4.2.1 Boundary Constraints

Constraints 1 :

The session must start between ready time and (ready time + deadline). *Boundary of starting time S_i : from r_i to r_i+d*

Constraints 2 :

Adding the duration of session (l_i) with the starting time, we will get the ending time.
Boundary of ending time E_i : from $(r_i + l_i)$ to $(r_i + l_i + d)$

The difference between ending time (E_i) and starting time (S_i) must be equal to the length of the session (l_i).

Duration of session: $l_i = E_i - S_i$

3.4.2.2 Resource Constraints

The required resource for the session is assumed as Bandwidth (b_i) and the resource is additive. In a particular time, the sum of quantities of the resources allocated to all the sessions cannot exceed the total available quantity of the resource. For example, the total resource required for n sessions in a particular time are $\sum_{i=1}^{n} b_i(t)$. If the available system resources of the multimedia server are B, the resource constraints are expressed by the following inequality.

$$B > \sum_{i=1}^{n} b_i(t)$$

$$\text{Where, } b_i(t) = \begin{cases} 0, & \text{when } t < S_i \text{ and } t > S_i + l_i \\ b_i, & \text{when } t >= S_i \text{ and } t <= S_i + l_i \end{cases}$$

3.5 Objectives of the System

Commercial Multimedia Server System must have an objective which optimizes the use of resources and maximizes the revenue. Hence the offered price (utility) for a session can be expressed in mathematical notations as follows -

Objective: $U = \sum u_i w_i$ is maximized (i.e. maximum revenue will be earned by the MSS).

Where,

u_i = Utility (price) offered for a session

$$w_i = \begin{cases} 0, & \text{if Request } s_i \text{ is rejected} \\ 1, & \text{if request } s_i \text{ is accepted.} \end{cases}$$

3.6 Output of the System

If a multimedia session request is submitted to the server, then the following output will be found-

- Depending upon the availability of resources, the multimedia session request may be accepted or rejected by the admission controller.
- If multimedia session request is accepted then the starting time (S_i) and ending time (E_i) of the session will be computed and it will be informed to the user by the admission controller.

  Starting time : The time when a user starts to get multimedia stream from the server is called the *starting time* (S_i).

  Ending time: The multimedia session will not be stopped on or before the *ending time* (E_i).

3.7 Difficulties of the Problem

We know that the **class P** consists of those problems which are solvable in polynomial time. More specifically, they are problems that can be solved in time $O(n^k)$ for some constant k, where n is the size of the input to the problem. The **class NP** consists of those problems that are "***Verifiable***" in polynomial time. The class **NP-complete** is that problems whose status are unknown. No polynomial-time algorithm has yet been discovered for an NP-complete problem.

NP-complete problems are intractable-without a conclusive outcome [32]. If we can establish our problem as NP-complete, we would spend our time to develop an approximation algorithm.

Proof by restriction is the simplest, and perhaps the most frequently applicable, of different proof types of NP completeness [32].
An NP completeness proof by restriction for a given problem $\Pi \in$ NP consists simply of showing that Π contains a known NP-complete problem Π' as a special case. The heart of such a proof lies in specification of additional restrictions to be placed on the instances of Π so that the resulting restricted problem will be identical to Π'. We do not require that the restricted problem and the known NP-complete problem be exact duplicates of one another, but rather that there be an "obvious" one to one correspondence between their instances that preserves "yes" and "no" answers. This one to one correspondence, which provides the required transformation from Π' to Π, is usually so apparent that it need not even be given explicitly. Now we will consider a decision problem, **Problem 1** as it is NP-complete [32].

Problem 1

Instance: Set T of tasks and for each tasks $t \in T$, a length $l(t) \in Z^+$, a release time $r(t) \in Z_0^+$ and a deadline $d(t) \in Z^+$.

Question: Is there a one-processor schedule for T such that there is a one to one function $\sigma : T \rightarrow Z_0^+$ with $\sigma(t) > \sigma(t) \Rightarrow \sigma(t) \geq \sigma(t') + l(t')$, such that for all $t \in T$, $\sigma(t) \geq r(t')$ and $\sigma(t') + l(t') \leq d(t)$

Theorem 1

Problem 1 is NP-complete [32].

Now we prove our problem to be NP-complete by poof by restriction. In our problem we have n sessions $s_1, s_2, s_3, \ldots s_n$ for a multimedia sever system which has only one server. We may correspond this server to the only processor of the Problem 1 and n sessions correspond to n tasks of set T. Now let us assume a restricted instance of our problem. We have direct correspondence between r_i (ready time) of our problem with $r(t)$ of Problem 1. $d(t)$ of Problem 1 can be mapped to the implicit deadline (d_i+l_i) of our problem. Again we assume that we do not need to maximize the utility (u_i) which effectively means that we just need a schedule of requests. There is a solution of our restricted problem if and only if Problem 1 has a solution. These imply the NP-completeness of our problem.

3.8 Summary

In this chapter we have defined the proposed Multimedia Server System and we discuss the organization of the server, network and clients computers, and requests of customers for multimedia streams and the contents of the request. We also discuss the practicability of the proposed problem and we found that our proposed system is feasible with respect to existing network. The minimum bandwidth required for MPEG1 is 1.5 Mbits/s and this minimum bandwidth may be ensured in the existing POTS by using ADSL technology.

We mathematically formulate our proposed reservation based admission control of a Multimedia Server System problem. We identified the inputs, outputs, constraints, and objectives of the problem. The difficulties of the problem are discussed and we found that our problem is NP-complete. The job of the admission controller is also discussed. The next chapter starts with data structures and algorithms required to solve our problem. We present a new algorithm for reservation based admission control of a Multimedia System.

Chapter 4

Admission Control Algorithms for Reservation based MSS

4.1 Introduction

In this chapter we will describe different steps of reservation schemes for the sessions submitted to MSS. The objective of the reservation based admission control algorithm is to maximize the revenue earned by the system. This will be achieved by admitting more new sessions with higher utilities in the MSS. We present an algorithm for reservation scheme to achieve this objective. The required data structure for this algorithm is also discussed here. This chapter concludes with the analysis of the complexity of admission control algorithms for reservation scheme.

4.2 Admission Control Principle

The admission controller determines weather a new request for a reservation can be satisfied along with the already granted ones. It does this calculation in the following steps:

- *Finding the preliminary accepted and rejected sessions*: This is an initial solution for the problem. A set of preliminary accepted sessions can be defined as subset of submitted sessions, which can be served by the MSS maintaining all the resource constraint at any duration in present and future. The remaining sessions in the list of submitted sessions are called as preliminary rejected sessions.
- *Finding the final accepted and rejected sessions*: This is a suboptimal solution to achieve maximum utility from the sessions. Some of the preliminary accepted or rejected sessions might not be finally accepted or rejected respectively in this step of admission control as it tries to achieve a better solution than the preliminary one.

4.2.1 Preliminary Accepted Sessions

There might be lots of sets that deserve to be preliminary accepted sessions. In our system we define the preliminary accepted as the sessions that follows one of the following properties:

- The new requests for which the required resources are available from ***ready time*** to ***ready time* + *duration***.

- The new requests for which sufficient required resources are not available from ***ready time*** to ***ready time + duration***. But if the ready time of the session is shifted to some extent on or before ***deadline*** then the resource might be available.

When a session is listed as a preliminary accepted session the required resources for this will be reserved although it is not finally accepted. The algorithm of finding preliminary accepted and rejected sessions will be presented in Section 4.5.1.

4.2.2 Examples of Preliminary Accepted and Rejected Sessions

Let us assume that there are 100 units of I/O bandwidth of a VoD server. A batch of sessions s_1, s_2, s_3 are already admitted in the system. The resources available for different time duration are shown in Figure 4-1.

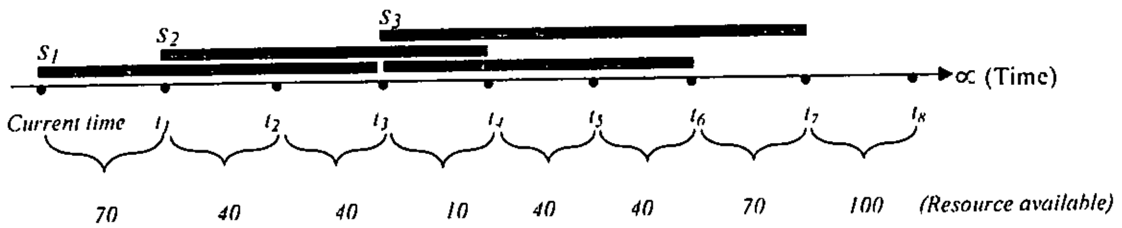


Figure 4-1: Already admitted sessions in a VoD system

Now an epoch of three sessions s_4, s_5, s_6 arrived at the server for reservation. For simplicity we assume that each session requires 30 units of resources.

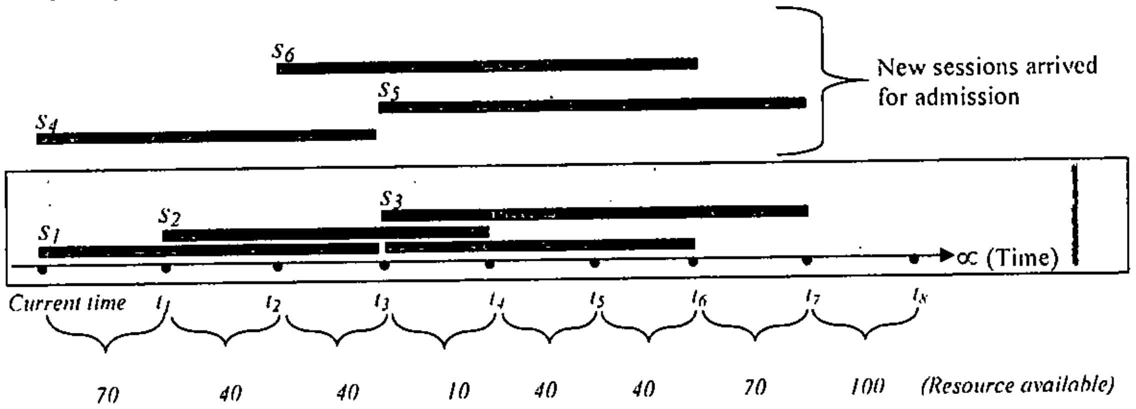
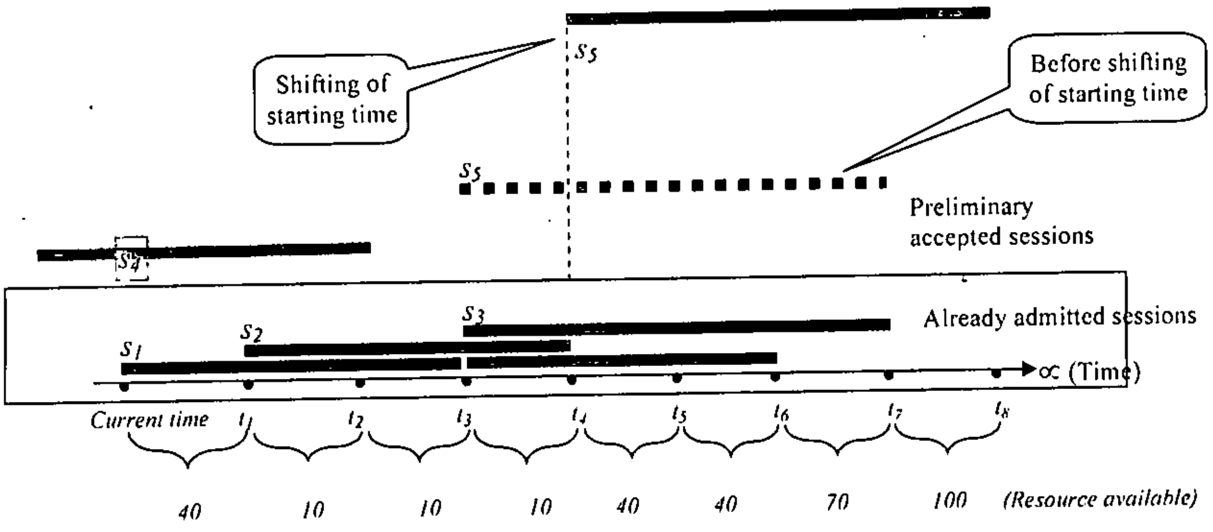


Figure 4-2: Arrived new sessions for admission and already admitted sessions in a VoD system

Session s_4 has required bandwidth (30 unit) hence it is listed as preliminary accepted session and its bandwidth will be reserved.

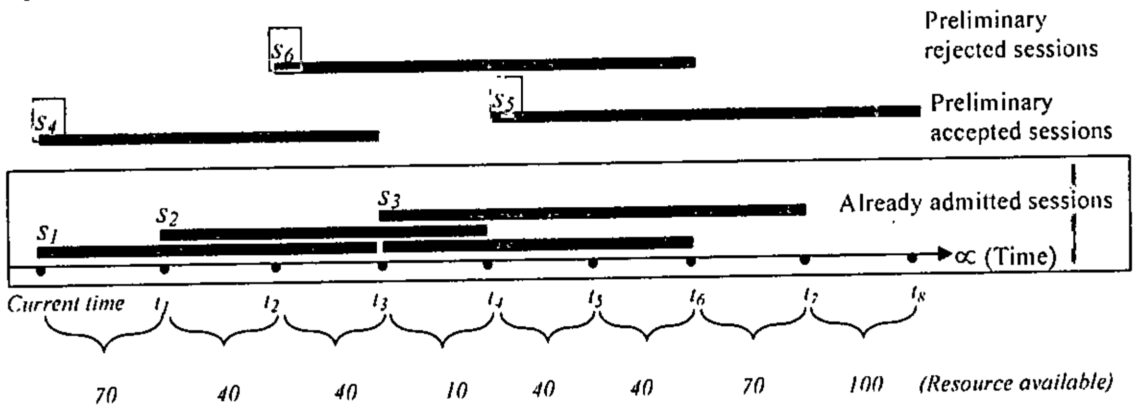
The required bandwidth (30 units) of session s_5 is not available but if the starting time is shifted t_3 to t_4 (on or before the deadline set by the customer) then the required bandwidth will be available. Hence it is listed as preliminary accepted session and its bandwidth will be also reserved. The shifting of starting time will be limited by the deadline given by the customers. The following diagram illustrates the details of this preliminary accepted session which is adjusted by shifting the starting time.



Shifting of starting time
s_5
Before shifting of starting time
s_5
Preliminary accepted sessions
s_4
Already admitted sessions
s_3
s_2
s_1
∝ (Time)
Current time
t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8
40 10 10 10 40 40 70 100 (Resource available)

Figure 4-3: Shifting the starting time of a session

The required bandwidth (30 units) of session s_6 is not available even if the starting time is shifted t_2 to t_3 (on or before the deadline). Hence it is listed as preliminary rejected session. The following Figure 4-4 illustrates the preliminary accepted and preliminary rejected sessions.



