A NEW SCHEDULING ALGORITHM FOR MULTIMEDIA COMMUNICATION

THESIS

Presented to the Graduate Council of the
University of North Texas in Partial
Fulfillment of the Requirements

For the Degree of

MASTER OF SCIENCE

By

Venkata Somi Reddy Alapati, B.Tech.
Denton, Texas
May, 1995

A NEW SCHEDULING ALGORITHM FOR MULTIMEDIA COMMUNICATION

THESIS

Presented to the Graduate Council of the
University of North Texas in Partial
Fulfillment of the Requirements

For the Degree of

MASTER OF SCIENCE

By

Venkata Somi Reddy Alapati, B.Tech.
Denton, Texas
May, 1995

C-9

Alapati, Venkata Somi Reddy, A New Scheduling Algorithm for Multimedia Communication. Master of Science (Computer Science), May, 1995, 51 pp., 4 illustrations, bibliography, 31 titles.

The primary purpose of this work is to propose a new scheduling approach of multimedia data streams in real-time communication and also to study and analyze the various existing scheduling approaches. Various techniques involved in scheduling multimedia data streams, like synchronization and jitter control techniques have been studied. An architecture for a general scheduling algorithm is discussed. A scheduling algorithm which supports a separate virtual connection for each information type as well as control data is proposed. The multiple data streams are supported by multiple channels on an ATM network. This algorithm will also deal with the issues of *"continuity"* and *"coordination"*, in the multimedia data streams over the network. The implementation issues of this scheduling algorithm have been identified and studied.

ACKNOWLEDGEMENTS

I am thankful to Dr. C. Q. Yang for the support and guidance he gave me during my Master's thesis. I am very grateful for the opportunity to work and learn under him. I also thank the Department of Computer Science for the financial support. Thanks to Dr. Paul S. Fisher and Dr. Tom Jacob for consenting to be the thesis committee members. My friends and family, though lastly mentioned, are the most important in terms of moral support and inspiration.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1. Multimedia Communication

Real-time Multimedia applications are those that require information to be transmitted in a continuous mode from one location to another [13]. In a single medium data stream, the continuity of the data may be interrupted by gaps and jitters. To support time-dependent data, multimedia applications require network and operating system components that ensure timely delivery of data. In addition, data of many applications such as video-conferencing and computer-supported collaborative work require coordination of multiple data sources inspite of the asynchronous nature of the network. The characteristics of multimedia data vary widely, and impose strong requirements on the underlying communication system. In considering multiple data streams in a distributed multimedia application, we consider the issues of coordination and temporal synchronization over related data streams. For multimedia connections, the large variation in data sizes of differing data types indicates a large variation in utilized bandwidth during the life of a connection. Due to this variation, channel capacity can be periodically exceeded and loss of destination timing can occur. Also, our scheduling algorithm must have the capability to present time-dependent data to the user. Time dependencies of multimedia data can be implied during creation, e.g. sequences of live data images along with sequences of live audio frames. These time dependencies must be supported by the system. A multimedia information system must be able to overcome any system delays caused by storage, computation and

communication latencies. These latencies are usually random in nature. The integration of several media in the same application on the same network and the nature of the application, induce specific requirements such as high throughput, low delays and multicast facilities.

To support presentation of time-dependent data, scheduling disciplines associated with real-time operating systems are necessary. However, the real-time requirements can be relaxed since data delivery can tolerate some occasional lateness, i.e., catastrophic results will not occur when data are not delivered in time.

The work in data communications has been applied to live, periodic sources, such as packet audio and video, and for applications that do not exceed the capacity of the communication channel. Packet delivery variations at the destination are effectively eliminated by buffering and the introduction of a constant time offset resulting in a reshaping of the distribution of arriving packets to reduce delay variance.

A scheduling algorithm which supports a separate virtual connection for each information type as well as control data is considered. This algorithm will also deal with the issues of *"continuity"* and *"coordination"*, in the multimedia data streams over the network. Practical considerations for applying the derived schedule are also considered.

1.2. Contribution of the Thesis

The primary purpose of this work is to propose a new scheduling approach of multimedia data streams in real-time communication and also to study and analyze the various existing scheduling approaches.

The thesis is organized as follows: Chapter 1 provides a brief introduction of real-time communication. Chapter 2 discusses the real-time communication issues along

with the summary of existing scheduling approaches; the need for synchronization of the multiple data channels and some current synchronization techniques; the need for jitter control schemes in real-time communication and the existing jitter control schemes. In Chapter 3, the new algorithm proposed is discussed and the various implementation issues associated are discussed. In Chapter 4, an architecture which provides a framework for comparing various scheduling algorithms is discussed.

CHAPTER 2

RELATED WORK

The definitions of the various terms used in real-time communication are as follows.

2.1. Definitions

Real-time applications:

Applications that require information to be transmitted in a continuous mode from one location to another are called real-time applications. Multimedia conferencing represents a typical real-time application. In general, high connectivity is needed for real-time applications. A guaranteed bandwidth is required to ensure real-time consistency, and to offer the throughput required by the different media.

Non-real-time applications:

Applications that do not require information to be transmitted in a continuous mode from one location to another are called non-real-time applications. Non-real-time applications such as multimedia mail, are less demanding than real-time applications in terms of throughput and delay. Multicast services and synchronization at presentation time have to be offered.

Guaranteed service:

Guaranteed service is the traditional form of real-time service which involves pre-computed worst-case delay bounds. With this kind of service, if the network hardware is functioning, and a client is conforming to its traffic characterization, then the service commitment will always be met.

Predicted service:

Predicted service uses the measured performance of the network in computing delay bounds of real-time traffic.

Jitter:

The variation in the delay in receiving a packet at the receiver is called jitter. This delay is introduced by the network.

Sharing:

Queueing is a fundamental consequence of the statistical sharing that occurs in packet switched networks. The idea of statistical sharing means that there are several sources using the available bandwidth. Any approach to real-time traffic scheduling, should treat the aggregation of traffic as an important issue. The network must be shared in such a way that clients (1) get better service than if there were no sharing (as in a circuit-switched or TDM network) and (2) are protected from the potentially negative effects of sharing (most obviously the disruption of service caused by sharing with a mis-behaving source that overloads the resource).

Isolation:

Isolation is the more fundamental goal for real-time traffic - it provides guaranteed service for well behaved clients and punishes misbehaving sources.

Jitter shifting:

Scheduling algorithms can be thought of as methods for *"jitter shifting"*, in which explicit actions are taken to transfer the jitter among flows in a controlled and characterized way [4]. A wide range of queue scheduling algorithms are developed and they ought to be examined in the perspective of the extent to which they perform *isolation* and the extent to which they provide *sharing.*

Play back applications:

The real-time applications which we consider are called *play-back* applications. In a play-back application, the source takes some signal, packetizes it and then transmits over the network. The network inevitably introduces some variation in the delay of each delivered packet. The receiver depacketizes the data and then attempts to faithfully play back the signal. This is done by buffering the incoming data to remove the network induced jitter and then replaying the signal at some designated *play-back* point. Any data that arrives before its associated *play-back* point can be used to reconstruct the signal and data arriving after the *play-back* point is useless in reconstructing the real-time signal. Not all real-time applications are play-back applications - but a vast majority of them fit into this paradigm.

Based on the play-back point real-time applications can be classified as:

- Rigid applications.
- Adaptive applications.

Some real-time applications will use *"a priori"* delay bound advertised by the network to set the play-back point and will keep the play-back point regardless of the actual delays experienced. These are called *rigid* applications. For other applications, the receiver will measure the network delay experienced by arriving packets and then adaptively move the play-back point to the minimal delay that still produces a sufficiently low loss rate. These applications are called *adaptive*. Adaptive applications have an earlier play-back point than rigid applications, and will suffer less performance degradation due to delay. The idea of adaptive applications is not relevant to circuit switched networks, which do not have jitter due to queueing. However, video frames can be made adaptive by dropping or replaying a frame as necessary, and voice

can adapt imperceptibly by adjusting silent periods. It should be noted that there are typically limits to this adaptability. For instance, once the delay reaches a certain level, it becomes difficult to carry out such interactive conservations.

2.2. Scheduling Algorithms

Scheduling algorithms for real-time data streams are those schemes which support the presentation of time-dependent data. The design of the scheduling approach must account for the playout times for individual data elements. Also, it must account for latencies in each system component used in the delivery of data, from its source to destination. Therefore, specific scheduling is required for storage devices, the CPU, and communication resources. The work in data communication has been applied to live, periodic data sources such as packet audio and video, and for applications that do not exceed the capacity of the channel. These applications typically use a point-to-point configuration, since there is seldom any need to deliver synchronous data streams from multiple independent live sources.

