EXPERIENCE IN DESIGNING A USING TCP* AS TRANSPORT PROTOCOL VOD SYSTEM OVER A DEDICATED NETWORK

De-Jen Lu
Department of Computer and Information Science
National Chiao-Tung University
Hsinchu, Taiwan, ROC

Yu-Chu Wang, Jan-Ming Ho, Meng-Chang Chen and Ming-Tak Ko
Institute of Information Science
Academia Sinica
11529 Nankang, Taipei, Taiwan, ROC

Abstract

In this paper, we address the problem of designing a TCP[11] and IP[10] (TCP/IP for short) based video on demand (VOD for short) system over a dedicated network based on standard internet protocols, i.e., Transmission Control Protocol (TCP for short, IETF RFC 793[11]) and Internet Protocol (IP for short, IETF RFC 791[10]). TCP and IP are usually refered together as TCP/IP. Despite of the general belief that TCP/IP is not suitable for real-time multimedia applications, after testing in laboratory environment for about one year, we are confident in announcing that our prototype is both efficient and reliable. This success depends on several important factors. First of all, the system operates on a dedicated network in which there exists no other traffic except a light of background traffic generated by the Ether Switch, the major network component used by our prototype. Second, system overhead due to the VOD server and the network adaptor card is minimized. Third, we maximize disk bandwidth by implementing a proprietary disk file system. Our experiment shows that it requires only about 330 KBytes at the client buffer to smooth packet delay jitters. In our experimental environment, due to resource limitations, up to 40 clients had been tested successfully. By looking at network bandwidth and CPU utilization, we estimate that a single VOD server should be able to handle more than 40 interactive clients playing MPEG-1 system streams simultaneously.

*http://info.internet.isi.edu:80/in-notes/rfc/files/rfc793.txt

1 Introduction

With the rapid growth in micro-electronics and network technologies, digital media service over network such as video on demand have become popular. A VOD system usually consists of a video server, the network, and a lot of clients. The video server retrieves data from disk and transports these data as video streams continuously through the network to clients in a timely fashion. The design goal is to support as many concurrent video streams as possible, subject to bandwidth limitations in CPU, disk, system bus, and network. In the case of disk, most researches focus on disk layout, disk scheduling [4, 12, 13, 6] and the integration of multiple disks to increase storage capacity. As for network, because transmission of media data is timing critical, most VOD systems use UDP[9] and IP (UDP/IP for short) with some quality of service (QoS for short) mechanisms [1, 8] to provide guaranteed real time transmission. However, it takes a long time for a new protocol to become mature [8] and it usually requires modification to the underlying operating system. In this paper, we address the problems of designing a TCP/IP based VOD sys-

Revised manuscript received September 16, 1998 0098 3063/98 $10.00 © 1998 IEEE

tem over a dedicated network. TCP/IP, a de facto network protocol standard, provides a reliable connection with flow control and error control mechanisms.

Many researchers claim TCP/IP is unsuitable for this new communication service [1, 8]. For example, Zhigang et al.[5] concluded that TCP is unsuitable for real-time data transmission for several reasons. First, the TCP congestion control algorithm controls data rate dynamically by changing its window size to prevent from overloading the network. It is thus difficult for one to apply this protocol to control the rate of a continuous media stream as desired. Second, reliable message delivery is not required for video and audio streams because they could tolerate minor frame losses. Third, TCP retransmission may cause even further packet delay jitters. However, over a dedicated network, the video server is the major source of packets flowing in the network. There are no other traffic sources competing for network bandwidth except packets generated from each VOD client and the signaling packets generated occasionally. Thus, the major locations where packet loss might occur are either at the network where packets might collide each other, or at the video clients when they don't respond to network requests quickly enough. In order to control data flow rate, we need to configure network topology appropriately and to make a proper schedule for the VOD server to transmit each video stream. We will show that this can be done easily. Though it is not necessary for the VOD system to support reliable transmission, it only takes about 200msec to retransmit a lost packet, and a 330 KBytes buffer allows 2 seconds waiting time for the retransmission of a lost packet. In other words, we are dealing with a real-time application with rather loose deadline.

In the following, we are going to present how we design and implement a true VOD system based on TCP/IP protocol suite. In section 2, we present the architecture of ASIS VOD v.1 including its network and storage subsystems. Then our design for supporting fast forward operation is present in section 3. Note that the support for backward operation is similar to that for supporting fast forward operation. In section 4, we present our experimental results. Concluding remarks are given in section 5.