Preliminary rejected sessions
s_6
s_5
Preliminary accepted sessions
s_4
Already admitted sessions
s_3
s_2
s_1
∝ (Time)
Current time
t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8
70 40 40 10 40 40 70 100 (Resource available)

Figure 4-4: Preliminary accepted and rejected sessions in a MSS

4.2.3 Sessions with Different States in a MSS

In the Multimedia System we have the following types of users:

(i) Some users who are enjoying multimedia sessions,

(ii) Some users who will enjoy the session in future as they have got admissions,

(iii) Some users who are requesting for reservations of new sessions.

Previously scheduled session or finally admitted sessions have fixed starting time and end time. These finally admitted sessions would not be re-scheduled. Starting time of some of the finally admitted sessions that have not started getting multimedia stream may be changed. But if this is done, the user may get bother. That is why finally admitted sessions will not be considered for modification. The users who have left the system after enjoying the session are not considered though they have released some resources.

Only the new requests, which are batched for admission, will be considered in the reservation process. Depending upon the availability of required resources, the new requests are listed as preliminary accepted sessions and preliminary rejected sessions. The following diagram illustrates the different sessions of MSS.

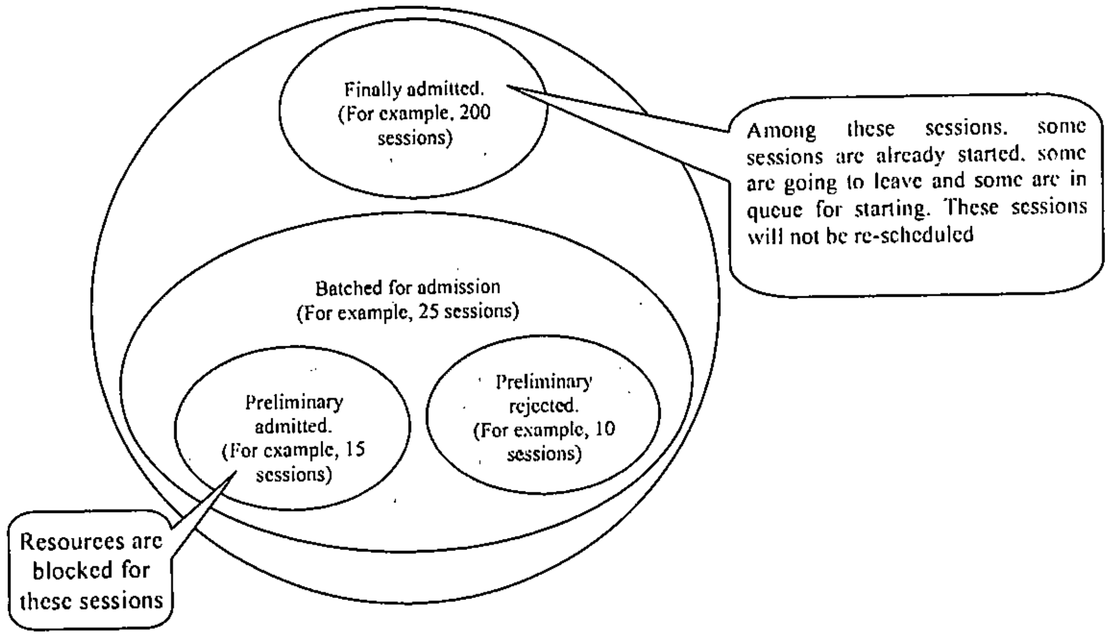

Finally admitted. (For example, 200 sessions)
Among these sessions, some sessions are already started, some are going to leave and some are in queue for starting. These sessions will not be re-scheduled
Batched for admission (For example, 25 sessions)
Preliminary admitted. (For example, 15 sessions)
Preliminary rejected. (For example, 10 sessions)
Resources are blocked for these sessions

Figure 4-5 : Different sessions in a MSS

4.3 Data Structures for Presenting Sessions

For the admission controller we need to design data structures to perform the following tasks:

- To specify the session (i.e. starting time and duration of the session) and the amount of resource required (i.e. Bandwidth) during the session
- To keep track of available bandwidth at a particular time interval, as the resource in the MSS is actually the bandwidth at a particular duration.
- To store the list of sessions with different states

We know that the tree data structures are very much efficient for searching, insertion and deletion of elements. Segment tree [33] is a balanced binary tree data structure that is used for tracking intervals.

4.3.1 Segments of the Session

Available bandwidth at a particular duration specified by a start and end point depends on the accepted sessions on that particular duration. For the reservation based admission controller we have to consider for the future from the current instant of time. That is why we can think of a *total time duration* starting from the *current time* to $\propto$. The starting and end points of a session may divide the total time durations into several parts. Each of these parts is called a *segment*. A session may consist of one or more segments.

The available bandwidth is different at different time as the admitted sessions are scheduled to start or finish at different times. The available bandwidth changes with the introduction of a new session at any time. When new sessions are accepted the length of the segment are updated and some new segments may be created. Accordingly the available resources will also be updated.

Let us assume that we have some session s_i that has a starting time S_i, ending time E_i ($S_i<E_i$) and resource consumed b_i. Total Bandwidth of the system is B ($B\geq\Sigma b_i$). The sessions may be described as $s_1(S_1, E_1, b_1)$, $s_2(S_2, E_2, b_2)$, ... , $s_n(S_n, E_n, b_n)$ etc.

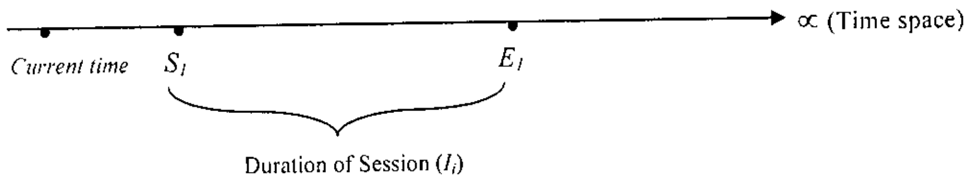


Figure 4-6 : Segmentation of time space

If only one session $s_l(S_l, E_l, b_l)$ is admitted in the server then the time segments will be (*Current time*.S_l, B), (S_l, E_l, B–b_l) and (E_l, ∝, B).

Now, let us consider a new multimedia session request s_l' is submitted to the server showned in the Figure 5.5. The starting time, ending time, and resource consumed of the new session s_l' are described by S_l', E_l', b'_l *respectively* where $S_l < S_l' < E_l$ and $E_l < E_l' < ∝$.

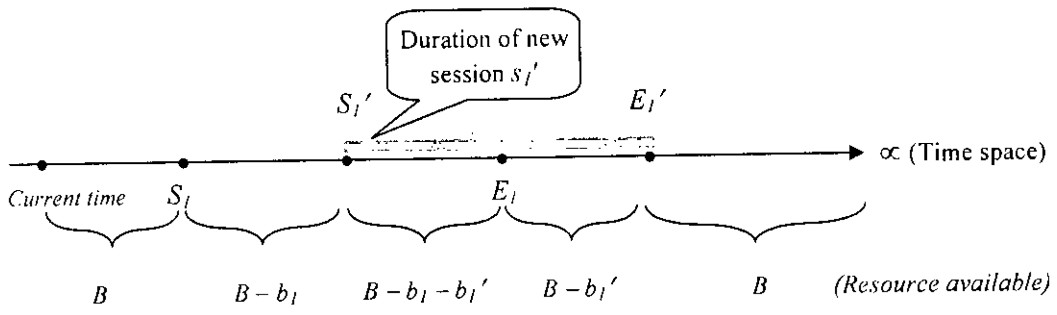


Figure 5.5

Figure 4-7 : Modification segments

Then the *segment* will be updated as (*Current time*, S_l, B), (S_l, S_l', B–b_l), (S_l', E_l, B-b_l-b_l'), (E_l, E_l', B–b_l') and (E_l', ∝, B).

4.3.2 Definition of Segment Tree

The segment tree structure, introduced by Bentley [33], is a balanced binary tree data structure that is used to store segments or intervals.

For two numbers s and t, with $s < t$, the segment tree $V(s, t)$ is recursively constructed as follows:

(i) It consists of the following-

- a root V, with parameters $B[V] = s$, and $E[V] = t$ indicating the start and end of an interval
- a left subtree $T(s, m)$ and a right subtree $T'(m, t)$, where $s<m<t$ as shown in the Figure 4-8.

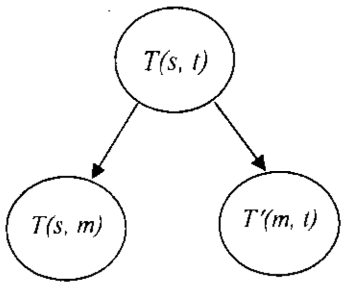


Figure 4-8 : Segment tree

(ii) The parameters $B[V]$ and $E[V]$, define an interval $[B[V], E[V]]$, called a standard interval associated with V.

(iii) The standard interval associated with a leaf node is called an elementary interval in left to right order.

(iv) Each internal node u with children v and v' is associated with an interval I_u. The interval I_u is the union of the intervals of its two children, $I_u = I_v \cup I_v'$ which is shown in the Figure 4-9.

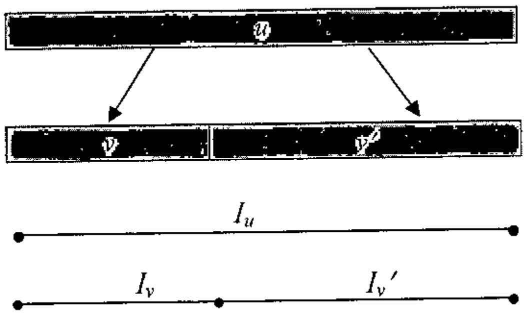


Figure 4-9 : Internal node of a segment tree

4.3.3 Segment Tree to Incorporate Multimedia Sessions

The segment tree is a rooted binary tree and it is a balanced tree. We can map our reservation based admission control problem of MSS to the segment tree with a little modification. The s and t, with $s < t$, of the segment tree $V(s, t)$ can be mapped into the start time and end time of a session where start time < end time. We modify the traditional segment tree. We add a field (available bandwidth of a segment) to each leaf node. Thus the contents of each leaf node in the segment tree with this modification are as follows:

- Starting time
- Ending time
- Bandwidth available (between the starting time and ending time)

Leaves of the segment tree contain all the segments and the available bandwidth. The internal node of the tree contains only the interval of its child node.

Let us consider a scenario of a segment tree where we assume current time is zero(0) and total bandwidth is B. A session of starting time (50), ending time (150) and bandwidth (50) is admitted. Root contains the segment (0, ∝). Root's left node contains (0, 50, B) and root's right node contains (50, ∝) segment. Root's right left leaf node contains (50, 150, B−50) segment and root's right right leaf node contains (150, ∝, B).

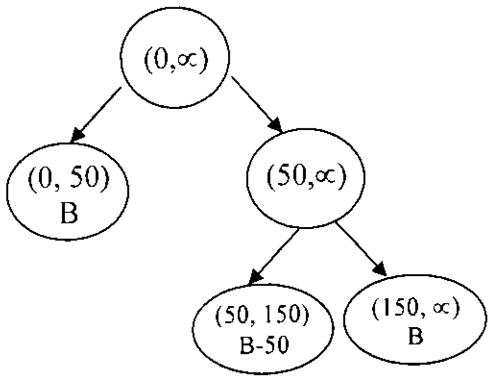


Figure 4-10 : Construction of a segment tree

Now let us consider another case that a second session of starting time (25), ending time (50) and Bandwidth (4) is admitted. In this case the starting time and ending time of the session lie on one node, Hence the updated segment tree will be as follows-

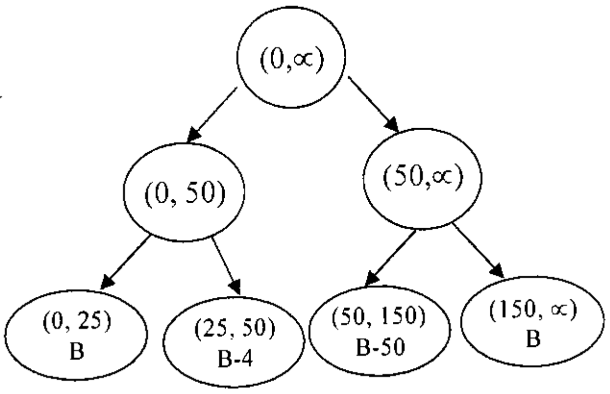


Figure 4-11: Inserting a session into the segment tree

Now consider a third session of starting time (30), ending time (180) and Bandwidth (5) is admitted. Here the starting time and ending time of the session lie on different node. Now the updated segment tree will be as follows.

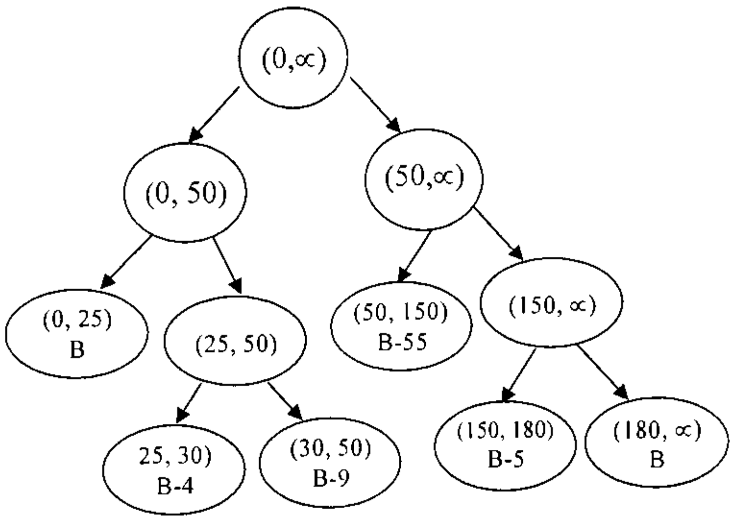


Figure 4-12: Updating a Segment tree

If a session arrived with starting point and ending point coincided with the existing segment's starting point and ending point (that means starting point is coincided with the segment's starting point and ending point is coincided with the segment's ending point) then the height of the tree will not be increased. The corresponding leaf nodes' bandwidths will be updated only in this case.

4.3.4 Operations of Segment Tree

There are four types of operations that will be performed on the segment tree in order to do reservations. They are as follows-

(i) Insertion (inserting a session into the segment tree),

(ii) Deletion (deleting a segment from the segment tree),

(iii) Searching (searching of segment into the segment tree) and

(iv) Shifting of starting time of a session (shifting involves one delete and one insert operation).