In stored-data applications, time-dependencies must be managed. Unlike live data, which are typically periodic during acquisition and presentation, stored data can require aperiodic playout, and can be retrieved from storage at arbitrary times prior to presentation to the user. The storage for audio and video data types is very large space, and will exist primarily in secondary storage. When data originate from storage, the system has more flexibility in scheduling the times at which data are communicated to the application. Since data is not generated in real-time, they can be retrieved in bulk, well ahead of their playout deadlines. This is contrasted with live sources (*e.g.*, a video camera), that generate data at the same rate as required for consumption. Stored-data applications in a distributed multimedia information

system require synchronization for data originating from independent sources and simple data sequencing cannot provide intermedia synchronization.

In the following paragraphs, some of the scheduling algorithms for real-time traffic are discussed and their merits and demerits are analyzed.

Fair Queueing Algorithms:

The aim of fair queueing algorithm proposed by Nagle [17] is simple: if N channels share an output trunk, then each channel should get 1/N of the bandwidth, with the provision that if any channel uses less than its share, the slack is equally distributed among the rest. Nagle's scheme is unfair because sources with longer packets could hog much of the available bandwidth. This can be overcome by doing a bit-by-bit round robin service among the channels, which gives a *"Bit round Fair Queueing"*, algorithm [5]. In this algorithm, each gateway maintains separate queue of packets for each individual source. Certainly, it is impractical to send packets bit by bit. This scheme achieves both transmitting data packet by packet and maintaining fairness of round-robin. Let $R(t)$ denote the number of rounds made in a round-robin service discipline upto time t; S_i^α, F_i^α are the values of $R(t)$ at the time the packet i starts and finishes the service, respectively; and P_i is the length of the packet i in the conversation α. Therefore, $F_i^\alpha = S_i^\alpha + P_i^\alpha$. In this algorithm, whenever a packet finishes transmission, the next packet chosen to send is the one with the smallest value of F_i. This algorithm creates firewalls that protect well behaved sources from the ill-behaved ones and gives low delay to sources using less than their fair share in the total bandwidth.

Weighted Fair Queueing (WFQ) algorithm:

Whenever there is a backlog in the queue, packets leave the queue at rates proportional to the clock rates. Consider the case, where all sources are sending packets at their

clock rates, except for one source which emits a burst of packets. The WFQ algorithm would continue to send packets from the uniform sources at their clock rates, so their packets are not queued for any significant period of time, whereas the backlog of packets from the bursty source will take a long time to drain. Thus, a burst by one source will cause a long delay in the packets of that source whereas the other uniform sources will continue to receive their requested service, and would have minimal effects on the delays seen by these sources. A burst will induce jitter directly, and mostly effects the source that emitted the burst. WFQ provides a degree of isolation so that sources are protected from other sources' bursts.

The nature of play back real-time applications allows the scheduling algorithm to delay all packets upto the play-back point without adversely affecting the application's performance. Thus this play-back point serves as the deadline and for such problems the *earliest-deadline-first* scheduling algorithm has been proved to be optimal.

FIFO algorithm:

If we consider a simple example wherein a class of clients have similar service requirements, this implies that they are all satisfied with the same delay jitter. Thus, they will have the same play-back point and the same deadline [4]. If the deadline for each packet is a constant offset to the arrival time, the earliest-deadline-first scheduling algorithm becomes FIFO; the packet that is closest to the deadline is the one that arrived first. In the FIFO scheme if a burst from one source arrives, this burst passes through the queue in a clump while subsequent packets from other sources are temporarily delayed. This latter delay is however much smaller than the bursting source would have received under WFQ. Thus, the playback point need not be moved out as far to accomadate the jitter induced by the burst. Furthermore, the source producing the burst is not singled out for increased jitter; all the sources share in all

the jitter induced by the bursts of all sources. The drawback of this algorithm is that if one source injects excessive traffic into the network, this disrupts the service for everyone. Hence, FIFO discipline is ineffective for providing real-time service. FIFO is an effective sharing scheme but not an isolation scheme.

Delay Earliest-Due-Date:

In this scheduling, each packet is assigned a single deadline, and the packets are sent in order of increasing deadlines [29]. The server negotiates a service contract with each source. The contract states that if the source obeys a peak and average sending rate, then the server will provide a delay bound. The key lies in the assignment of deadlines to packets. The server sets a packet's deadline to the time at which it should be sent, had it been received according to the contract. This is just the expected arrival time added to the delay bound at the server.

Jitter Earliest-Due-Date:

The Jitter Earliest-Due-Date [29] discipline extends Delay Earliest-Due-Date to provide delay-jitter bounds (that is, a bound on the minimum as well on the maximum delay). After a packet has been served at each server, it is stamped with the difference between its deadline and the actual finishing time. A regulator, at the entrance of the next switch holds the packet for this period before it is made eligible to be scheduled. This provides the required minimum and maximum delay guarantees.

Stop-and-go queueing policy:

This scheme provides loss-free communication with guaranteed throughput and it maintains an end-to-end delay per connection which is a constant plus a small bounded jitter term [9]. This strategy provides an attractive solution for the transmission of real-time traffic and other applications which have stringent loss and delay requirements. This scheme uses a multiplexing discipline which ensures that an individual

session's output characteristics, even after being multiplexed with other sessions at a switch's output port, are the same as that session's input traffic characteristics. The approach is based on a key observation about formation of packet bursts inside a network. Congestion and buffer overflow pose serious threats to the transmission of delay sensitive traffic in packet networks. This scheme has an admission policy imposed per connection at the source node, and a particular queueing mechanism at the switching nodes, called the *stop-and-go queueing*. The admission policy requires that packet stream of each connection to possess a certain smoothness property upon arrival to the network. It is based on the notion of time frames. If a traffic transmission rate for a connection k is setup as r_k, the packets arriving in one time frame is said to be $r_k * T$ smooth with the time frame of length T. Also, the total length of admissible packets arriving in one time frame is limited to $r_k * T$, for any access link of this connection. In order to enforce this traffic smoothness along all the intermediate nodes, this policy ensures that a packet arriving in one frame at a switch is never transmitted over an output link during the same time frame over which it arrived. To satisfy this constraint a packet may have to be held (stopped) in the switch's output buffers while the output link is purposefully allowed to go idle. Stop-and-go queueing mechanism is realizable with little processing overhead and minor modification to the conventional FIFO queueing structure. The advantage is specially important at broad-band speeds where more elaborate queueing mechanisms like round robin scheduling and time slot assignment may not be convenient.

Multi-Hop Sharing:

The jitter need not increase with the number of hops as the increase in the number of hops provides more opportunities for sharing and hence more opportunities for reducing jitter [4]. The key is to correlate the sharing experience which a packet has

at successive nodes in its path. This scheme is called FIFO+. In FIFO+, FIFO style sharing is induced across all the hops along the path to minimize jitter. For each hop, the average delay seen by packets in each aggregate class at that switch, is measured. For each packet, the difference between the particular delay and the class average is computed. This difference is added to a field in the header of the packet, which thus accumulates the total offset for this packet from the average for its class. This field allows each switch to compute when the packet should have arrived, if it were indeed given average service. The switch then schedules the packet as if it arrived at this average expected time; this is done by ordering the queue by these expected arrival times rather than the actual arrival times.

The effect of FIFO+ as compared to FIFO, is to slightly increase the mean delay and jitter of flows on short paths, slightly decrease the mean delay and significantly decrease the jitter of flows on long paths which means that the overall delay bound goes down and the precision of estimation goes up on long paths. FIFO and FIFO+ differ in one important way: the queue management discipline is no longer trivial in FIFO+, but instead requires that the queue be ordered by deadline, where the deadline is explicitly computed by taking the actual arrival time, adjusting this by the offset in the packet header to find the *expected* arrival time, and then using this to order the queue. FIFO+ is also extended to multiple hops, as an explicit means to minimize the jitter by obtaining as much as benefit from sharing. The modification of FIFO+ merely extends the concept of *sharing* present in FIFO between flows at a single hop, to flows at multiple hops.

Unified Scheduling Algorithm:

This algorithm handles guaranteed, predicted and datagram service [4]. Consider a set of real-time flows, some requesting guaranteed service and also a set of datagram

sources. The scheduling algorithm at a switch takes care to isolate each of the guaranteed service flows from each other and from the predicted service flows. WFQ is used as a framework into which, the other scheduling algorithms can be fitted. Each guaranteed service client has a separate WFQ flow α, with some clock rate, r^α. All of the predicted service and datagram service is assigned to a pseudo WFQ flow, call it flow 0, with, at each link, $r^0 = \mu - \Sigma r^\alpha$ where the sum is over all the guaranteed service flows passing through that link. Inside this flow 0, there are a number of strict priority classes, and within each priority class, the FIFO+ algorithm is implemented. Once each predictive service flow has been assigned to a priority level at each switch, the scheduling algorithm is completely defined.