4.3.5 Algorithm and Complexity of Insert Operation in a Segment Tree

Inserting a session into the segment tree involves the following tasks-

(i) Creating necessary segments for the starting point of the session,

(ii) Creating necessary segments for the end point of the session and

(iii) Updating the available bandwidth of the segments covered by the session.

Procedure *create_segment*(*root, pt*)

/* This is a recursive procedure for inserting a session into the modified segment tree. *pt* might be start or end point and *midpt* is the point at which the node is divided.*/

```
1.  if(root→ left and root→ right both are null) then                // leaf node
2.      if(root→ start < pt < root → end) then
                        // The point is in between the start and end pt of the leaf segment
3.          root → midpt = pt;
4.          create_node(root, LEFT, root →start, pt)            // Create left subtree
5.          create_node(root, RIGHT, pt, root → end))           // Create right subtree
6.      else
7.          no action
            /* no need to update the tree, the pt coincide with existing start and end point
            or it is out of the range of the segment.*/
8.      endif
9.  else
```

```
10.     if(pt< root → midpt) then                     // The point is at the left of the node
11            create_segment(root→ left, pt)          // Searching in the left sub trees
12.     else
13.           if(pt> root → midpt) then               // The point is at the right of the node
14.                 create_segment(root → right, pt)  // Searching in the right sub trees
15.           endif
16.     endif
17.  endif
end create_segment.
```

Procedure *create_node*(*root, sibling_direction, start, end*)

```
/* This is the procedure for creating a node into the segment tree. */
1.      create a node with start, end, left = null, right = null
2.      root→ sibling_direction = node        // Creating the link between parent and child
end create_node.
```

Cost of Insertion into segment tree: Insertion of a session into segment tree consists of three phases.

- In the first phase, the starting point or end point of the session is searched into the segment tree. Worst case searching cost of a segment tree is $O(h)$, where h is the average height of the segment tree.
- In the second phase, segment is created if necessary. Cost of creating a new node into the segment tree is $O(1)$.
- In the third phase, the bandwidth fields of the leaf node in between start point and end point of the session is updated. Updating involves finding all the existing segments in between the starting point and end point of the session. Thus updating cost involves the search cost. of $O(h)$.

Hence the worst case complexity of insert operation of a session into the segment tree is $O(h)$ where h is the average height of the segment tree.

4.3.6 Algorithm and Complexity of Delete Operation in a Segment Tree

```
Procedure delete_session(root, startpt, endpt)
/* This is a recursive procedure for deleting a session from the modified segment tree. */

1.  if(startpt, endpt) and (root→ start, root→ end) does not coincide then

/* The duration specified in the node does not coincides with the duration between startpt and
endpt */

2.          return
3.  endif
4.  if(root→ left and root→ right both are null) then                          // leaf node
5.          if(startpt  ≤ root→ start < root → end ≤ endpt) then
                                    // The leaf is one of the segment of the session duration
6.                  update available_bw
7.                  if(available_bw = total_bandwidth) then
8.                          dispose_node(root →parent)
                              // Deleting the leaf node and destroying its link with the parent
9.                  endif
10.         endif
11. else
12.         delete_session(root → left, startpt, endpt)        // Searching in the left sub trees
13.         delete_session(root → right, startpt, endpt)       // Searching in the right sub trees
14. endif
end delete_session.
```

Cost of Delete operation into segment tree: Cost of deletion of an interval from a segment tree is analyzed in similar way of insertion operation that is described in the previous section. The cost of deletion of an interval from the segment tree is $O(h)$ where h is the average height of the segment tree.

4.4 Determination of Finally Accepted and Rejected Sessions

The multimedia session request for booking a multimedia session is defined by s_i. It contains a *ready time* r_i, a *deadline* d_i, duration of the *session* l_i, required *bandwidth* of the session b_i and *utility* of the session u_i.

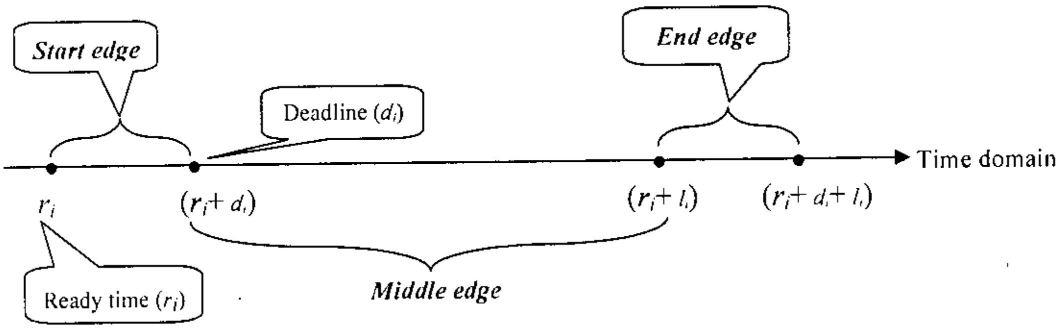


Figure 4-13: Different edge of user request

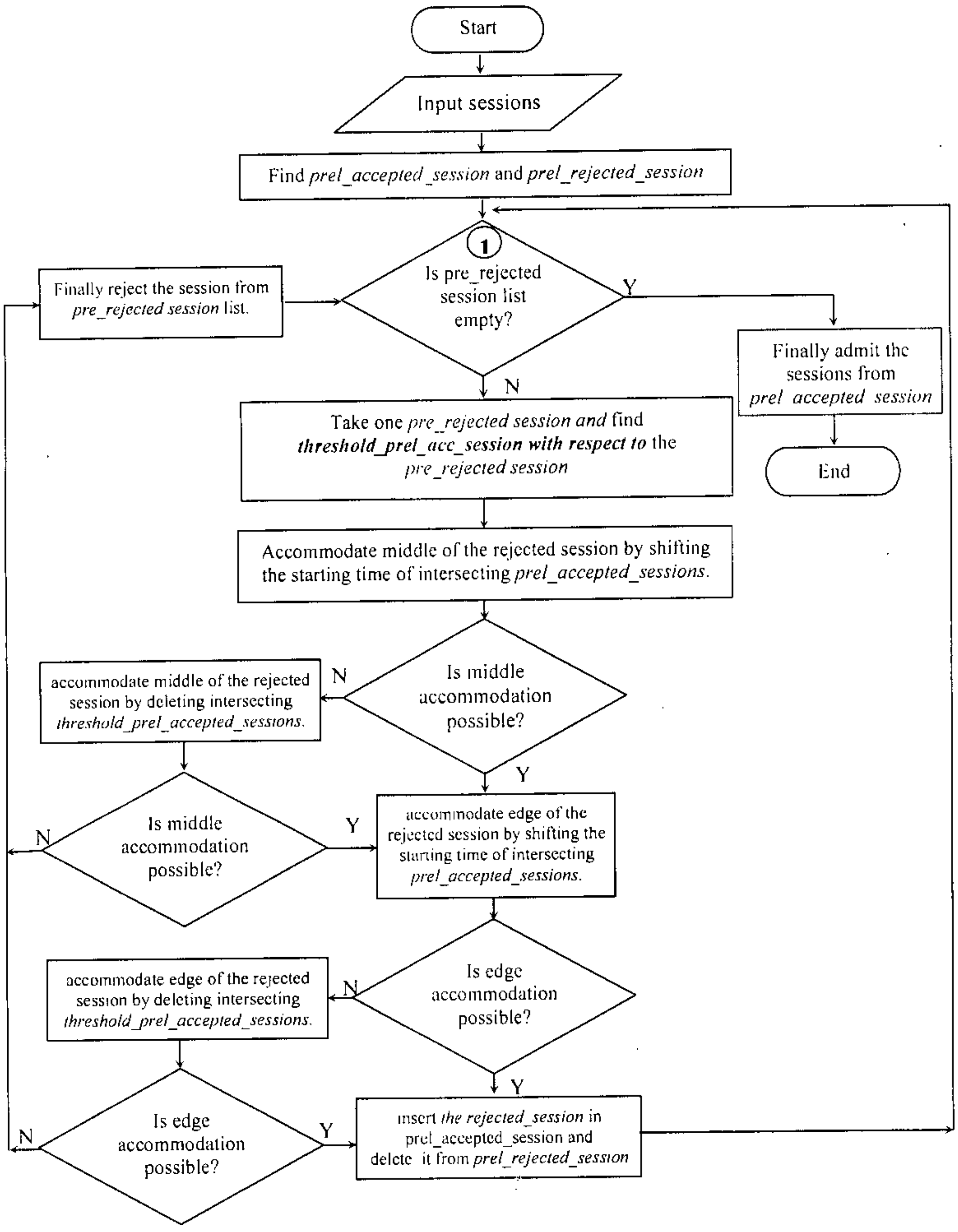
A multimedia session request can be splited into three parts, which are *start edge segment*, *middle segment* and *end edge segment*. The segment of *ready time* to *deadline* of a multimedia session request shown in Figure 4-13, is *start edge segment*. From *ready time+deadline* to *ready time+duration* of a multimedia session request shown in Figure 4-13, is *middle segment* and from *ready time+duration* to *deadline+duration* is *end edge segment*.

We have to determine preliminary accepted session *(prel_accepted_session)*, preliminary rejected session *(prel_rejected_session)* and threshold preliminary accepted session *(threshold_prel_acc_session)* before finalizing admitted and rejected sessions.

The *prel_accepted_session* are those sessions which have been accepted temporary and bandwidth has been reserved for those sessions. The *prel_rejected_session* are those sessions which have been rejected temporary and Bandwidth is not available for these sessions. The procedures for determining preliminary accepted and preliminary rejected sessions are described in Section 4.5.1.

The *threshold_prel_acc_session* are the subset of preliminary accepted sessions where the utility of each session is less than the utilities of a preliminary rejected session. Actually it is required to find the preliminary accepted sessions of lower utility with respect to the rejected session so that more profitable rejected sessions are to be accommodated instead of low utility preliminary accepter sessions.

The following flow chart describes the selection procedures of finally accepted and rejected session of a Multimedia Server System.



Start
Input sessions
Find *prel_accepted_session* and *prel_rejected_session*
1
Is pre_rejected session list empty?
Y
N
Finally reject the session from *pre_rejected session* list.
Finally admit the sessions from *prel accepted session*
End
Take one *pre_rejected session and* find ***threshold_prel_acc_session with respect to*** the *pre_rejected session*
Accommodate middle of the rejected session by shifting the starting time of intersecting *prel_accepted_sessions*.
Is middle accommodation possible?
accommodate middle of the rejected session by deleting intersecting *threshold_prel_accepted_sessions*.
Is middle accommodation possible?
accommodate edge of the rejected session by shifting the starting time of intersecting *prel_accepted_sessions*.
Is edge accommodation possible?
accommodate edge of the rejected session by deleting intersecting *threshold_prel_accepted_sessions*.
Is edge accommodation possible?
insert *the rejected_session* in prel_accepted_session and delete it from *prel_rejected_session*

Figure 4-14 : Flow chart of selecting finally accepted and rejected sessions.

4.5 Algorithms of Different Steps of Selection

Our proposed heuristic Reserve-HEU consists of the following two steps-

- Finding the preliminary accepted and rejected sessions. This is an initial solution for the problem.
- Finding the final accepted and rejected sessions. This is a suboptimal solution to achieve maximum utility from the sessions.

Following are the variables and procedures to describe the steps of algorithm.

sessions:	A batch of session requests made by the customer for future reservation. We called it an epoch.
rejected_session:	A session from the queue of *prel_rejected_session* that is trying to be accommodates.
utility(s_i) :	A function returning the utility of a session s_i.
ready_time(s_i) :	A function returning the earliest start time of a session s_i.
bw(s_i) :	A function returning the required Bandwidth of a session s_i.
available_bw(*begin*, *end*) :	Available bandwidth of a segment from *begin* to *end.*
is_available(*bw*, *segment*) :	A function that checks the *bw*(bandwidth) is whether available in the *segment* or not.
deadline(s_i) :	A function returning the deadline of a session s_i.
duration(s_i) :	A function returning the duration of a session s_i.
middle(s_i) :	A *segment* from [*ready time*+ *deadline*] to [*ready time* + *duration*] of a session s_i
start_edge_segment(s_i):	A segment from *ready time* to *deadline* of a session s_i
end_edge_segment(s_i):	A *segment from* [*ready time*+ *duration*] to [*deadline* + *duration*] of a session s_i
intersection(s_1, s_2) :	This function returns the intersecting segments of two given session s_1 and s_2.

4.5.1 Algorithm for Finding Preliminary Accepted and Rejected Sessions

Procedure *preliminary_solution*

/* This is the procedure for selecting preliminary accepted and preliminary rejected sessions. */

```
1.  batch_of_sessions ← scan for new requests                    // Arrival of an epoch
2.  sessions ← sort batch_of_sessions                   // Sort according to their utility
3.  do while sessions ≠ empty
4.  c_session ← pop from sessions
5.      if(bw(c_session) <= available_bw(c_session)) then
                                               // Bandwidth is available for the session
6.          prel_accepted_session+ = c_session
7.      else if(shift_starting(c_session) = true) then
                                          // Session can be accommodated if shifted
10..        prel_accepted_session + = c_session;
14.     else
15.         prel_rejected_session + = c_session;
                                         // Session could not be accommodated anyhow
17.     endif
18. end do
end procedure
```

Procedure *shift_starting(c_session)*

/* This procedure finds weather a given session's starting time can be shifted or not. It also checks whether there is available bandwidth if shifted. */

```
1.      if(bw(middle(c_session))<= available(middle(c_session))) then
2.          if(find_suitable_start_end_pt(c_session) = true) then
3.              return true
4.          endif
5.      else
```

```
6.              return false
7.          endif
end procedure
```

Procedure *find_suitable_start_end_pt(c_session)*

```
/* This procedure finds suitable start time and end time of a given session */
1.  start_edge_segments = find_segments(ready_time(c_session), deadline(c_session));
//  start_edge_segment is a list of segments in the descending order of occurrence
2.  earliest_start = deadline(c_session)
3.  for each edge_segment in start_edge_segments do
4.      if available_bw(edge_segment) >= bw(c_session) then
5.          earliest_start = start(edge_segment)
        // Finding the earliest start possible for the session satisfying the bandwidth constraint.
6.      else
7.          break from the for loop;
8.      endif
9.  end for
10. end_edge_segment = find_segments(ready_time(c_session)+duration(c_session),
    deadline(c_session) + duration(c_session))
//  end_edge_segment in ascending order of occurrences.
11. latest_end = ready_time(c_session) + duration(c_session)
12. for each edge_segment in end_edge_segments do
13.     if available_bw(edge_segment) >= bw(c_session) then
14.         latest_end = end(edge_segment)     /* Finding the latest end time possible
for the session satisfying the bandwidth constraints */
15.     else
16.         break from the for loop;
17.     endif
18. end for
//  Checking the boundary conditions
19. if (earliest_start < ready_time(c_session)) then
20.     earliest_start = ready_time(c_session)
21. endif
```

```
22.  if ($latest\_end > deadline(c\_session) + duration(c\_session)$) then
         $latest\_end = deadline(c\_session) + duration(c\_session)$
23.  endif
24.  if ($latest\_end - earliest\_start$)>= $duration(c\_session)$ then
                                                   // The bw is sufficient for the session
         $start(c\_session) = earliest\_start$
25.      modify the start time of $c\_session$;
26.      insert $c\_session$ in the segment tree;
29.      return true
30.  else
31.      return false
32.  endif
end procedure
```