The effect of priority is to shift the jitter of higher priority class traffic to the lower priority classes. Datagram service is assigned the lowest priority class. Let there be K other priority levels above the datagram priority level. At the service interface, K widely spaced target delay bounds D_i for predicted service are provided. The priorities are used to separate the traffic for the K different classes. These bounds, D_i are not estimates of the actual delivered delays. Rather, they are *a priori* upper bounds and the network tries, through the admission control policies, to keep the queueing delays at each switch for a particular class i well below these bounds D_i. The above bounds are indicative of the limits on the adaptability of these adaptive applications. The *a priori* delay bound advertised to a predictive service flow is the sum of the appropriate D_i along the path. This scheme has the problem that, since delay is additive, asking for a particular D_i at a given switch does not directly mean that D_i is the target delay bound for the path as a whole. Rather, it is necessary to add up the target delays at each hop to find the target upper bound for the path. The true *post facto* bounds over a long path are expected to be significantly lower

than the sum of the bounds D_i at each hop. Since this is an adaptive service, the network should not attempt to characterize or control the service to great precision, and thus should not just use the sum of the D_i's as the advertised bound.

If the highest priority class has a momentary need for extra bandwidth due to a burst by several of its sources, it steals the bandwidth from the lower classes. The next class thus sees as a baseline of operation the aggregate jitter of the higher class. This gets factored together with the aggregate burstiness of this class to produce the total jitter for the second class. This cascades down to the datagram traffic, which gets whatever bandwidth is leftover and suffers from the accumulated jitter. Datagram traffic should be given an average rate of at least 10% both to insure that it makes some progress on the average and provide a reasonable pool of bandwidth for the higher priority traffic to borrow from during momentary overloads.

For the lower priority class, if the target goals for jitter are widely spaced then the exported jitter from the higher priority class should be an order of magnitude less than the intrinsic behavior of the class, and the classes should usually operate more or less independently. Thus, a particular class is isolated from the higher priority classes because their jitter will be so much smaller than that of the particular class. Simulation results have shown that all the guaranteed service flows received worst case delays [4]. The Guaranteed peak flows experienced much lower delays than the guaranteed average flows. Likewise, the predicted high flows experienced lower delays than the predicted low-flows.

2.3. Present Synchronization Techniques

There have been tremendous improvements in network technology resulting in higher speeds and greater reliability. With the emergence of new network standards such as

FDDI, SONET, ISDN, speeds much higher than gigabits/sec are now realizable. So the network model should be extended to support such high data rates and handle the synchronization requirements of multimedia requirements.

The new applications are characterized by their multimedia nature and the complexity of their interactions. Since the data streams belonging to these applications are not independent, they have to be handled in parallel. Since these often involve more than two partners, multicast protocols linking several entities in a network are required.

MPEG, prepared by ISO/IEC JTC 1/SC 29/WG11 [6], is the international standard for coding moving pictures and associated audio for multimedia applications. Coded audio, video and other data streams are multiplexed into an MPEG stream. MPEG and ISO 11172 are used interchangeably. The System coding layer present in MPEG, defines a multiplexed structure for combining elementary streams, including coded video, audio and other data streams and specifies means of representing timing information needed to replay synchronized sequences in real-time. Upto 32 ISO 11172 audio and 16 ISO 11172 video streams may be multiplexed simultaneously.

Since coded MPEG streams consists of continuous media streams *i.e.,* video and audio, it is necessary to synchronize these media for real-time play back. Media synchronization is defined as the occurrence in which each medium is played out at it's fixed rate determined by the type of the medium and/or the application concerned, and specified temporal relationships among associated media are maintained. There are two aspects of continuous media synchronization.

- Intra-media synchronization.
- Inter-media synchronization.

Intra-media synchronization ensures the continuity for smooth playback of each medium whereas inter-media synchronization ensures synchronization between associated media.

ISO 11172 standard specifies syntax and semantics based on conceptual decoder model, System Target Decoder (STD). This model assumes that multiplexed MPEG stream is stored on a constant latency digital storage medium and data transfer and decoding within decoder are instantaneous. ISO 11172 stream is designed such that STD will be able to decode and display elementary streams synchronously.

ISO 11172 storage medium has broad meaning including CD-ROM, magnetic hard disk, digital audio tape and computer networks etc. These storage media are of indeterministic nature in terms of delay and transmission rate instead of constant latency as assumed in STD model. In order to feed ISO 11172 decoder with stream of constant latency and a constant delay, a medium specific decoder is required.

2.3.1. ISO 11172 Stream

While MPEG video and audio specify coding of each individual stream, MPEG systems specifies the syntax and semantics of information that is necessary in order to reproduce one or more MPEG audio or video compressed data streams in a system. An ISSO 11172 stream consists of streams multiplexed together. An elementary stream consists of a number of access units (AU). The definition of the access unit depends on the medium. In the case of compressed audio an access unit is defined as the smallest part of the encoded bit stream which can be decoded by itself. In the case of compressed video, an access unit is the coded representation of a picture. A decoded access unit or decoded picture is called a presentation unit (PU). In a coded video stream, there are three types of access units: I-pictures, P-pictures and B-pictures.

I-pictures are coded without referring to other pictures. P-pictures are coded using forward prediction and B-pictures are coded using both forward and backward predictions. Due to the way video stream is coded, picture order in the coded stream may differ from the display order. The decoder must carry out reordering if needed.

An ISO 11172 stream is organized into two layers:

1. Pack Layer.

2. Packet Layer.

The pack layer is for system operations and the packet layer is for stream specific operations. An ISO 11172 stream consists of one or more packs. A pack commences with a pack header and is followed by zero or more packets. The pack header begins with a 32-bit start code and is used to store system clock reference (SCR) and bit rate information, mux_rate. The SCR is a 33-bit number, indicating the intended time of arrival of the last byte of the SCR field at the input of the system target decoder. Mux_rate is a positive integer specifying the rate at which the system decoder receives the ISO 11172 multiplexed stream during the peak in which it is included. The value of mux_rate is measured in units of 50 bytes/second rounded upwards. The value encoded in mux_rate may vary from pack to pack in an ISO 11172 multiplexed stream. The mux_rate value together with the SCR value defines the arrival time of each byte at the input to the system target decoder.

Data from elementary streams is stored in packets. A packet consists of a packet header followed by packet data. The packet header begins with a 32 bit start code that also identifies the stream to which the packet belongs. The packet header defines the buffer-size required at each elementary decoder for smooth decoding and playback of elementary stream. The packet header may also contain decoding and/or presentation

time/stamps (DTS and PTS) that refer to the first access unit in the packet. The purposes of DTS and PTS are discussed later.

The packet data contains a variable number of contiguous bytes from the same elementary stream. A data packet never contains data from more than one elementary stream and byte ordering is preserved. Thus, after removing the packet headers, packet data from all streams with a common stream identifier are concatenated to recover a single elementary stream. The multiplex of different elementary streams is constructed in such a way as to ensure that specified STD buffers do not overflow or underflow.

The system header is a special packet that contains no elementary stream data. Instead, it indicates decoding requirements for each of the elementary streams. It indicates a number of limits that apply to the entire ISO 11172 stream, such as data rate, the number of audio and video streams, and the STD buffer size limits for the individual elementary streams. A decoding system may use these limits to establish its ability to play the stream. The system header also indicates whether the stream is encoded for constant rate delivery to the STD. The system header must be the first part of the ISO 11172 stream. It may be repeated within the stream as often as necessary. Repeat of the system header will facilitate random access. Real-time encoding systems must calculate suitable values in the header before starting to encode. Non-real-time systems may make two passes over the data to find suitable values.

The MPEG systems coding specification provides data fields and semantic constraints on the data stream to support the necessary system functions. These include the synchronized presentation of decoded information, the management of buffers for coded data and random access. Random access is made possible by repeated ap-

pearance of the information needed to start decoding such as SCR, PTS and system headers and use of I-pictures. Other functions are all related to the smooth and synchronous playback of coded streams .

In STD, playback of N streams is synchronized by adjusting the playback of all streams to a master time base rather than by adjusting the playback of one stream to match that of another. The master time base may be one of the N decoders' clock, the DSM or the channel clock, or it may be some external clock. In this scheme, each presentation unit has a time stamp and presentation units with the same time stamps are displayed at the same time to achieve synchronization.

MPEG Systems provide for end-to-end synchronization of complete encoding and decoding process. This is achieved by use of time stamps, including system clock reference (SCR), presentation time stamp (PTS), decoding time stamp (DTS).

In a pro-typical encoding system, there is a single system time clock (STC) which is available to the audio and video encoders. Audio samples entering the audio encoder are encoded into audio presentation units. Some, but not necessarily all of the audio PUs have PTS values associated with them, which are samples of the STC at the time the first sample of the PU is input to the encoder. Likewise, STC values at the times when video pictures enter the video encoder are used to create video PTS fields. SCR values specify the time when the last byte of the SCR field leaves the encoder. DTSs specify the time the access units are decoded. The STD model assumes instantaneous decoding of the access units. In audio streams, and for the B-pictures in the video streams, the decoding time is the same as the presentation time and so only the PTSs are encoded; DTS values are implied.

In a pro-typical decoding system, the ISO 11172 stream arrives according to the arrival schedule specified by SCR and mux_rate fields in the pack header. The first

SCR value is extracted and used to initialize the STC in the decoder. The correct timing of the STC is maintained by ensuring that STC is equal to the subsequent SCR values, at the time the SCRs are received. Elementary access units are decoded at times specified by their DTSs and PUs are presented when their PTS values are equal to the STC value. In this way both inter-stream and intra-stream synchronization is maintained. Intra-stream synchronization is maintained by ensuring the STCs at the encoder and decoder run at the same rate. Inter-stream synchronization is maintained by presenting each PU at their specified PTS relative to STC.

All timing information is specified in terms of 90 KHz clock, which provides sufficient accuracy for audio interchannel phase alignment. The time stamps are encoded into 33 bits which are long enough to support absolute program durations of at least 24 hours.

Buffer management is also an important aspect for synchronization because if there is data starvation or buffer overflow there will be loss of synchronization. In ISO 11172, it is specified that for all multiplexed streams the delay caused by the system target decoder input buffering is less than or equal to one second. The input buffering delay is less than or equal to one second. The input buffering delay is the difference in time between a byte entering the input buffer and when it is decoded.

2.3.2. Synopsis of the Protocols for Multimedia Synchronization

Two levels of synchronization and transmission service are identified to support general purpose multimedia communications, based on the scheduling framework. These are for the *independent*-connection traffic (IC) and *aggregate*-connection traffic (AC) consisting of multiple synchronized IC traffic. For IC traffic, a service mechanism is defined for the support of applications requiring synchronization of sequences of

data of the same type or origin. This mechanism provides synchronized output for a set of data and their playout schedules. For AC traffic, a temporal specification of a multimedia object is utilized to produce playout schedules for synchronized communication of multiple classes of data. By decomposing these two levels of functionality, the IC protocol mechanism is isolated from inter-class dependencies and the temporal specification, and the AC protocol mechanism achieves an abstract interpretation of the objects and their origins rather than their protocol mechanism.

The IC protocol requires as input a set of multimedia objects, their playout times, their aggregate start time, and the desired probability of data lateness. The protocol can then provide a predicted end-to-end control time T and data transport mechanism for the set of data elements. The protocol performs the following operations:

1. Reservation and negotiation for channel capacity C.

2. Identification of current channel delay parameters.

3. Computation of retrieval schedule and required buffers based on (2).

4. Data initiation and transfer.

The AC protocol takes a selected object, decomposes it into elements of distinct traffic classes for the IC protocol, and then provides inter-class synchronization. Once an object is identified, the remaining steps required to retrieve and present the sought multimedia object are provided by the AC. These include:

1. Traversal of an object's temporal hierarchy

2. Decomposition of traffic classes based on type and location.

3. Generation of playout schedules from temporal relations.

4. Invocation of IC protocol to determine individual control times

5. Identification of maximum control time for intermedia synchronization.

6. Initiation of synchronous data transfer.

2.4. Delay Jitter Control

2.4.1. Need for Jitter Control

Delay jitter is the variation of delays with which packets traveling on a network connection will reach their destination. For good quality of reception, continuous media (audio, video, image) streams require that jitter be kept below a sufficiently small upper bound.

Digital audio, video and image-sequence transmissions, both of the interactive and noninteractive kind, need a high degree of regularity in the delivery of successive frames to the destination. Ideally, the end-to-end connections that carry those continuous media streams should delay each element (*e.g.*, a frame) by the same amount of time. This should normally be the case even when elements of the stream are not transmitted at a constant rate.

An ideal constant delay network could be characterized as a zero-jitter one, where jitter is any variation of frame delays. A more realistic requirement for continuous media transmission is that jitter have a small upper bound. A jitter bound alone will not guarantee faithful reproduction of the input pattern at the output. To achieve this result, such a bound should be accompanied by a delay bound guarantee.

The computers and networks that connect users may introduce large amounts of jitter due to the fluctuations of their loads. End-to-end delay bounds can be guaranteed by suitable design of the various components of a connection, including

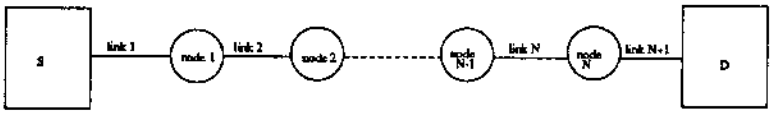

Figure 2.1: A Path in a Store and Forward Network

the host computers at the two ends.

2.4.2. Jitter Control Techniques

One of the jitter schemes proposed has time stamping of each frame by the source and the destination knowing the timestamp and the ideal constant delay, buffers each stream frame until the time it has to be delivered to the receiving user. This scheme can be implemented at the application layers as well as at the lower layers.

Another scheme proposed by Verma *et.al* [25] is a distributed scheme - thus it can only be implemented at the Network layer not by the upper protocols or by the application. The distributed scheme requires a smaller amount of buffer space and yields a more uniform buffer allocation along the path of a bounded jitter connection.

A store and forward network is assumed. A path from source to destination in such a network is depicted in figure 2.1.

Nodes 2, 3, ... N - *1* represent the switches traversed by the path. Nodes 1 and node N implement the protocol layers below the transport layer on the host systems. For these systems to offer end-to-end delay guarantees, they must be scheduled by a policy that gives higher priority to the real-time activities of the upper protocol layers, as well as to the output queues of the "resident network node", for real-time packets.

A real-time channel is characterized by performance guarantees that a client may request such as delay bound for each packet transmitted. A client has to specify the maximum traffic on the channel to be established and also the minimum time between successive packets x_{min}. This will automatically specify the client's maximum bandwidth requirements.

Each switch on the path from a source to destination will receives the value of its local delay bound $d_{i,n}$, for a channel i, and uses it to compute the local delay of each packet transmitted on that channel.

The path of a channel from the sending host to the receiving host will traverse a number of networks, interconnected by Gateways. A hierarchical model of the network is considered in which level 1 nodes are gateways. The level 1 path of the channel is a path consisting of gateways interconnected by *logical links*. In this hierarchical model a path through the network is abstracted by transforming into a simple logical link at the immediately higher level. The process of abstraction ends with a level 0 path, which is a single logical link connecting the sender and the receiver. This is the logical link whose delay and jitter are to be bounded.

The delay bound $dl_{i,l}$ of a level 1 logical link l for channel i is divided into a fixed delay $df_{i,l}$ and a maximum variable delay term $dv_{i,l}$, where $df_{i,l}$ is the delay encountered by the smallest possible packet traveling on the logical link under zero load conditions and $dv_{i,l}$ is the upper bound on the variable components of packet delays along the same link. Each level 1 node traversed by channel i has a delay bound $d_{i,n}$ assigned to it by the destination at channel establishment time. This bound is the maximum residence time of a packet in node n. The minimum residence time is the packet's processing time in the node - $dp_{i,n}$. The maximum queuing time in a node is $dq_{i,n}$. Thus we have

$$d_{i,n} = dp_{i,n} + dq_{i,n}$$

In the level 1 path of a real-time channel, all components of the end-to-end delay bound are known after the establishment of the channel. Each logical link l of channel i has a total delay bound equal to $df_{i,l} + dv_{i,l}$, and each node has a local delay bound of $d_{i,n}$ assigned to it by the destination. Thus, it is easy to calculate the maximum delay between a node and the next and to compensate for the jitter due to the variable parts of the delay, which may make the actual delay shorter than the maximum.