4.5.2 Algorithm for Finding Finally Accepted and Rejected Sessions

The following algorithm determines the finally accepted and rejected sessions-

Procedure *accomodate_session*

/* This is the main procedure to accommodate sessions using heuristic. */

```
1.   do
2.       while $prel\_rejected\_session \neq$ empty
3.       $rejected\_session \leftarrow$ pop from $prel\_rejected\_session$
4.       $threshold\_prel\_acc\_session \leftarrow$ The subset of $prel\_accepted\_session$ where
         the utility of each session is less than the $rejected\_session$
5.       sort $threshold\_prel\_acc\_session$ in the ascending order of utility
6.       if ($accomadate\_rej\_session(rejected\_session)$ = false)
         /* Trying to accommodated by adjusting the preliminary accepted session */
7.           $finally\_rejected\_session+ = rejected\_session$
8.       else
9.           $prel\_accepted\_session+ = rejected\_session$
10.          $prel\_rejected\_session-- = rejected\_session$
11.      endif
12.  end do
13.  $finally\_accepted\_session+ = prel\_accepted\_session;$
end procedure
```

Procedure *accomodate_rej_session*(*rejected_session*)

/* This procedure is used to accommodate preliminary rejected sessions. */

```
1.  save all the states of sessions, prel_accepted_session, threshold_prel_acc_session,
    rejected_session
2.  limit_of_delete = utility(rejected_session)
3.  if (accomodate_middle_by_shifting(rejected_session) = false) then
                                  // Accomodation of middle by adjustment is not possible
4.      deleted = accommodate_by_deleting(rejected_session, limit_of_delete, position)
                                  // Trying by deleting preliminary accepted session
5.      if (deleted > limit_of_delete) then unsave the state and return false
                                  // Previous solution is better than the current situation
6.  endif
7.  limit_of_delete = limit_of_delete –deleted
8.  if (find_suitable_start_end_pt(rej_session) = false) then
                                  // Accomodation of edge by adjustment is not possible
9.      deleted = accommodate_by_deleting(rejected_session, limit_of_delete, edge_segment)
                                  // Trying by deleting preliminary accepted session
10. if (deleted<0) then unsave the state and return false;
                                  // Previous solution is better than the current situation
11. endif
12.     return deleted
end procedure
```

Procedure *accommodate _by_deleting*(*rejected_session, limit_of_delete, position*)

/* This procedure is used for accommodating the specified part (*middle, start_edge_segments, or end_edge_segments*) of a session by deleting other less utility preliminary accepted sessions. */

```
//  Preliminary accepted sessions are sorted according to the utility provided.
//  position may be middle, start_edge_segment, or end_edge_segment
1.  deleted = 0;
2.  for each p_accept in prel_accepted_session do
3.      if (deleted > limit_of_deleted) return deleted;
                                  // Deleting is no more earning better revenue
```

```
4.      if(intersection(p_accept, position(rejected_segment)) ≠ null) then
                    // Rejected session does not coincide with the preliminary accepted session
5.          if (deleted + utility(p_accept)) <limit_of_deleted) then
                                                                        // Deleting is profitable
6.          delete p_accept
7.          deleted + = utility(p_accept)
8.          prel_rejected_session + = p_accept
                        // Preliminary accepted session is being rejected preliminarily
9.          prel_accepted_session − = p_accept
10.         endif
11.     endif
12.     if is_ available(bw(rejected_session), position(rejected_session)) then
                                        // Preliminary rejected session has been accepted
13.         finally_rejected_session − = rejected_session;
            prel_accepted_session + = rejected_session;
14.     return deleted
15.  endif
16.  end for
end procedure
```

Procedure accomodate_middle_by_shifting(*rejected_session****)***

/* This procedure is used for accommodating the middle portion of a preliminary rejected session by shifting preliminary accepted sessions. */

```
1.   if is_available(bw(rejected_session), middle(rejected_session)) = true then return true;
                                                        // else bandwidth is not available
2.   for each p_accept in prel_accepted_session do
3.      if intersection(middle(rejected_session), duration(p_accept)) ≠ NULL then
                    // Checking whether the preliminary accepted sessions coincide or not
4.          shift_starting(p_accept) to make bandwidth free in middle(rejected_session)
                        // Trying to shift starting point of the preliminary accepted session
5.          if is_available(bw(rejected_session), middle(rejected_session)) = true then
6.              return true;                    // The middle has been accommodated
7.          else
8.              return false;                   // The middle could not be accommodated
```

```
9.                endif
10.           endif
11.     end for
end procedure
```

4.5.3 Complexity Analysis of Reserve-HEU

Complexity of the reservation process depends on the cost of insertion and deletion of sessions, and searching of resource into segment tree. These operations mostly require comparisons, additions and subtractions. We assume the same complexity of these basic operations (comparison, addition and subtraction) in our analysis. Here we present the complexity analysis of the reservation process of admission controller in terms of these basic operations. We know that the reservation process takes place in two steps: the first step is finding the initial solution for the problem; and the second step is finding the suboptimal solution to achieve maximum utility.

Complexity analysis for the first phase:

In this phase, we find the major cost involved in sorting the sessions, searching an element in segment tree, inserting or shifting of starting point of the sessions in the segment tree. The complexities of these operations are as follows:

- The average cost for sorting n sessions according to the utility using quick sort is $O(n\log_2 n)$ comparisons.
- The searching cost of a segment in the segment tree is $O(h)$ comparisons, where h is the average height of the segment tree. It is worth mentioning that the height of the tree is dependent on the number of nodes in the segment tree, which is a function of number of sessions in the MSS.
- Insertion of a session into the segment tree requires the following: (i) finding the right segment where the session be inserted and (ii) performing insertion by creating nodes and updating bandwidth. Searching for right segment requires $O(h)$ comparisons whereas creation of nodes and updating bandwidth requires $O(1)$ addition and subtraction operations.
- Cost of shifting needs one deletion and one insertion. In Section 4.3.6, We found that the deletion cost is $O(h)$ comparisons. Thus shifting cost is $O(h)$.

Consider the situation when total number of preliminary admitted sessions is i. Now the size of segment tree in the worst case is $(4i+1)$ as both starting and end point of the session might be situated at the middle of two segments creating four new segments in the segment tree. Thus the average height of the segment tree would be $\log_2(4i+1)$. So, searching cost of finding preliminary acceptance of $(i+1)$st session would be $\log_2(4i+1)$.

Thus the total number of comparisons for finding preliminary accepted and rejected sessions is $= \sum_{i=1}^{n} \log_2(4i+1) \approx \sum_{i=1}^{n} \log_2(4i) = \sum_{i=1}^{n} (\log_2 4 + \log_2 i) = \sum_{i=1}^{n} (2 + \log_2 i)$

$$= 2n + \sum_{i=1}^{n} \log_2 i = 2n + \sum_{i=1}^{n} \log_2 i = 2n + \log_2 n!$$

$= 2n + n\log_2 n$ [Using Stirling's approximation, $n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$ [34]

$\approx n\log_2 n$

Thus the complexity of the first phase is $O(n\log_2 n)$

Cost analysis for the second phase:

Now we will analyze the worst case cost of the second phase of reservation scheme. Let us assume that the number of preliminary accepted sessions is p and the number of preliminary rejected sessions is q. Hence the total number of sessions is $(p+q)$. In this situation the worst case cost for accommodation of rejected sessions is elaborated as follows:

- Finding the *threshold_prel_acc_session* (the preliminary accepted sessions whose utility is less than the preliminary rejected session) is done by sorting the preliminary accepted sessions once. It requires $O(p\log_2 p)$ comparisons in the worst case.
- The cost for accommodation of the rejected sessions into the p preliminary accepted sessions are as follows:
    - It requires $O(p)$ comparisons to find intersections between p preliminary accepted sessions and a preliminary rejected session to be incorporated. Thus for q rejected sessions we need $O(pq)$ comparisons.

- It requires $O(ph)$ complexity (comparison, addition and subtraction) for shifting and deleting of maximum p sessions during the accommodation of q rejected sessions.
- It requires $O(pqh)$ complexity (comparison, addition and subtraction) for retaining all the p preliminary accepted sessions to their previous positions by inserting and shifting.

Thus the worst case complexity for accommodation of a rejected session is $O(p\log_2 p) + O(pqh)$. If we consider $(4p+1)$ nodes for a segment tree with p accepted sessions then the height of the segment tree $h = \log_2(4p+1)$. Thus the complexity can be expressed as $O(pq\log_2 p)$, which is $O(n^2\log_2 n)$ if $p = q = n/2$.

4.6 Summary

In this chapter we described different steps, the data structures and algorithms for reservation based admission control of a Multimedia Server System. The admission controller classify new sessions into preliminary accepted sessions or preliminary rejected sessions. After that the admission controller finalizes the admission process according to the proposed algorithms based on maximization of revenue of the system. We also analyze the computational complexity of the proposed algorithms.

In the next chapter we will describe another solution of our problem, which is known as Knapsack problem approach. We mapped our problem into MMKP, a variant of Knapsack problem and finally we solve it by applying the heuristic of Knapsack problem.

Chapter 5

Reservation Based Admission Control and MMKP

5.1 The Knapsack Problem

The classical 0-1 Knapsack Problem (KP) is to pick up items for a knapsack for maximum total values, so that the total resources required does not exceed the resource constraint R of the knapsack. The classical 0-1 KP and its variants are used in many resource management applications such as cargo loading, industrial production, menu planning and resource allocation in multimedia servers.

The classical 0-1 KP may be explained as follows. Let there be n items with values $v_1, v_2, v_3, \ldots, v_n$ and let the corresponding resources required to pick the items be $b_1, b_2, b_3, \ldots . b_n$ respectively. The items can represent services and their associated values can represent the revenue earned from that service. In mathematical notation, the 0-1 Knapsack Problem is to find-

V = maximize $\sum_{i=1}^{n} x_i v_i$, subject to the constraint $\sum_{i=1}^{n} x_i b_i \leq B$(resource constraint)

and $x_i \in \{0, 1\}$is the picking variable.

5.1.1 Multidimensional Multiple-choice Knapsack Problem (MMKP)

There are variants of classical 0-1 Knapsack Problems. They are Multi Dimensional Knapsack Problem (MDKP), Multiple Choice Knapsack Problem (MCKP), Multidimensional Multiple-choice Knapsack Problem (MMKP). The Multidimensional Knapsack Problem (MDKP) is one kind of KP where the resources are multi dimensional, i.e. there are multiple resource constraints for the knapsack. The Multiple Choice Knapsack Problem (MCKP) is another KP where the picking criteria for items are restricted. In this variant of KP there are one or more groups of items and only one resource constraint. Exactly one item will be picked from each group. The MMKP is a combination of the MDKP and the MCKP.

Let there be n *groups of items*. Group i has l_i items. Each item of the group has a particular value and it requires m resources. The objective of the MMKP is to pick exactly one item from each group for maximum total value of the collected items, subject to m resource constraints of the knapsack. In mathematical notation, let v_{ij} be the value of the j th item of the i th group, $\bar{b}_{ij} = (b_{ij1}, b_{ij2}, \cdots, b_{ijm})$ be the required resource vector for the j th item of the i th group and $\bar{B} = (B_1, B_2, \cdots, B_m)$, be the resource bound of the knapsack. Now, the problem is to find-

$V =$ maximize $\sum_{i=1}^{n}\sum_{j=1}^{l_i} x_{ij}v_{ij}$ (objective function), so that, $\sum_{i=1}^{n}\sum_{j=1}^{l_i} x_{ij}b_{ijk} \le B_k$ (resource constraints), where, V is the *value of the solution*, k = 1, 2,..., m, $x_{ij} \in \{0,1\}$ are the picking variables, and $\sum_{j=1}^{l_i} x_{ij} = 1$.

Figure 5-1 illustrates an MMKP. We have to pick exactly one item from each group. Each item has two resources, b_1 and b_2. The objective of picking items is to maximize the total value of the picked items subject to the resource constraints of the knapsack, that is $\sum(b_1$ of picked items$) \le 17$ and $\sum(b_2$ of picked items$) \le 15$.

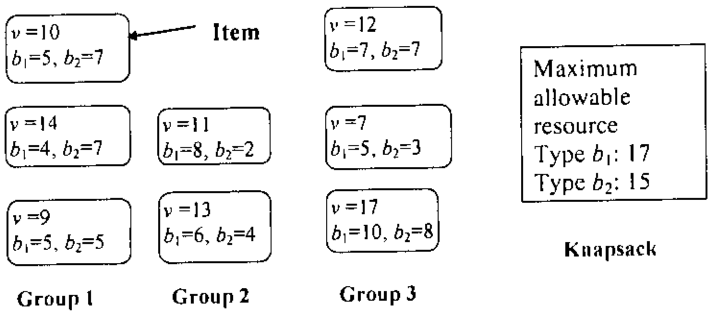


Figure 5-1: Multidimensional Multiple-choice Knapsack Problem.

5.1.2 Solutions of MMKP

The 0-1 Knapsack Problem is an NP-Hard problem [5]. The variants of KP (i.e. MCKP, MDKP and MMKP) are as hard as KP. So these are also NP-Hard problem. There are two methods of finding solutions for an MMKP: one is a method for finding exact solutions and the other is heuristic solution.

The computation time for any exact algorithms of MMKP may grow exponentially with the size of the problem instance in the worst case [5]. Hence this may not be acceptable for time-critical applications such as admission control and dynamic resource allocation in a Multimedia Server System. These applications are forced to accept a near-optimal solution if the computational time for optimal solution exceeds real-time bounds.

HEU is a heuristic developed by Khan [2] to find near optimal solutions of MMKP. HEU achieves 94% optimal solution in $O(mn^2l^2)$. where n = number of groups, l = number of items in each group (assumed constant for convenience of analysis) and m = resource dimension.

The main steps of the HEU are as follows-

- The items of each group are sorted in nondecreasing order according to the value associated with them and it selects the lowest valued items from each group as the initial solution. It then upgrades the solution gradually by choosing new items as long as the solution remains feasible.
- Here the main idea is to penalize the use of resources depending on the current resource state. It applies a large penalty for a heavily used resource, and a small penalty for a lightly used resource.
- The aggregate resource of the *j*th item of the *i*th group is defined by $a_{ij} = \sum_k b_{ijk} \times C_k \Big/ |C|$, where C_k = amount of the *k*th resource consumption and $|C| = \sqrt{\sum C_k^2}$.
- To find the next item to be picked, it chooses the one which maximizes the value gain per unit of aggregate resource, i.e., $(\Delta v_{ij})/(\Delta a_{ij})$.

5.2 Mapping of Reservation Based Admission Control of MSS as MMKP

Let there be n session requests for admission control in an epoch. Each session request has a *ready time* (r_i), *deadline* (d_i), *utility* (u_i), *duration* (l_i) and required *resources* (b_i). The session may be started at any time in between *ready time* and *ready time+deadline*.

There might be multiple instances of the session with different starting time and end time but the same bandwidth requirement and utility. The following Figure 5.2 describes the different instances of a session in a group. Let s_{ij} be the jth instance of the ith session which starts at $r_i + j\Delta t$ and ends at $r_i + j\Delta t + l_i$ $(0 \leq j \leq \frac{d_i}{\Delta t})$.

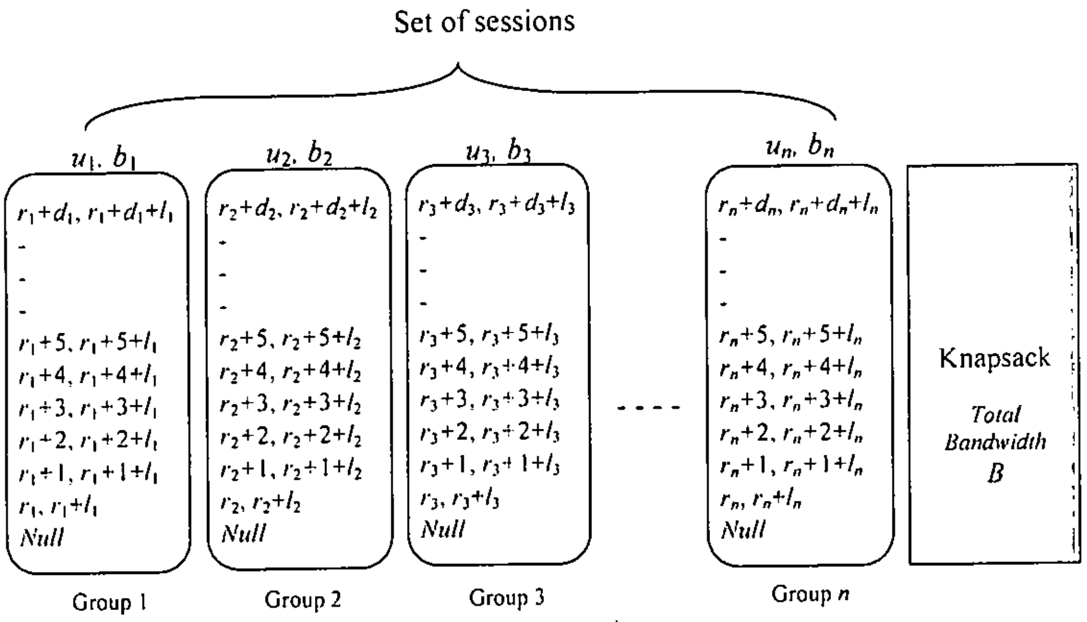


Figure 5-2 : Mapping of Reservation Scheme to MMKP

Reservation based admission control of MSS can be mapped as a *Multidimensional Multiple-choice Knapsack Problem* (MMKP). The mappings are as follows-

- The group of instances can be viewed as a group of items.
- The resource constraint of MSS is equivalent to the constraint of MMKP.
- The total earned revenue of MSS is mapped to the objective function of MMKP.
- The utility u_i and required bandwidth b_i of instances can be mapped to the value provided by the items and weight co-efficient of that item.
- Selection of an instance of session can be viewed as picking (exactly) one item from each stack.
- The default picked value of each group is NULL. If the selection of instance from each group is not possible then the NULL will be remain as the picked item.

As our problem is mapped to MMKP; we should apply the existing heuristic of MMKP in order to solve this reservation based admission control problem. But the existing heuristic algorithm of MMKP will not be applicable to our problem. Because there are some differences between the traditional MMKP and our reservation problem.

- The utility u_i of an instance can not be used as picking criteria of an item from the group because the utility u_i remain constant for a group in our reservation problem.
- The resource constraints of instances are defined by the duration of start time and end time. Hence the total resources consumed by admitting an instance are not constant over the duration of that instance. The resource consumption is different for different segments of an instance. Thus there is unlimited number of resources constraint in our reservation problem.

So, we have to find a new heuristic solution for our reservation based admission control problem.

5.3 Reserve-MMKP: Admission Control of MSS by Solving MMKP

The objective of the MMKP is to pick at most one item from each group for maximizing the total utility, subject to resource constraint R of the knapsack. Picking of items or selection of s_{ij} depends on the "*Selection function*" which indicates selection of those session that gives the highest utility per unit time duration and bandwidth. The selection function of the j^{th} instance of the i^{th} session is defined as-

$$Selection\ Function\ (i,j) = \frac{Utility(i)}{Duration(i) \times Bandwidth(i) \times Average\_Utilized\_Resource(i,j)}$$

$$\Rightarrow SF_{ij} = \frac{u_i}{l_i \times b_i \times AUR_{ij}}$$

The *Average Utilized Resource(AUR)* used in the denominator of the selection function indicates the selection of the instance which has the higher available bandwidth.

The *Average Utilized Resource(i,j)* or AUR_{ij} is defined as follows-

$$Average\ Utilized\ Resource(i,j) = AUR_{ij} = \frac{\sum_{k=r_i+j\Delta t}^{r_i+j\Delta t+l_i} UB_k}{l_i}$$

Utilized_Bandwidth is the total resource consumed between the starting and ending of an instance of a session.

In the Reserve-MMKP, the admission control process is done in two phases. In the first phase, the initial solutions are found i.e. the accepted sessions and preliminary rejected sessions are classified. In the second phase, the final solutions are considered i.e the preliminary rejected sessions are tried to be accommodated so that the utility is maximized.

Phase-1: Finding the accepted sessions and preliminary rejected sessions

Step-1: The value of selection function is calculated for all the instances of each session request which satisfy the resource constraint.

Step-2: The instance that has the highest value of selection function is picked.

Step-3: The picked instance is added to the accepted session and its group is removed from session request list as it is admitted.

Step-4: Step-1 to Step-3 is repeated until no instance can be picked any more.

Step-5: The sessions whose no instances are picked, will be considered as preliminary rejected sessions.

These preliminary rejected sessions will be tried to accommodate in Phase-2.

Phase-2: Admitting preliminary rejected sessions

Step-1: The preliminary rejected sessions are sorted according to the descending order of *utility*. This is done in order to give priority to higher utility sessions so that the objective of maximizing the revenue is achieved.

Step-2: Preliminary rejected session from the sorted list is taken one by one for admission according to their order of utility.

Step-3: The "*Best instance*" among the instances of a session is determined. The "*Best instance*" is that instance for which the required resource of that session is available for maximum duration. Mathematically *Best instance* of session s_i may be defined by maximizing $\sum_{k=r_i+j\Delta t}^{r_i+j\Delta t+l_i} C_k$.

$$\text{Where, } C_k = \begin{cases} 1, & \text{when } \textit{available resource} \geq b_i \\ 0, & \text{else} \end{cases}$$

Step-4: In order to accommodate the "*Best instance*" of that preliminary rejected session, the instances of all accepted sessions are so chosen that the required resource to accommodate the "*Best instance*" increases for longer duration. The appropriate instance of an accepted session is determined by using **minimum value** of the *"Selection Index"*. The selection index of *j*th instance of the *i*th session for rejected session s_k is defined by -

$SI_{ijk} = [s_{ij} \cap s_k^{Best}] \times UB_{s_{ij} \cap s_k}^{Best}$; where s_k^{Best} is best "*Best instance*" of the rejected session *k*.

Step-5: If the rejected session can be accommodated then it is admitted otherwise all the shifting done so far are roll backed to previous state.

Step-6: Step 3 to 5 is repeated until remaining preliminary rejected sessions are tried to accommodate.

Step-7: The acceptance and rejection decision is finalized.

5.4 Complexity Analysis of Reserve-MMKP

Let us consider an epoch where admission controlling of *n* sessions is done. For simplicity of calculation the deadline of all the sessions are considered to be the same. Thus the number of instances in each session would be $d = \frac{d_i}{\Delta t}$. So, the total number of instances in the MSS would be $n \times d$. Again, for the simplicity of the analysis we consider $l_i = L$, i.e., the length of each session is equal.

Complexity of *selection function* of each instance depends on the *duration* of the session and cost of calculation of *AUR*(*i,j*). The cost of calculation of *AUR*(*i,j*) depends on the average height of the segment tree. If the total number of accepted session is *p* then the total number of nodes in the segment tree is $4p+1$ and the height of the segment tree is $\log_2(4p+1)$. Hence, in the worst case, the complexity of the selection function will be $L \times \log_2(4p+1) \approx O(L \log_2 p)$

Cost analysis of first phase:

In the ith iteration of the first phase we need to evaluate the items of $(n - i + 1)$ groups. So the number of operations required in the worst case to find the selection function in the ith iteration is $(n - i + 1)dL\log_2(4p+1)$. The total number of operations in this phase is:

$$\sum_{i=1}^{n}(n-i+1)dL\log_2(4p+1)=\frac{n(n-1)}{2}dL\log_2(4p+1) \text{ i.e., } O(n^2 dL\log_2 p)$$

Cost analysis of Second phase:

Consider q rejected sessions from Phase 1 is being accommodated in Phase 2. For each rejected session we do the following operations:

- $dL\log_2(4p+1)$ operations for finding the best instance
- $pdL\log_2(4p+1)$ operations to find the selection index of all the items previously accepted sessions.

Thus the cost of Second phase for q rejected sessions is expressed by $(qdL + pqdL)\log_2(4p+1)$. Considering $p = q = n/2$, the complexity of the reservation technique can be represented by $O(n^2 dL\log_2 p)$

5.5 Summary

In this chapter we briefly described the classical 0-1 Knapsack problems and the variants of Knapsack problems. We mapped our problem into MMKP. We proposed a solution by applying the heuristic of Knapsack problem. We also analyze the computational complexity of the proposed algorithms of Reserve-MMKP. In the next chapter we will describe the simulation of the proposed system and experimental results.

Chapter 6

Simulations and Results

In this chapter we will present the simulation technique of MSS by describing the time advancement mechanism, random number generation, arrival mechanism of the sessions. The simulation has been conducted by initializing simulation parameters that matches a practical scenario of a Multimedia Server System. Here we considered a scenario of a Video on Demand system as an example of MSS. Before describing the whole process we need to describe time advanced mechanism for controlling simulation clock and random number generation technique for generating session requests for Video on demand (VoD) services. We also present the experimental results plotted in the graph followed by a discussion on the obtained results.

6.1 Simulation Environment

In order to evaluate the performance of proposed reservation based admission control algorithm we implemented it using C++ in a single processor Pentium 4 (2.0 GHZ, 256 MB RAM) machine running Windows XP. We have to generate different sets of data for our experiment. These are described in the following sections.

6.2 Time Advance Mechanism

In order to simulate our proposed solution, we have to keep record of the current time and future time. Hence we need a mechanism to advance the time of simulation clock from one value to another. Historically, two principal approaches have been suggested for advancing the simulation clock i.e, fixed increment time advance, next event time advance [35].

6.2.1 Fixed Increment Time Advance

In this approach, the simulation clock is advanced in increments of exactly Δt time units for some appropriate choice of Δt. After each update of the clock, we determine where any event should have occurred during the previous interval of length Δt. If one or more events were scheduled to have occurred during this interval, these events are considered to occur at the end of the interval and the system states are updated accordingly. Fixed increment time advance is illustrated in the following Figure, where the curved arrows represent the advancing of simulation clock.

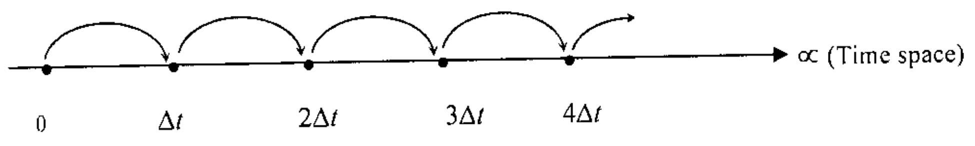


Figure 6-1 : Fixed increment time advance

In our Reserve-HEU, processing of admission control is done in a regular time interval which we called an epoch (described in the Section 3.4.1). Thus we used fixed increment time advance mechanism in our reservation based admission control simulation.

6.2.2 Next Event Time Advance

In this approach, the simulation clock is initialized to zero and the times of occurrence of future events are determined. The simulation clock is then advanced to the time of occurrence of the most imminent (first) of these future events, the state of the system is updated, and the time of occurrence of future events are determined, etc. The details of next event time advance approach is illustrated in the Figure 5-2 where t_i is the time of arrival of session, e_i is the time of occurrence, and A_i is the interarrival time between two consecutive arrivals.

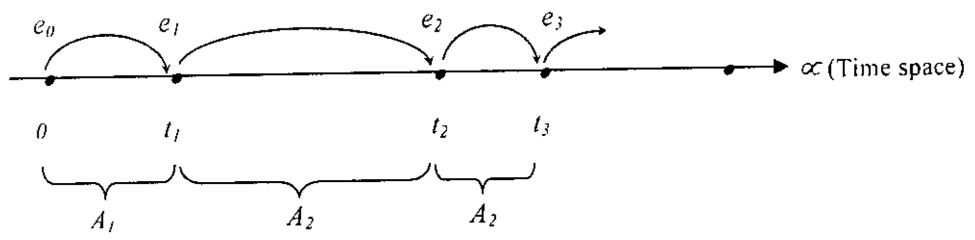


Figure 6-2 : Next event time advance

In case of online admission control simulation, where the processing of admission control is done immediately after the arrival of sessions, next time advance mechanism may be used.

6.3 Random Number Generation

Random numbers are used in simulations in order to generate random samples. There are many techniques for generating random numbers. In our simulation we used two types of random numbers i.e., (i) random numbers generated from uniform distribution and (ii) from exponential distribution.

6.3.1 Uniform Random Number

Probability distributions are the tools used to estimate the likelihood that random values will occur within certain ranges. Probability distributions are defined by the probability density function. A uniform random variable over the range $[a, b]$ has the *probability density function* -

$$f(x) = \begin{cases} \frac{1}{b-a} & ; a \le x \le b \\ 0 & ; \text{elsewhere} \end{cases}$$

In our simulation we use the following steps to generate a uniform random number:

1. Generate uniform random variate, $U(0, 1)$.
2. Return $X = a + (b - a)U$.

If many X values are to be generated, the constant $(b - a)$ should be, of course, be computed beforehand and stored for use in the algorithm. In our simulation, uniform random number generator is used to generate *duration* l_i, *bandwidth* b_i and *utility* u_i.