The mechanism for jitter control requires timestamping each packet's header by the source host, reading the clock at a packet's arrival time at a level 1 node, and keeping the packet until the *canonical* (i.e., zero jitter) arrival time at that node before giving that packet to the node's scheduler. Each node is informed, when a real-time channel is created, about the maximum delay that separates it from the source for the packets on the channel - this is the *canonical delay* $D^*_{i,n}$ for the channel i and node n. The maximum delay with respect to the canonical arrival time at the previous node (node n - 1) equals to $df_{i,l} + dv_{i,l} + d_{i,n-1}$. The node calculates the canonical arrival time by using the information present in the header of the packet and compares it with the actual one. The packet will be kept in storage for a time duration of the difference between the canonical arrival time and actual arrival time. When that interval expires that packet becomes *"schedulable"*; the jitter introduced by the scheduler in node n will be corrected exactly by the same mechanism in node $n + 1$. The only jitter that cannot be corrected in this way is that introduced by the last node in the path. This will be corrected by the receiving host, which will know the maximum delay assigned to the last node and keep the packet until that bound is reached before giving it to the application. If this is done, a *"zero - jitter"* can be achieved. If it is not done, the only end-to-end jitter is that generated by

the last node, since those produced by all the previous nodes and links are perfectly compensated by this mechanism.

CHAPTER 3

THE PROPOSED ALGORITHM

A scheduling algorithm which supports a separate virtual connection for each information type as well as control data is proposed. The multiple data streams are supported by multiple channels on an ATM network. This algorithm will also deal with the issues of *"continuity"* and *"coordination"*, in the multimedia data streams over the network. Practical considerations for applying the derived schedule are also considered.

Some general service requirements of real-time data are as follows:

- There is often a real-time interaction between the two ends of an application, as in a voice conversation. The application performance, is sensitive to the data delivery delay. In general, lower delay is much preferable.

- In order to set the play-back point, the application needs to have some information (preferably an absolute and statistical bound) about the delays that each packet will experience.

- Since all data is buffered until the play-back point, the application is indifferent as to when data is delivered as long as it arrives before the play-back point.

- The play-back applications can often tolerate the loss of a certain fraction of packets with only a minimal distortion in the signal. Therefore play back points need not be so delayed that absolutely every packet arrives before hand.

3.1. The Underlying ATM

ATM [23] is based on Asynchronous Time Division Multiplexing(ATDM). The unit of multiplexing is the cell, which is a small packet of a constant length (53 bytes). The cells carrying an identical virtual channel identifier (VCI) within their header (5 bytes) form a virtual channel (VC). An end-to-end path is established by the concatenation of VCs and a mapping between an input VCI and an output VCI is performed by the ATM switches. In ATM, the bandwidth available at a multiplexing point is not statistically shared among the VCs for a better resource utilization. This feature is particularly suitable for supporting bursty traffic. Instead of reserving a fixed capacity channel based on the peak rate, a source may request the opening of a virtual circuit, based on the average rate and still be allowed to transmit bursts.

The ATM adaptation layer (AAL) enhances the services provided by the ATM layer and supports four service classes that have been defined by CCITT. Classes A to D identify the mode of service, the bit rate and the timing needs between the source and destination. AAL type 1 offers a class A service: connection-oriented, constant bit rate, timing required between the source and destination. AAL type 2 offers a class B service: isochronous service, but with variable bit rate support. AAL type 3 offers a class C service: connection-oriented, variable bit rate and no timing requirements between the source and the destination. AAL type 4, offers a class D service: same as class C, but with connectionless mode. When no AAL is used, the service provided is the connection-oriented and non-isochronous transfer mode of the ATM layer.

Some important issues nevertheless need to be addressed for providing guarantees on cell delay and on loss probability in ATM networks. Adequate policies limiting the number of connections passing through multiplexing point as well as rate control

policies for ensuring that a source obeys its claimed rate are needed. Efficient resource management schemes and service disciplines should also be used.

3.2. Multimedia Data Streams on the ATM Network

In the approach proposed, the multimedia data streams are separated into multiple streams according to the variable transmission requests of the medium they belong to. Data streams in different groups are "independent". Each stream is transmitted on a suitable connection. The speed of the communication connection required by data of a particular medium is determined by its source rate (sample rate and sample size).

The advantages of having multiple channels for multiple data streams and having a synchronization channel are:

- An increases in parallelism and hence reduces the overall transmission time.
- Possibility of employing "lazy transmission", i.e., retrieving a part of information on demand only.
- No interference among data streams and structuring information.
- Transmission of the different data over the various channels leads to a natural and efficient usage of resources.

3.3. Properties of Time-Dependent Data

For management of time-dependent data in a computer system we are confronted with sequences of data objects which require periodic and aperiodic computation, and communication that is oriented towards presentation of information to the user.

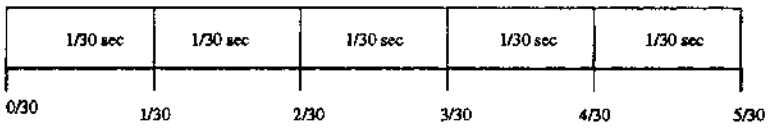

Figure 3.1: Temporal-interval Based Specification for Video Frames

A temporal model to characterize the time-dependencies of the data objects is shown in the following paragraph.

Temporal Model:

A common representation for the temporal component of time-dependent data is based on temporal intervals [14] in what is called temporal-interval-based (TIB) modeling. This technique associates a time interval to the playout duration of each time-dependent data object. For example, the playout of a sequence of video frames can be represented by using temporal intervals as shown in Figure 3.1.

A binary relationship, called a *temporal relationship (TR)*, can be identified between pairs of intervals. A temporal relationship can indicate whether two intervals overlap, abut, etc. With such a TIB modeling scheme, complex timeline representations of multimedia object presentation can be represented by application of the temporal relations to pairs of temporal intervals. Each interval can be represented by its *start time* and *end time* via instant-based modeling. Therefore, for a sequence of data objects we can describe their time dependencies as either a sequence of related intervals or as a sequence of start and end times.

3.4. Synchronization Technique Used

If we use independent channels for different data streams, the transmission time will be different on various channels. Also, the processing time, queueing delay and retransmission attempts in the transport layer from different network channels may be different. Thus, one must assume that the overall delay of information on different streams will be different. This means, for example, that in an application where a commentary (voice) accompanies the movement of a pointer (data) on a map (image) in order to explain a route, the voice may, after a while, no longer be in synchronization with the point on the map. This would destroy all usefulness of the application. Under circumstances such as these, where the information is required to be presented in the same way as it is sent, synchronization mechanisms between the data streams are necessary [22].

There are two techniques used to achieve synchronization.

The first technique consists of inserting the synchronization information within the data streams at the sender side. The advantages of this technique are:

1. Only minor modifications to the protocols are required.

2. Little overhead is necessary as it does not require any extra channel.

The disadvantages of the above technique are:

1. More buffering is required at the receiver.

2. User data is modified by the transport system.

The second technique consists of having a special connection called the synchronization channel, used for purpose of transmitting synchronization commands and

parameters. The information carried by this channel contains the control data (commands and parameters) as well as the references to the beginning and end of each control unit (unit references).

The advantage of this technique is:

- Data streams are not modified - so video equipment for e.g., may be attached easily.

The possible disadvantages are:

1. It has the drawback of requiring more bandwidth because of the extra channel.
2. Also, more processing is required as an additional channel is used.

In our algorithm we use the second technique of having an extra synchronization channel.

3.5. Issues Involved in the Scheduling Approach

Playout Timing with Delays

In the presence of real system latencies, several delay parameters are introduced. To properly synchronize some data element x with a playout time π, sufficient time must be allowed to overcome the latency λ caused by data generation, packet assembly, transmission etc., otherwise, the deadline π will be missed. If we choose a time called the *control time* T such that $T \geq \lambda$, then scheduling the retrieval at T time units prior to π guarantees successful playout timing. The *retrieval time*, or packet production time β is defined as $\beta = \pi - T$. τ represents the playout duration of the packet. The Figure 3.2 illustrates the timing parameters including latency in a multimedia information storage system.

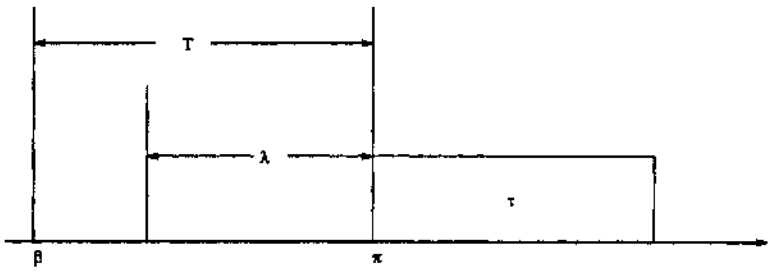

Figure 3.2: Timing Parameters including Latency

Data Arrival Timings and Nature of Delay

To develop an approach for scheduling the playout times, we must characterize the properties of the multimedia objects and the communication channels. For a typical multimedia object, the total end-to-end delay for a packet belonging to a stream i can be decomposed into three components: a constant delay $D_{p,i}$ corresponding to the propagation delay and other constant overheads, a constant delay $D_{t,i}$ corresponding to the packet size, and a variable delay $D_{v,i}$ which is a function of the end-to-end network traffic. $D_{t,i}$ is determined from the channel capacity C_i of stream i as $D_{t,i} = \frac{S_m}{C_i}$, where S_m is the packet size for medium m. The end-to-end delay for a single packet in a data stream i can be described as :

$$D_{e,i} = D_{p,i} + D_{t,i} + D_{v,i}$$

Since multimedia data objects can be very large, they can consist of many packets. If $|x|$ describes the size of some object x in bits, then the number of packets r constituting x is determined by $r = (\frac{|x|}{S_m})$. The end-to-end delay for an object x is then:

$$D_{e,i} = D_{p,i} + rD_{t,i} + \sum_{j=1}^{r} (D_{v,i})_j$$

Delay, Capacity and Synchronization Constraints

Assume that the playout of a object x in stream i is at $\pi_{x,i}$, the arrival of this packet starts at say $\theta_{x,i}$, the playout of the same object x in stream j is at $\pi_{x,j}$, the arrival of this object starts at say $\theta_{x,j}$. Assume that the retrieval time of object x in stream i is $\beta_{x,i}$ and that of stream j is $\beta_{x,j}$. There are three constraints which determine the playout time for a set of objects. These are the minimum delay (MD) constraint [13] and the finite capacity (FC) constraint [13], which are used for intra-media synchronization and the synchronization constraint (SC) constraint which is used for inter-media synchronization.

- MD constraint states that the item cannot be played out before arrival.

$$\pi_{x,i} \geq \beta_{x,i} + T_{x,i} \; i.e., \; \pi_{x,i} \geq \theta_{x,i}$$

  where $T_{x,i}$ is defined as the skew between putting a packet x onto the channel and playing it out.

- When multiple objects are retrieved from a channel it is possible to exceed the capacity of the channel and therefore we must consider the timing interaction of the objects. FC constraint determines the relation between the retrieval time and its successor when the channel is busy and accounts for the interarrival time due to transit time and variable delays. It also determines the minimum retrieval time between successive objects.

$$\beta_{x-1,i} \leq \beta_{x,i} - T_{x-1,i} + D_{p,i}$$

  This can be proved in the following manner. if the channel is busy then,

$$\forall x, \;\; \beta_{x,i} \geq \pi_{x-1,i} - D_{p,i} \Rightarrow \beta_{x-1,i} = \pi_{x-1,i} - T_{x-1,i} \text{ and}$$

if the channel is idle then,

$$\beta_{x,i} < \pi_{x-1,i} - D_{p,i} \Rightarrow \beta_{x-1,i} = \beta_{x,i} - T_{x-1,i} + D_{p,i}$$

Any object in the set can find the channel idle or with a backlog of items. If it is slack, the MD constraint is optimal, by definition. Similarly, when the channel is busy, the FC constraint is optimal. The channel is slack when $\beta_{x,i}$ occurs before $\pi_{x-1,i}$, but it can also be slack when $\beta_{x,i} \geq \pi_{x-1,i} - D_{p,i}$ due to the pipeline delays.

- Let S be the average time length of a synchronization unit and L_p the average number of bits required for the control of a single unit. Then the average bit rate R_c supported by the control command in this case should be $R_c \geq \frac{L_p}{S}$. If the maximum number of bits required for the control of one unit is L_{pmax}, the peak bit rate will be greater than L_{pmax}/S. To save the communication bandwidth a smaller value of L_{pmax} and a greater value of S is desired.

  SC constraint determines the mismatching tolerance $\tau_{i,j}$ between two data streams i *and* j $(1 \leq i \leq N, 1 \leq j \leq N)$, where N is the number of data streams. Let t_i and t_j be defined as the average random delay between the playout of two adjacent packets in stream i and j respectively. Let $\pi_{x+1,i}$ - $\pi_{x,i}$ ($\pi_{x+1,j}$ - $\pi_{x,j}$) be the time slot of one i *(j)* data packet or frame. Then the synchronization constraint with a synchronization unit of time length S is constrained by

  $$(\frac{S}{\pi_{x+1,i} - \pi_{x,i}} - 1) * t_i - (\frac{S}{\pi_{x+1,j} - \pi_{x,j}} - 1) * t_j \leq \tau_{ij}, \forall\ 1 \leq i, j \leq N$$

In order to filter the gaps, smooth the jitters and thus decrease the mismatching at the end of a control unit, an "intention delay" is placed at the beginning of

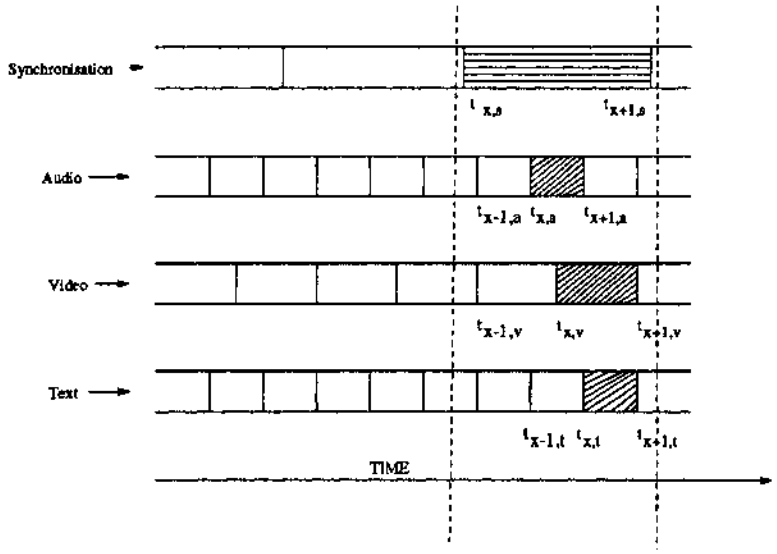

Figure 3.3: A Synchronization Session with a Synchronization Channel

each synchronization unit. That is, in a unit, the first packet of each data stream will be intentionally delayed before being played out. This extra delay helps reconstructing the data streams. The intention delay is already applied in voice packet switching systems and some video application systems. Different media require different intention delay times for recovering the gaps and jitters [2]. The synchronization delay and the intention delay are carried out concurrently. Every data stream waits for the others while performing the intention delay for itself.

The timing diagram of multiple channels with the use of the synchronization channel is shown in the Figure 3.3. In this figure, assume that the arrivals of packet numbered *x+1* in synchronization, audio, video, text streams are $t_{x+1,s}$, $t_{x+1,a}$, $t_{x+1,v}$, $t_{x+1,t}$ respectively. The audio data packet numbered *x+1* is the first packet to arrive.

This packet is delayed until all the packets numbered *x+1* from the other streams also arrive. So the playout time of packets numbered *x+1* are greater than equal to $t_{x+1,v}$, as the video packet is the last among all the packets to arrive.

Synchronization control in simultaneous real-time data delivery is essentially a *"rate matching"* problem of multimedia data streams. In order to playout all the data streams in one unit at the same time, the streams which arrive first are delayed until the remaining ones are received.

Transmission of objects is either back to back or introduces slack time, depending on their size and the time of playout. We define an optimal schedule for a set of objects in different channels. For a particular packet, say x, in different streams to be synchronized with all other streams' packets, the admission of packets should follow the following algorithm. Our admission control strategy is based on the playout times. Given the characteristics of the channel and of a composite multimedia object, a schedule can be constructed.

3.6. The Algorithm

The algorithm is a concurrent processing of all the N channels. Since we know the mismatching tolerances among different data streams, we can calculate the maximum value τ_{max} among these tolerance values and append this on to the synchronization packet. The algorithm begins by establishing an optimal retrieval time for the final, mth object in different streams, *i.e.*, $\beta_m = \pi_m - T_m$. The remainder of the schedule for adjacent objects, can be determined by iterative application of the minimum delay and finite capacity constraints, keeping in view of the synchronization constraint. Before, the retrieval of the next object, the previous β value is checked to see whether it falls within β_{min}, which is the minimum among all the β values in different streams

for a particular numbered object, and the maximum mismatching tolerance τ_{max}. If it is not within these limits, an exception_handling routine, which takes care of dropping certain frames or replaying certain frames of packets is activated.