Duration: Generally the duration of a movie is 3 hours but it may vary from 1.5 hours to 4.5 hours. We generate a uniform random variate in the range of $[1.5, 4.5]$ as

$l_i = 1.5 + 3 \times U(0, 1)$

Bandwidth: The average bandwidth requirement for MPEG1 movie stream is 1.5 Mbits/s and for MPEG2 movie stream is 4 Mbits/s [8]. Hence we consider it as a uniform random number and it is generated as

$b_i = 2 + 2 \times U(0, 1)$

Utility(Revenue): Generally the amount paid for a particular movie depends on the duration of the movie and the quality of the movie(the bandwidth requirement). To simulate other unknown factors we have used the uniform random number as

$u_i = b_i \times l_i \times 5 + 5 + 10 \times U(0, 1)$

Here we consider 5 unit utility per unit bandwidth per unit duration.

6.3.2 Exponential Random Number

The probability density function of *exponential distribution* of random number with mean β (any positive real number) is $f(x) = \frac{1}{\beta} e^{-x/\beta}$ for $x \geq 0$. In order to generate random number of exponential distribution we have to do the following-

1. Generate $U(0, 1)$
2. Return $X = -\beta \ln U$

Ready time: In our simulation we consider that a customer places a request d time unit earlier than the ready time. Thus d indicates variable showing the difference between ready time and requesting time. Generally this difference is almost constant in most of the cases. That is why an exponential random variable of 3 hour (average difference is 3 hour) is used to find the ready time as follows-

$r_i = \textit{Current time} - 60 \times 180 \times \ln U(0, 1)$

6.4 Arrival of Session Requests

As we described earlier (Section 3.4.1) the multimedia session requests are submitted on batches in a regular time interval which we called an epoch. It is assumed that the inter arrival time of the multimedia session in MSS follow exponential arrival pattern as the call arrival pattern in telecommunication industry follows this behavior. Here we assume that the users of the system do not have any special attention to any time to put requests for watching videos. Although this is not correct always, we assume this to simplify the simulation. In the simulation we advance the simulation clock in each epoch by fixed increment time advance mechanism.

Following are the steps of initializing simulated system-

1. Initialize the simulation clock (i.e., *current time*)
2. Generate the arrival time t_1. (i.e., *current time+ epoch*)
3. Generate exponential random variates with a mean interarrival time to find the sessions which arrives between *current time* and *current time + epoch.*

4. The exponential random variates will be treated as *ready time* r_i of the sessions. *Ready time* is dependent on current time but independent on any previous ready times. It is very much common for all kinds of reservation scheme that there is an average difference between reservation time and session starting time. Most of the users will place reservation request d_{avg} time before the ready time. Thus an exponential random variate with this average is used to determine the ready time.
5. Generate the uniform random variate that is independent on current time and previous random numbers to simulate *duration* of the session. Similarly *utility* and *bandwidth* will be generated using uniform random number.
6. The arrival method returns the following session request s_i in the ith epoch for admission control.

   s_i = (*Ready time, duration, bandwidth, utility*)

The above mentioned arrival method of sessions are used in the following two simulations for admission control. They are-

- Reserve-HEU uses the algorithms which are described in Section 4.5
- Reserve-MMKP uses the algorithms which are described in Section 5.3.

We compare the performance of proposed reservation algorithms with the online version of admission controller. In online admission control session requests are admitted if the resources are available based on utility. Neither future reservation nor shifting of ready time is done here.

In case of the simulation of online admission control, the sessions entered in the reservation based system for all epoch are collected and sorted according to the *ready time* and *utility*. Then the simulation clock is advanced by next event time advancement mechanism so that the sessions are considered for admission depending upon the availability of resource and utility.

6.5 Flow of Control

The simulation begins in the empty and idle state; i.e., no user is present and the server is idle. At time t = 0, the main program will invoke the initialization routine, where the simulation clock will be set to zero, the system state (i.e. bandwidth) and counters are also initialized. Session requests are received and the admission controller determines which sessions will be admitted and which sessions will be rejected subject to maximization of the server revenue. Reservation based admission control algorithms have already been described in Section 5.6 and 5.7. These steps will be repeated for all the epochs. The following diagram illustrates the control flow of the simulation.

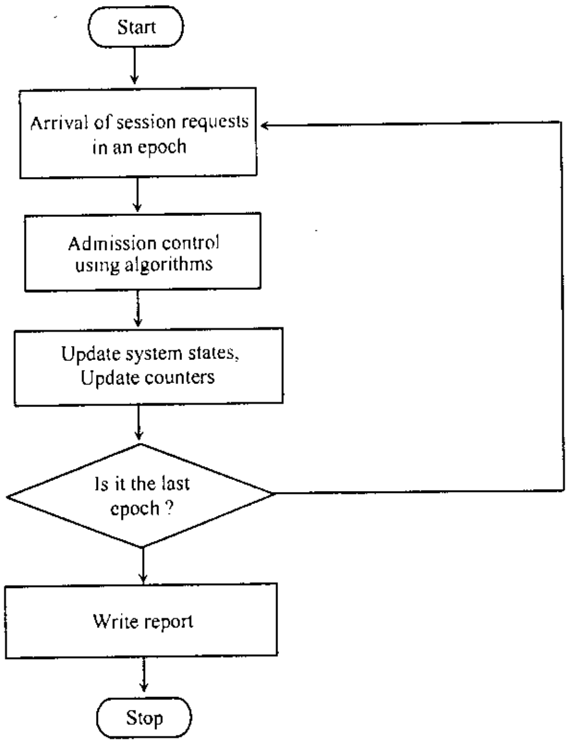

Start
Arrival of session requests in an epoch
Admission control using algorithms
Update system states, Update counters
Is it the last epoch ?
Write report
Stop

Figure 6-3 : Flow control of Simulation.

6.6 Discussion of Simulation Results

Following simulation results have been gathered to demonstrate the performance of reservation based admission controller.

- Average sessions per epoch for different system sizes,
- Earned revenue at different simulation time of the system,
- Earned revenue using different reservation schemes for different system sizes, arrival rate, epoch size and deadline.
- The average time required to do admission control of each epoch using different algorithms.

Simulation results are explained individually in the following sub sections.

6.6.1 Average Sessions per Epoch for Different System Sizes

This experiment is done in order to check the validity of the simulation. If the system size (i.e. the bandwidth) is increased then the system must be able to accommodate more sessions in each epoch. This fact holds in the three simulations (i.e., Reserve-HEU, Reserve-MMKP and Online). The following graph of Figure-6.4 shows that the admitted sessions per epoch are almost proportional to the system size.

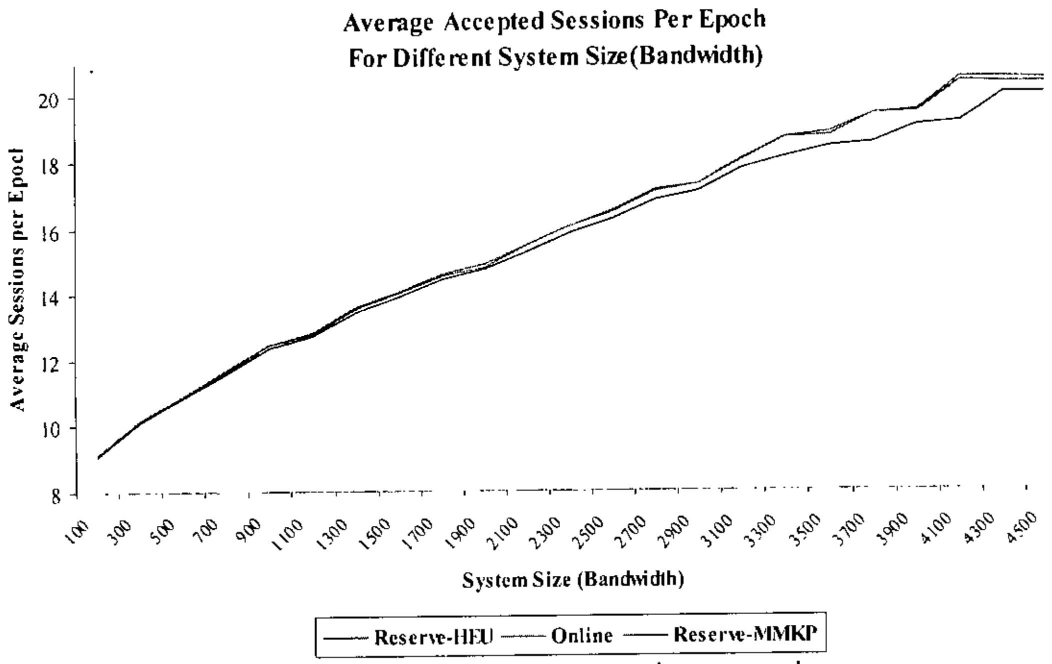


Figure 6-4 : System size vs. average sessions per epoch

6.6.2 Current time vs. Current revenue

This experiment is also done in order to check the validity of the simulation. This simulation result shows the revenue earning at different situation of the system starting from beginning to saturation. Figure 6-5 shows earned revenue at a particular current time for 2Gbit/s and 1Gbit/s bandwidth. At the very beginning there is no session in the system and it remains empty. The new sessions could easily get admitted in this situation. The beginning epochs earns more and more as there is more incoming sessions than leaving sessions. At some point (here current time ≈350) the system becomes full with reserved sessions. Only new sessions could get in if somebody leaves and thereby the earned revenue in each epoch remains almost constant. This situation of the system is called saturation.

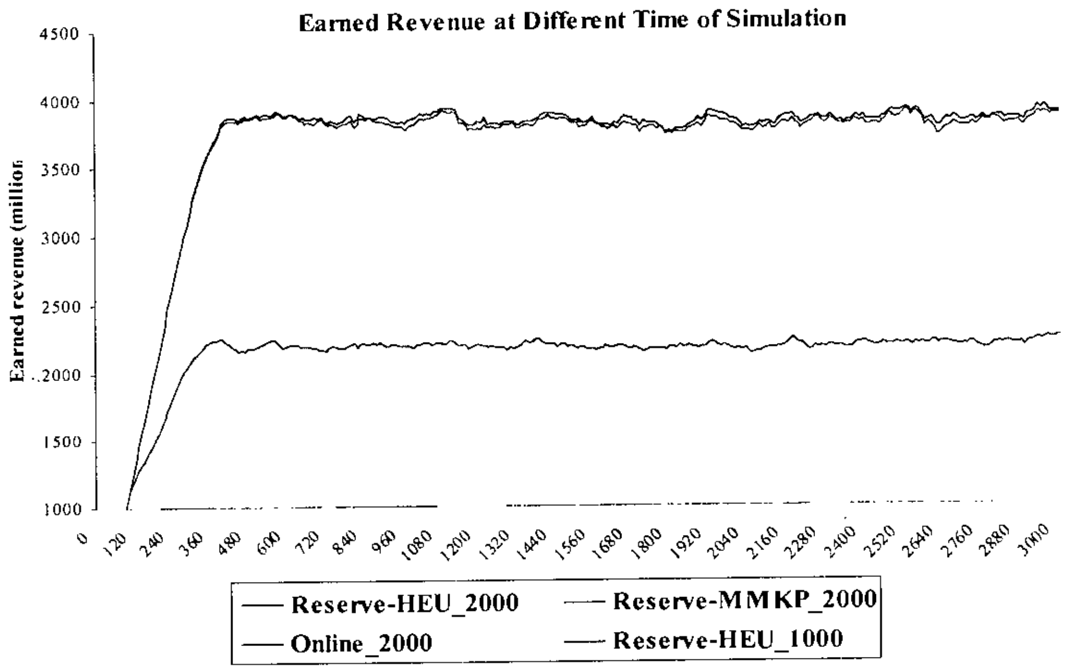


Figure 6-5: Current time vs. Current revenue

6.6.3 Effect of Revenue on System Size (Bandwidth)

The number of users in the system depends on the total available bandwidth in the system. Figure 6-6 shows the effect of total bandwidth in earning revenue. The earned revenue of the system against bandwidth is plotted in this graph. Figure 6-6 shows three curves for online admission controller, Reserve-HEU and Reserve-MMKP.

Careful observation shows that online admission controller earns better revenue than the other reservation schemes. The reason behind it is that online admission controller does not need to admit the nonprofit requests. In Reserve-HEU and Reserve-MMKP, early non profitable requests get admitted as they place the request long before starting off. In this simulation the rate of the service is considered same in both the cases. But practically it is not a regular practice. Users will pay more for reservation because users will be certain that they will be able to enjoy the session at their desired time. The Quality of Service (QoS) will be guaranteed as well. Hence the reservation scheme will be much more profitable for service providers. Thus loss of revenue observed in Fig 6-6 can easily be compensated by taking extra reservation fees. Among the reservation schemes, Reserve-MMKP earns better revenue than the Reserve-HEU as it considers all possible instances with higher complexity.

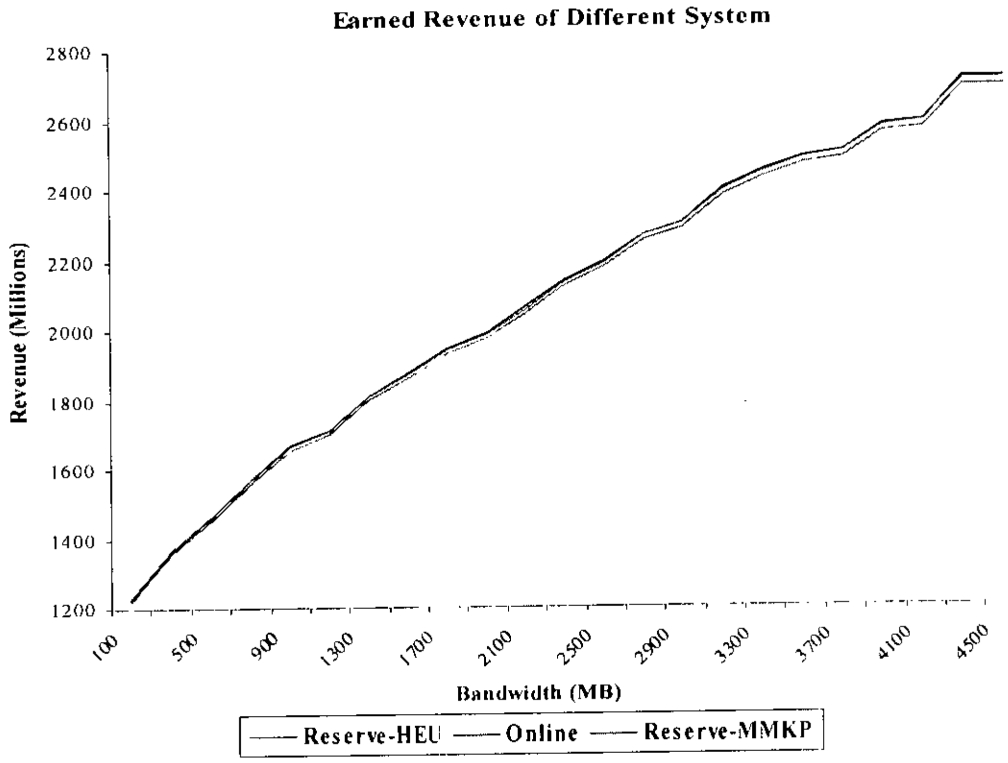


Figure 6-6: Effect of Revenue on System size (bandwidth)

In order to compare the performance of revenue earning capacity, we plotted the differences of revenue that is earned by Reserve-HEU and Reserve-MMKP from the earned revenue of Online admission controller for different system size in Figure 6-7.