Psuedo code of the Algorithm:

process i = 1 : N

begin

$\beta_{m,i} = \pi_{m,i} - T_{m,i}$

for j = 0 : m -2

begin

compute β_{min} among all the β_{m-j}'s ;

if $\beta_{m-j,i} \leq \beta_{min} + \tau_{max}$ then

continue

else

exception_handling();

endif

if $\beta_{m-j,i} \leq \pi_{m-j-1,i} - D_{p,i}$ then

$\beta_{m-j-1,i} = \beta_{m-j,i} - T_{j-1,i} + D_{p,i}$

else

$\beta_{m-j-1,i} = \pi_{m-j-1,i} - T_{m-j-1,i}$

endif

end

end

β_{min} can be calculated by having a barrier synchronization at which all the values of the β values for an object numbered x, have to be calculated in different streams.

The resultant schedule indicates the times at which to put objects onto different channels, between source and destination, or it can be used to determine the worst-case buffer requirements.

When the packets' numbered x of different streams, arrive at the destination, their arrival should be within the mismatching tolerances, among them. We can also determine the number of buffers required when using the playout schedule β, noting that it is necessary to obtain an element prior to its use. For each fetch time $\beta_{x,i}$, the number of buffers K_x required is equal to the size of the objects with elapsed playout deadlines between $\beta_{x,i}$ and $\pi_{x-1,i}$.

Using this method, the buffer use profile, $\beta_{x,i}$ versus K_i, and the maximum delay incurred by buffering, can be determined, which can be used for allocation of storage resources.

3.7. Implementation Issues

In this section we show how the derived schedule β can be used. First, the sender can determine the times at which to put the objects onto the channel. The receiver then buffers the incoming objects until their deadlines occur. In this case, the source must know β, the destination must know π, and the overall control time is $\pi_1 - \beta_1$ (the computed delay of the first element in the sequence). This scheme can be facilitated by appending π_i on to the objects in the synchronization channel, as they are transmitted in the form of time stamps. Another way to use β is to determine the worst-case skew between any two objects as $T_w = max(\{\pi_i - \beta_i\})$, and then to provide T_w amount of delay for every object. Transmission scheduling can then rely on a fixed skew T_w and the π schedule; that is it can transmit all items with π_i in the interval $(t \leq t + T_w)$. The first method can be useful in minimizing the buffer

utilization and delay, since the proposed schedule is derived based on these criteria. Furthermore, required buffer allocation need not be constant but can follow the buffer use profile. The second mechanism provides unnecessary buffering for objects, well ahead of their deadlines, and requires constant buffer allocation. However, it provides a simpler mechanism for implementation. Consider a long sequence of objects which can be placed on different channels, which are of identical size and with regular (periodic) playout intervals. For each of this sequence, $\pi_{1,i} - \beta_{1,i} = T_w$, and $\pi_{k,i} = \beta_{k,i} - T_w$, where k varies from 1 to total number of packets in a channel i, and assume that the channel capacity is not exceeded on each channel. In this case, the minimum buffering is also provided by the worst-case delay. Such sequences describe constant bit rate (CBR) video or audio streams which are periodic, producing fixed size frames at regular intervals. Rather than manage many computed deadlines/s (e.g, 30/s for video) a transmission mechanism can simply transmit objects based on sequence number and T_w.

If we consider data originating from different live sources, the destination has no control over packet generation times, and sufficient channel capacity must be present to preserve the real-time characteristic of the streams. In this case the control time can be determined from the size, delay, and channel capacity requirements of a representative data object, *e.g.*, a CBR video frame, and only provides jitter reduction at the receiver. For variable bit rate (VBR) objects, the source data stream is statistical in size and frequency of generated objects, and we apply a pessimistic method worst-case characterization of the largest and the most frequent VBR object to determine the control time. $\{\beta_{k,i}\}$ defines the packet production times, and playout times are $\pi_{k,i} = \beta_{k,i} + T$. The service can be supported by appending time stamps, in the form

of individual deadlines $\pi_{k,i}$ on to the objects in the synchronization channel.

In the general case, a playout schedule is aperiodic, consisting of periodic and aperiodic deadlines. Typically, during class decomposition these are isolated and one of the two scheduling schemes above can be applied. If both exist in the same playout schedule π (*e.g.,* if classes are not decomposed), then the derived schedule can be used as follows: choose a new control time T_E such that $(\pi_{1,i} - \beta_{1,i} \leq T_E \leq T_w)$, and drop all deadlines $\beta_{k,i}$ from β such that $T_E \geq \pi_{k,i} - \beta_{k,i}$ for all k. The result reflects the deadlines that will not be satisfied by simple buffering based on T_E. By choosing T_E to encompass a periodic data type (*e.g,* video stream), the burden of managing periodic deadlines is eliminated, yet aperiodic objects using extensive utilization, can be dealt with using $\beta_{k,i} - T_E$, where $\beta_{k,i} - T_E > 0$.

3.8. Summary

Multimedia applications require the ability to store, communicate, and playout time-dependent data. In this chapter, a new scheduling algorithm to satisfy timing requirements in the presence of real-time system delay given a limited band width component such as communications channel or storage device is presented. The approach uses a set of monotonically increasing playout times for the multiple channels to be considered. The playout times are shown to be derivable from a temporal-interval-based timing specification. From the timing specification a retrieval schedule is computed based on the characteristics of the limited-capacity resource.

CHAPTER 4

DISCUSSIONS

4.1. Current Architecture

The current generation of telephone networks and the current generation of computer networks were each designed to carry specific and very different kinds of traffic: analog voice and digital data. However with the digitizing of telephony in ISDN and the increasing use of multimedia in computer applications, this distinction is fast disappearing. Merging these sorts of services into a single network creates a *"Integrated Services Packet Network"* (ISPN), and would yield a single telecommunications infrastructure offering a multitude of advantages, including vast economies of scale, ubiquity of access, and improved statistical multiplexing [4]. The most difficult problem that blocks the path towards an ISPN is that of supporting real-time applications in a packet network. Real-time applications are quite different from standard data applications, and require service that cannot be delivered within the typical data rate service architecture. The salient characteristic of real-time traffic is that it requires a bound on the delivery of each packet. While this bound is statistical, the bound itself must be known a *"priori"*. The traditional data service architecture underlying computer networks has no facilities for prescheduling resources or denying services upon overload, and thus is unable to meet the real-time requirement.

Therefore, an enhanced architecture is needed for an ISPN. There are four key components for this type of architecture. The first piece of the architecture is the nature of the *commitments* made by the network when it promises to deliver a certain quality of data service. There are two types of commitments.

1. guaranteed

2. predicted

The second part of architecture is the service interface, i.e., the set of parameters passed between the source and the network. For applications to know when their packets will arrive, the service interface must include the characterization of the quality of the service the network will deliver Also, it should contain the characterization of the source's traffic, thereby allowing the network to knowledgeably allocate resources.

The third part of the architecture is the packet scheduling behavior of Network switches needed to meet these service requirements, as well as the scheduling information that must be kept in packet headers, which we have dealt in Chapters 2 and 3.

The final part of the architecture is the means by which the traffic and service requirements are established. The ability of a network to meet its service requirements is related to the criteria the network uses to decide whether to accept another request for service. Though a specific algorithm to regulate the admission of new sources is not presented, a relation is shown between other parts of the architecture and a general approach to the admission control problem is described.

4.2. Service Commitments

For a network to make a service commitment to a particular client, it must know beforehand the characteristics of the traffic that will be offered by that client. For the network to reliably meet its service commitment, the client must keep its traffic commitment (i.e., its traffic must conform to the characterization it has passed to the

network). Thus, the service commitment made to a particular client is predicated based on the traffic commitment of that client.

Guaranteed service is appropriate for intolerant and rigid clients since they need absolute assurances about the service they receive. In this case, if the network hardware is functioning and the client is conforming to its traffic characterization, then the service commitment will be met. This level of commitment does not require that other clients conform to their traffic requirements.

Guaranteed service is not necessarily required for tolerant and adaptive clients. Adaptive clients can adjust their playback point to reflect the delays their packets are receiving and in the process they are gambling that the network service in the near future will be similar to that delivered in the recent past. Any violation of that assumption in the direction of increased delays will result in a brief degradation in the application's performance as packets begin missing the playback point. Thus, an adaptive application ignores the known a priori bounds on delay and adapt to current delivered service.

Predicted Service has two components. First, if the past is a guide to the future, then the network will meet its service characterization. This component takes into account recent measurements of the traffic load in estimating what kind of service it can deliver reliably. This is in contrast to the worst-case analysis that underlies the guaranteed service commitment. Second, the network attempts to deliver service that will allow the adaptive algorithms to minimize their play back points. The intent of the second commitment is that when the network schedules packets so that the current *post facto* delay bounds are small. Predicted Service has built into it very strong *implicit* assumptions about the behavior of other network clients by assuming that the network conditions will remain relatively unchanged, but involves very few

explicit assumptions about these network clients. Thus, for predicted service the network takes steps to deliver consistent performance to the client; it avoids the hard problem, which must be faced with guaranteed service of trying to compute *a priori* what the level of the delivered service will be.