From the Figure 6-7 it may be observed that the differences between online admission controller and Reserve-MMKP are less than that of Reserve-HEU. It also observed from the mentioned graph that sometime the earned revenue in Reserve-MMKP may be higher than that of the online admission controller (Here it is from 100 to 700 Mb).

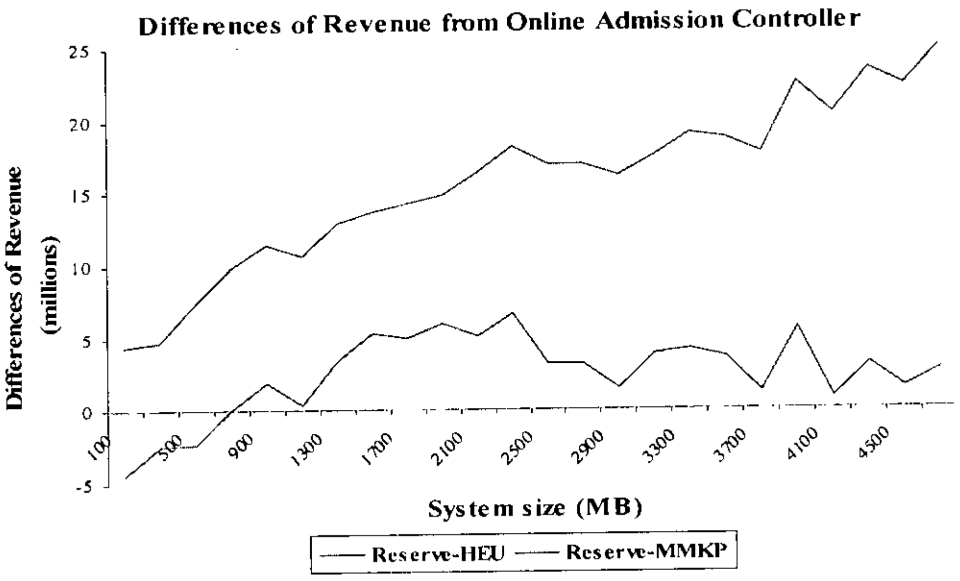


Figure 6-7: Differences of revenue from online system

6.6.4 Effect of Admission Controlling Time on Average Sessions per Epoch

In our simulation, it is found that in the primary stage when resource is available, it does not require longer time to do admission control. But when average numbers of sessions in an epoch are increased then the number of preliminary rejected sessions will also be increased. As there are more preliminary rejected sessions, the admission controller will try to accommodate these preliminary rejected sessions. Hence more time is needed for admission controlling in each epoch. The effect of admission controlling time on average sessions per epoch using Reserve-HEU and Reserve-MMKP are shown in the Figure 6-8 and Figure 6-9 respectively.

Figure 6-10 compares the results of Reserve-HEU and Reserve-MMKP in the same graph. The quadratic increase of the time required in Figure 6-8 and Figure 6-9 justifies the worst case time complexity of Reserve-HEU and Reserve-MMKP. We find the time required by Reserve-HEU is negligible compared to the time required by Reserve-MMKP. This is because of the term dl in the complexity of Reserve-MMKP.

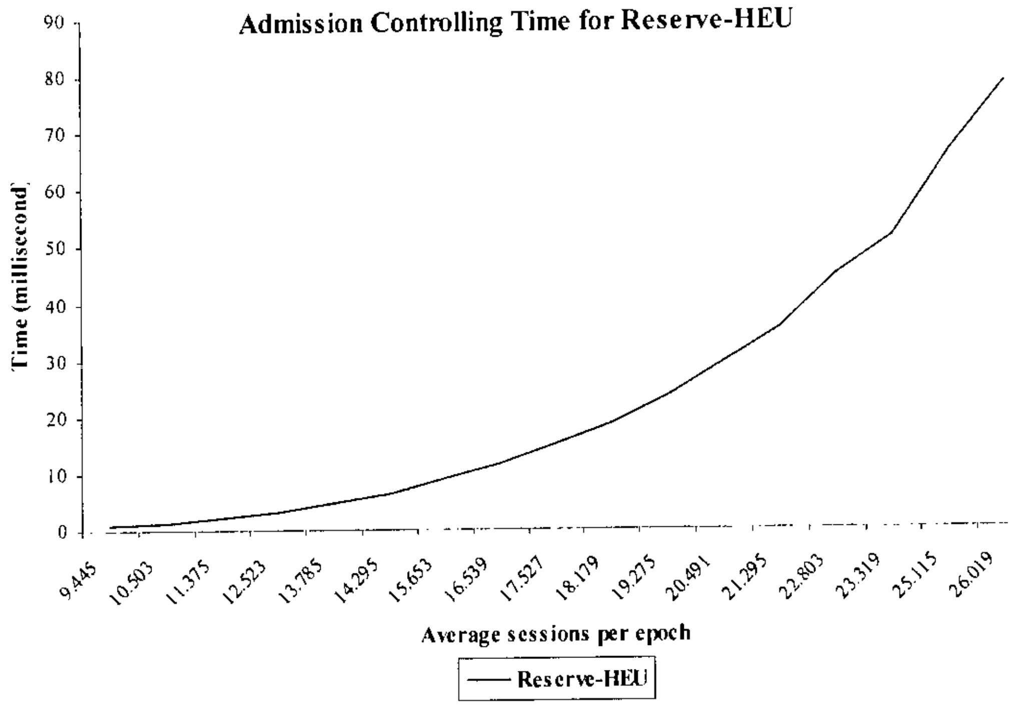


Figure 6-8: Effect of admission controlling time on average sessions per epoch for Reserve-HEU.

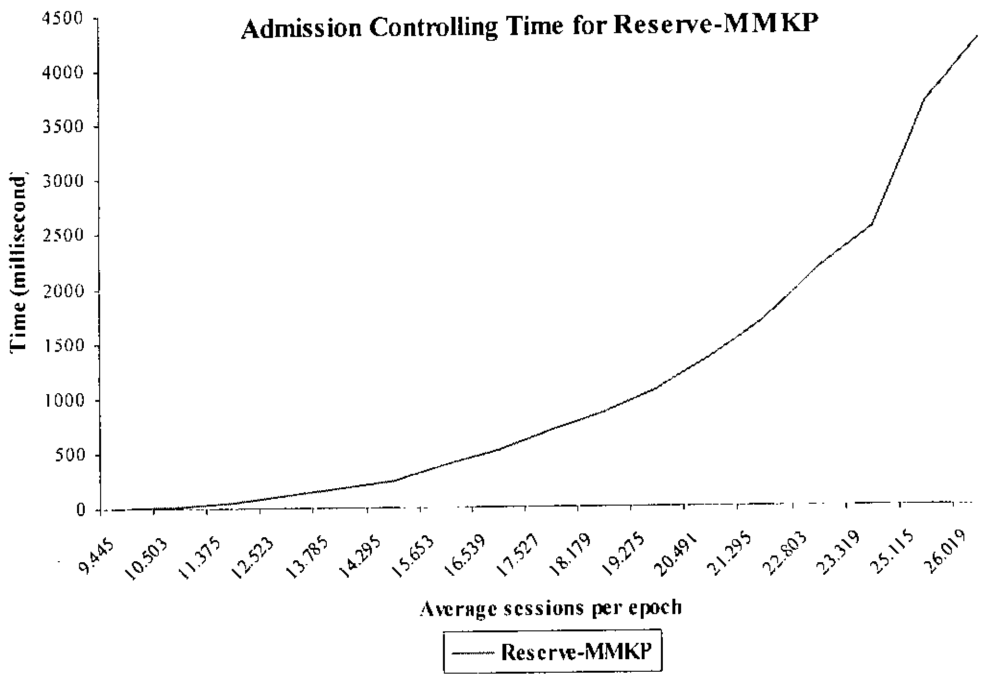


Figure 6-9: Effect of admission controlling time on average sessions per epoch for Reserve-MMKP.

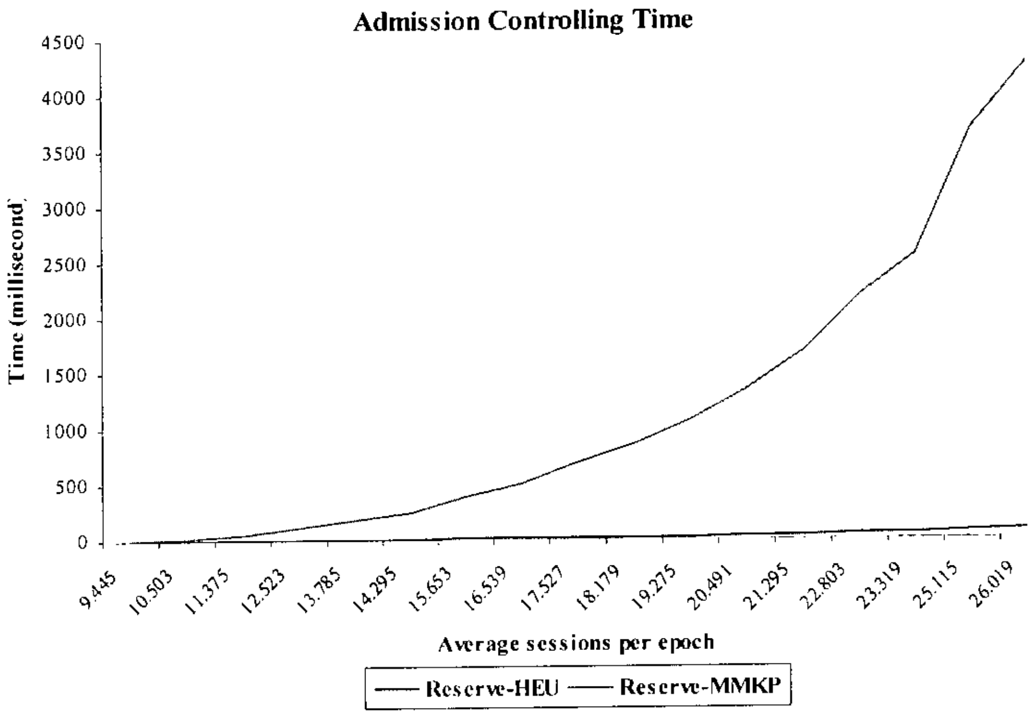


Figure 6-10: Effect of admission controlling time on average sessions per epoch.

6.6.5 Effect of Arrival Rate on Revenue

Arrival rate is the inverse of interarrival time. If arrival rate increases then the revenue will also be increased. From the curve of the Figure 6-11, it is seen that revenue is almost proportional to arrival rate. But after a certain arrival rate the revenue is found to be constant. This is because after a certain arrival rate the distribution of the system will become saturated.

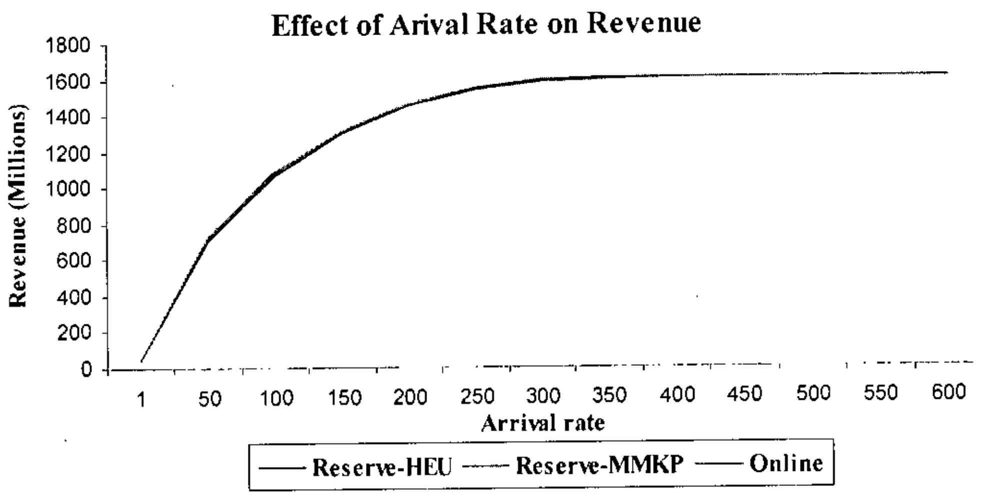


Figure 6-11: Effect of arrival rate on revenue.

6.6.6 Effect of Epoch Size on Revenue

Revenues for different epoch sizes are plotted in the Figure 6-12. Careful observation shows that if the epoch size is larger, then more session requests can be considered, as a result better optimality in admission controlling is found. Thus there will be some more revenue for larger epoch size.

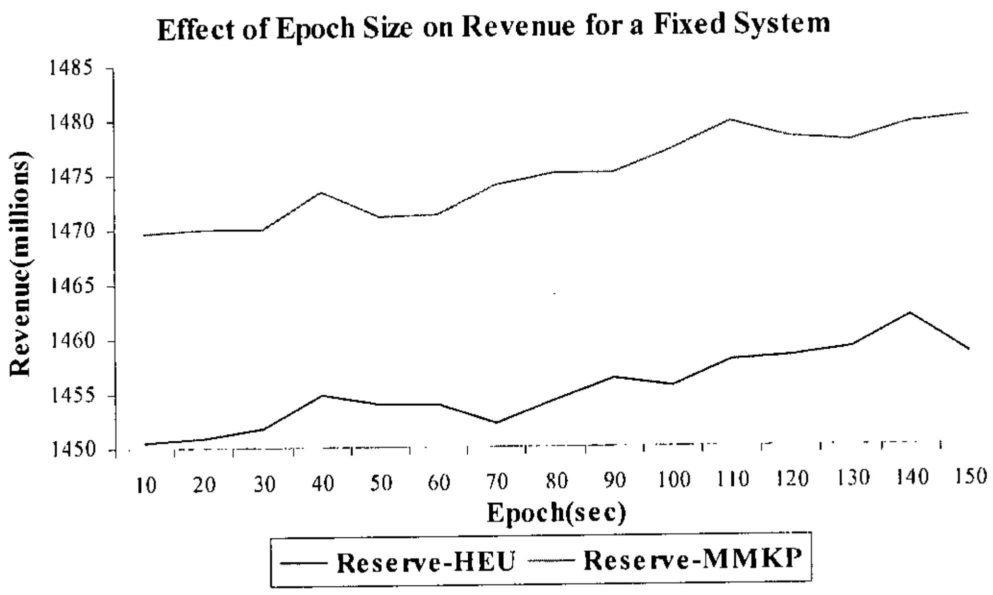


Figure 6-12: Effect of epoch in a variable system size.

The same behavior is observed for different values of bandwidth (system size) in Figure 6-13.

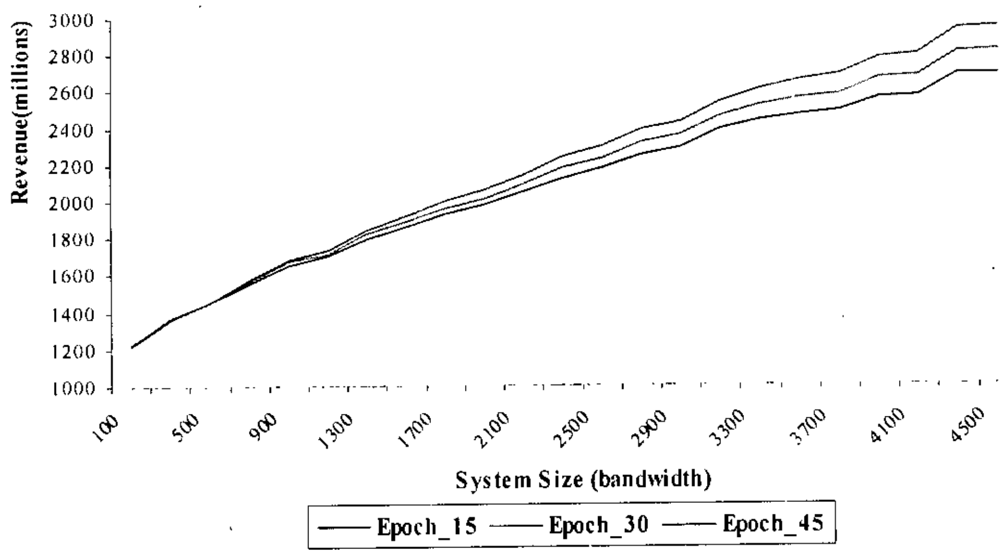


Figure 6-13 : Effect of epoch in a increased system size

6.6.7 Effect of Deadline on Revenue

Earned revenues by Reserve-HEU and Reserve-MMKP for different deadlines are plotted in the Figure 6-14. Careful observation shows that if the deadline of the session is increased then more sessions can be accommodated by the admission controller and hence earned revenue will be increased. This is because the admission controller has wide choice of adjusting the starting time. But there is a disadvantage of increasing the deadline, if the deadline is increased than the admission controller needs more time for calculations. Figure 6-14 shows that Reserve-MMKP earns better revenue than Reserve-HEU.

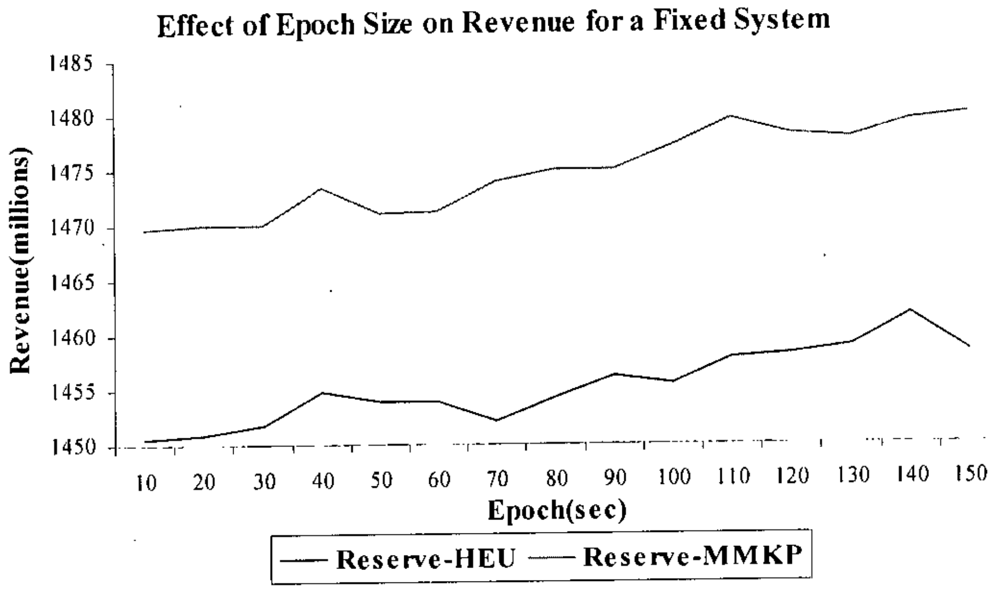


Figure 6-14: Effect of deadline on revenue.

6.7 Summary

In this chapter we described the simulations of the admission controller for reservation based MSS particularly for the Video on Demand system users. Simulation results are plotted and explained comparing the other admission control algorithms. The result shows that reservation is an elegant way of admission controlling without hampering the earned revenue. It might be a good approach of earning more revenue by charging a little bit higher price to the users for the guaranteed satisfaction of enjoying Video on Demand in a predefined duration. The next chapter represents the conclusion of the thesis.

Chapter 7

Conclusion

This chapter summarizes the major contributions of this thesis and presents suggestions for future research work.

7.1 Contributions

Formulation of a new problem: We formulate a new problem for reservation based admission control of a Multimedia Server System. The main objective of the research work is to manage a reservation based admission control mechanism of a Multimedia Server System to maximize the earned revenue. This would be a profitable service that can be implemented by using existing network and hardware, and customers will be allowed to book their session before the starting time of the session.

Class of the problem: We analyzed the class of the problem. The NP Completeness of this problem has been justified using proof by restriction method.

Heuristic solution of the problem: We present two heuristic solutions of the problem. The first heuristic finds the solution by accommodating profitable sessions into the system. These accommodations are done by adjusting or shifting the starting times of the sessions.

Mapping the problem to the MMKP: The reservation based admission control has been mapped to the classical Multiple Choice Multi Dimensional Knapsack Problem. A heuristic Reserve-MMKP has been proposed to solve this special case of the MMKP.

Complexity analysis of the problem: Complexity of the proposed admission control techniques have been presented in terms number of comparisons.

Simulation of the MSS: A simulation of the MSS has been done by using the C programming language to determine the performance of the proposed admission control heuristics.

Analysis of the results from the simulation: The results from the simulation are presented in graphs and the behaviors of the curves are analyzed with respect to the complexity of the algorithm and general intuition.

7.2 Future Research Works

This work has opened many problems and issues, which require further research. Some of them are listed as follows:

- ***Distributed multimedia servers system***: Our research is based on a single Multimedia Server System. The research may be extended to *Distributed Multimedia Server Systems*.
- ***Multiple QoS***: We considered only one QoS level, which will be guaranteed for the users. It would be interesting if multiple QoS level could be considered for the customer.
- ***Shifting the starting time of accepted sessions***: In our research we did not shift the starting time of those sessions which are already accepted in previous epochs. Thus the research may be extended if shifting the starting time of accepted sessions are allowed.
- ***Predicting the future session request***: We did not predict the future session calls. So, predicting the future calls might be an important research idea.
- ***Making a practical system of reservation scheme:*** We did not make any practical system based on our research. Future research should be done on the implementation of the proposed reservation scheme.
- ***Considering other resources***: We considered only I/O bandwidth as resource in our research. So future research may address the other resources like network bandwidth, CPU cycles, memory etc.
- ***Application of reservation scheme***: The outcomes of our research may be applied for Service Level Agreement (SLA) in order to control an Enterprise Network.
- ***Consideration of reservation fees***: We do not consider any fees for reservation. Simulation and observation of a system with differential reservation fees (low reservation fees for earlier reservation and high reservation for instantaneous entry) could be done to find the appropriate business models for the service providers.

References

[1] D. Sitaram and A. Dan. Multimedia Server: Applications, Environment, and Design. Morgan Kaufmann publishers, First edition, 2000.

[2] S. Khan and K. F. Li and E. G. Manning. The Utility Model for Adaptive Multimedia Systems. International Conference on Multimedia Modeling, Singapore, pp 111-126, Nov 17-20, 1997.

[3] S. Khan and K. F. Li and E. G. Manning. Padma: An Architecture for Adaptive Multimedia Systems. IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, pp 105-108, Aug 20-22, 1997.

[4] M. M. Akbar. The Distributed Utility Model Applied to Optimal Admission Control and QoS Adaptation in Multimedia Systems and Enterprise Networks, PhD Dissertation. Department of Computer Science, University of Victoria, 2002.

[5] S. Khan, K. F. Li, E. G. Manning, M. M. Akbar. Solving the Knapsack Problem for Adaptive Multimedia System, Studia Informatica, 2002.

[6] F. Halsall. Multimedia Communications: Applications, Networks, Protocols and Standards. Pearson Education Asia, 2001.

[7] R. Steninmetz and K. Nahrstedt. Multimedia: Computing, Communications, and Applications. Prentice Hall, Inc., 1995.

[8] J. D. Gibson. Multimedia Communications: Directions & Innovations. Harcourt India Private Limited., First edition, 2001.

[9] K. Kawachiya and H. Tokuda. QoS-Ticket: A New Resource-Management Mechanism for Dynamic QoS Control of Multimedia. Proceedings of Multimedia Japan 1996, pp 14-21, April 1996.

[10] S. V. Raghavan and S. K. Tripathi. Networked Multimedia Systems : Concepts, Architecture & Design. Prentice Hall Inc. New Jersey, pp 34-35, First edition, 1998.

[11] W. Stallings. High-Speed Networks and Internets: Performance and Quality of Service. Pearson Education (Singapore) Pte. Ltd., Indian Reprint, 2004.

[12] J. W. S. Liu, K. J. Lin, W. K. Shin, and A. C. Yu. Algorithms for Scheduling Imprecise Computations. In IEEE Real Time Systems Symposium, pp 252-260, San Jose, CA, 1987.

[13] C. L. Liu and J. W. Layland. Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment. IEEE Journal of the ACM, 20(1): pp 46-61, January 1973.

[14] J. Y. Chung and J. W. S. Liu. Performance of Algorithms for Scheduling Periodic Jobs to Avoid Timing Faults. In Proceedings of 22nd Hawaii International Conference on System Sciences, pp 683-692, Hawaii, 1989.

[15] D. W Craig and C. M. Woodside. The Rejection Rate for Tasks with Random Arrivals, Deadlines and Preemptive Scheduling. IEEE Transactions on Software Engineering, 16(10), pp 1198-1208, October, 1990.

[16] J. P. Lehoczky and L. Sha. Performance of Real Time Bus Scheduling Algorithms. ACM Performance Evaluation Review, 14(1), pp 44-53, May, 1986.

[17] J. W. S. Liu. Real Time System. Pearson Education Asia, Indian Reprint, 2001.

[18] W. Stallings. Computer Organization & Architecture – Designing for Performance. Prentice-Hall of India, 6th edition, 2003.

[19] P. Chen, E. Lee, G. Gibson, R. Kartz, and D. Patterson. "RAID: High Performance, Reliable Secondary Storage." In IEEE ACM Computing Surveys, June,1994.

[20] H. Domjan and T. R. Gross. Managing Resource Reservations and Admission Control for adaptive Applications, Proceedings of IEEE International Conference, Zurich, 2001.

[21] H. M. Vin, A. Goyal, and P. Goyal. An Observation-Based Admission Control Algorithms for Multimedia Servers. Proceedings of the International Conference on Multimedia Computing and Systems, pp 234 – 243, May 15-19, 1994.

[22] I. R. Chen and C. M. Chen. Threshold-Based Admission Control Policies for Multimedia Servers. The Computer Journal, Vol. 39, No. 9, pp 757-766, February, 1997.

[23] I. R. Chen, and S. T. Li. A cost-based admission control algorithm for handling mixed workloads in multimedia server systems. Proceedings of the Eighth International Conference on Parallel and Distributed Systems, ICPADS 2001. pp 543–548, June 26-29, 2001.

[24] N. Shacham and H. Yokota. Admission control algorithms for multicast sessions with multiple streams. Selected Areas in Communications, IEEE Journal, Volume: 15, Issue: 3, pp 557 – 566, April 1997.

[25] F. Y. S. Lin. Optimal real-time admission control algorithms for the video-on-demand (VOD) service. IEEE Transactions on Broadcasting, Volume: 44, Issue: 4, pp 402 – 408, Dec. 1998.

[26] A. Jamin, P. B. Danzig, S. J. Shenker, L. Zhang. A Measurement-based Admission Control Algorithm for Integrated Service Packet Networks. IEE/ACM Transactions on Networking: pp. 56-70, Vol. 5, No. 1, 1997.

[27] A. W. Moore. An implementation-based comparison of Measurement-based Admission Control algorithms. Journal of High Speed Network: pp 87-101, Volume 13, Number 2, 2004.

[28] J. Guo and L. Bhuyan, R. Kumar and S. Basu. QoS Aware Job Scheduling in a Cluster-Based Web Server for Multimedia Application, Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05), Denver, April 3-8, 2005.

[29] S. Elnikety, E. Nahum, J. Tracey, W. Zwaenepoel. A Method for Transparent Admission Control and Request Scheduling in E-Commerce Web Sites, Proceedings of the 13th World Wide Web Conference in New York City, May 17-22, 2004.

[30] A. S. Tanenbaum. Computer Networks. Prentice-Hall Of India Private Limited, 4th Edition, 2003.

[31] H. Frazier and H. Johnson. Gigabit Ethernet: From 100 to 1000 Mbps. IEEE Internet Computing., January/February, 1999.

[32] M. R. Garey and D. S. Johnson. Computer and Interactability : A Guide to the Theory of NP-Completeness. W. H. Freeman and Company, Sunfranciscod, 1st Edition, 1979.

[33] J. B. Bentley. Segment Tree. In http:// www.iis.sinica.edu.tw /~dtlee/ CSIE/ segment_tree.html

[34] M. A. Weiss. Data Structures and Algorithm Analysis in C. Preason Education Asia, Sixth Indian Reprint, 2001.

[35] A. M. Law, W. D. Kelton. Simulation Modeling & Analysis. MacGraw-Hill International Series, 3rd Edition, 2000.