There is a third class of traffic, called the *datagram service* to which the network makes no service commitments at all, except to promise not to delay or drop packets unnecessarily.

4.3. Service Interface

There are two forms of service interface [4]. Those are

- Service Interface to guaranteed traffic
- Service Interface to predicted traffic

Service Interface to guaranteed traffic: In this interface, the source only needs to specify the needed clock rate r^α, then the network guarantees the rate. The source uses its known value of $b^\alpha(r^\alpha)$ to compute its worst case queueing delay, where b^α is the token bucket size. If the delay is unsuitable, it must request a higher clock rate r^α.

Service Interface to predicted traffic: This service interface must characterize both the traffic and the service. For the characterization of the traffic, we have the source declare the parameters (r, b) of the token bucket traffic filter to which it claims traffic will conform. Separately, the source should also specify the needed service by selecting a suitable delay D and a target loss rate L the application can tolerate. The network will use these numbers to assign the source to an aggregate class at each switch for

sharing purposes. Thus, for predicted service the parameters of the service interface are the filter rate and size (r, b) and the delay and loss characteristics (D, L).

To provide predicted service, the network must also enforce the traffic commitments made by the clients. Enforcement is carried out as follows. Each predicted service flow is checked at the edge of the network for conformance to its declared rate and burstiness; nonconforming packets are dropped or tagged. This conformance check provides the necessary isolation that is mandatory for entering a shared world. After the initial check, conformance is never enforced at later switches; this is because any later violation would be due to the scheduling policies and load dynamics of the network and not the generation behavior of the source.

In the case of predicted service, specifying the token bucket traffic filter also permits the network to estimate if it can carry the new source at the requested rate and burstiness and still meet the service targets for this, and all the existing flows. This is the function of the admission control computation.

4.4. Scheduling Behavior of Switches

The scheduling algorithms can be classified into

- Scheduling algorithms for predicted service
- Scheduling algorithms for guaranteed service

These algorithms are discussed in detail in chapter 2.

4.5. Admission Control

There are two criteria to apply when deciding whether or not to admit additional flows into the network [4]. The first admission control criterion is that we should reserve

no more than 90% of the traffic for real-time traffic, thereby letting the datagram traffic have access to at least 10% of the link; while the numerical value, 10%, of this quota is completely *ad hoc* and experience may suggest that other values may be more effective. This quota ensures that datagram service remains operational at all times; having the datagram traffic completely shut out for arbitrarily long periods of time will likely put impossible demands on the datagram transport layers. In addition, the datagram quota ensures that there is enough spare capacity to accomadate sizable fluctuations in the guaranteed and predicted service traffic. The second admission control criterion is that we want to ensure that the addition of a flow does not increase any of the predicted delays over the bounds D_i.

4.6. Summary

In this chapter, an architecture for a general scheduling algorithm is discussed. The architecture described, provides a framework for comparing various scheduling algorithms.

REFERENCES

[1] D.P. Anderson and G. Homsy, "A continuos media I/O Server and Its Synchronisation mechanism", Computer Journal, October 1991, pp. 51-57.

[2] G. Barberis, D. Pazzaglia Analysis and Optimal Design of a Packet Voice Receiver, IEEE Transactions on Communications, Vol. Com-28, No. 2, Feb. 1980.

[3] R. Chipalkatti, J. Kurose, and D. Towsley, Scheduling policies for Real-time and Non-Real-time Traffic in a Statistical Multiplexer, Proceedings of GlobeCom '89, pp. 774-783, 1989.

[4] David D. Clark, Scott Shenker, Lixia Zhang, Supporting Real-Time applications in an Integrated Services Packet Network: Architecture and Mechansim, *Computer Communications*, 1992.

[5] A. Demers, S. Keshav and S. Shenker, Analysis and simulation of a Fair Queueing Algorithm, Proc. Symp. on Communication Architectures and Protocols, ACM SIGCOMM'89, Sept. 1989.

[6] Draft International Standard ISO/IEC DIS 11172, 1992.

[7] D. Ferrari and D. Verma, A Scheme for Real-time Channel establishment in Wide-Area Networks, IEEE JSAC, Vol. 8, No. 4, pp. 368-379, April 1990.

[8] Ferrari, D., Distributed Delay Jitter Control in Packet Switching Internetworks, *Computer Communications*, 1991.

[9] S. Jamaloddin Gelostani, Congestion-Free Communications in High-Speed Packet Networks, *IEEE Trans. on Communications*, 39(12), Dec. 1991.

[10] Heijenk G.J, Hou X, Neinegeers, I.G, " Comminication Systems Supporting Multimedia Multiuser Applications ", IEEE Network, Vol 8, No 1, Jan/Feb 1994, pp. 34-44.

[11] Tohru Hoshi, Yasuhiro Takahashi, Kenjiro Mori "An Integrated Multimedia Desktop Communication and Collaboration Platform for Broadband ISDN: The Broadband ISDN Group Tele-Working System", ACM SIGCOMM, Computer Communication Review , July 1992, pp 14-15

[12] Little, T.D.C., Ghafoor, A., A Network considerations for distributed multimedia object composition and communication, *IEEE Network* , Vol 4 No 6, Nov. 1990.

[13] Little, T.D.C., Ghafoor, A., Scheduling of Bandwidth Constrained Multimedia Traffic, *Computer Communications*, May. 1992.

[14] Little, T.D.C and Ghafoor, A, Synchronization and storage models for multimedia objects, IEEE JSAC, Vol. 8, No. 3, April 1990.

[15] G.J. Lu, H.K. Pung, T.S. Chua, Mechanisms of MPEG Stream Synchronisation *SIGCOMM '94* pp. 57-67 (Jan. 1994).

[16] S.Lu and P.R. Kumar. "Distributed Scheduling Based on Due Dates and Buffer Priorities", IEEE Transactions on Automatic Control, 36, pp 1406-1416, 1991.

[17] J. Nagle On Packet Switches with Infinite Storage, *IEEE Trans. on Communications*, 35(4), pp. 435-438, April 1987.

[18] C. Nicolaou, An Architecture for Real-time multimedia Communications Systems, IEEE JSAC, Vol. 8, No. 3, April 1990.

[19] P.V. Rangan, S. Ramanathan, H.M. Vin and T. Kaeppner, "Techniques for Multimedia Synchronisation in Network File Systems", Computer Communications, March 1993.

[20] S. Sarin, I. Greif, Computer Based Real-Time Conference Systems, Computer, Oct. 1985.

[21] Wolfgang Schodl, Uwe Briem, Hans Kroner and Thomas Theimer, Bandwidth allocation mechanism for LAN/MAN internetworking with an ATM network, *Computer Communications*, Feb. 1993.

[22] D. Shepherd, M.Salmony, "Extending OSI to support syncronisation required by multimedia applications", Computer Communications, Vol 13, No. 7, September 1990 pp. 399-406.

[23] Sylvie Dupuy, Wassim Tawbi and Eric Horlait, Protocols for high-speed multimedia communications networks, *Computer Communications*, Vol 15, No 6, July/August 1992.

[24] R. Steinmetz, Synchronisation Propperties in Multimedia Systems IEEE JSAC, Vol. 8, No. 3, April 1990.

[25] Verma, D.C, Zhang, H and Ferrari, D Delay jitter control for real-time communication in a packet switching network, *Proc. TriComm '91 Conference.* (April 1991).

[26] G.M. Woodruff and R. Kositpaiboon, Multimedia Traffic Management Principles for Guaranteed ATM Network Performance, IEEE JSAC, Vol. 8, No. 3, April 1990.

[27] Wu M, Qazi N.U Ghafoor, " A synchronisation Framework for Communication of pre-orchestrated Multimedia information", IEEE Network, Vol 8, No 1, Jan/Feb 1994, pp. 52-61.

[28] Raj Yavatkar, Leelanivas Manoj, "End-to-end approach to large-scale multimedia dissemination", Computer Communications, March 1994, Vol 17, No 3, pp 205-216

[29] Hui Zhang and Srinivasan Keshav, Comparison of Rate-Based Schemes, SIGCOMM'91, Sept. 1991

[30] Lixia Zhang, Congestion Control in Packet-Switched Computer Networks, Proc. of 2nd International Conf. on Computers and Applications, .pp 273-280, June 1987.

[31] Lixia Zhang, VirtualClock: A New Traffic Control Algorithm for Packet Switching Networks *SIGCOMM '90*, pp. 19-29 (Sept. 1990).