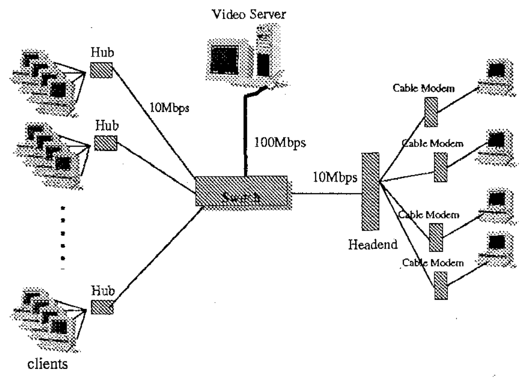

Figure 1. ASIS VOD System Architecture.

2 System Architecture

The architecture of the current ASIS VOD system is illustrated in Figure 1. The video server is a Pentium Pro200 running Linux operating system. The video server contains a ultra-wide SCSI 2 adaptor card and a 100BaseT network adapter card. The clients are either PC-486 running DOS or Pentium 133 running Windows 95. The video server connects to a fast Ether Switch via 100BaseT Ethernet. This switch has two 100BaseT ports and six 10BaseT ports. Each 10BaseT port of the fast Ether Switch connects to a 10BaseT Ethernet hub. Each hub then connects N_c video clients, where N_c is a constant which we will explain in more details in the following.

2.1 Network Subsystem

Network plays a crucial role in providing VOD service. We've briefly described the network architecture of ASIS VOD system. In the following, we'll concentrate on the decision on which transport protocol to use. First of all, because a video stream is playback at a VOD client by its MPEG-1 decoder at a specific rate $R(t)$, this transport protocol must be able to cope with the timing constraint imposed by $R(t)$. This capability is usually referred to literatures as rate control mechanism [5]. Second, this transport protocol must be able to handle packet errors. This is important especially for MPEG system streams containing multiplexed audio and video streams. Loss of packets might cause unpleasant audiovisual perceptions to different degrees depending on loss probability and type of the lost packets. For example, losing a data packet of a B-frame only makes a certain area of that particular frame to appear noisy. On the other hand, once a packet containing the header of an I-frame is lost, then the entire GOP (group of pictures, which usually consists of 6 to 45 frames of pictures) could not be recovered which might last for more than one second. For a typical MPEG decoder, it is also catastrophic to the synchronization of audio and video streams and could even mess up the internal states of the decoder. None of these problems have a trivial solution ready for engineering implementations [2].

Let's briefly consider transport protocols available in industry and academy to support real-time continuous media applications, e.g., VOD. TCP has several features such as checksum, flow control, congestion control and re-transmission to guarantee reliable transmission. Because of these features, the implementation of TCP is more sophisticated than UDP. TCP is usually considered as unsuitable for real-time multimedia applications for reasons mentioned in Section 1. There exist several new protocols such as VDP, VTP, and RTP[5, 8, 1] for supporting new applications such as VOD. However, it takes a long time to fully evaluate and for them to become generally accepted as a standard. Furthermore, portability requirements for software of the VOD server and clients are also an important consideration. TCP is a better choice considering its popularity across all kinds of platform and its reliability if timing behavior is not an issue. Furthermore, after testing the system for about a year, we are rather confident to announce that TCP/IP could be effective in supporting VOD applications as long as the network is properly configured. That is, we configure the network such that the probability of network congestion is almost zero. Although delay jitters introduced by TCP retransmission are usually considered adverse for real-time multimedia applications. However, in our prototype, packet retransmission rate is less than 0.1%. The probability of having to retransmit a packet for more than once is much lower than 0.1%. We also employ preload buffer at the clients to regulate delay jitters in network delay and in the response time for the client to process an incoming packet. For DOS-based clients, 330KBytes of buffer space are shown to be sufficient. This is roughly equivalent to 2 seconds of MPEG-1 data. Since it takes 200msec for a TCP packet sender to retransmit a lost packet, this implies that we could have 10 retransmissions before client buffer is drained. Thus, TCP retransmission is not a critical performance bottleneck of our system. As for flow control, a VOD client requests the server to send more packets whenever client buffer is not full. In this way, TCP flow rate is controlled at the playback rate of an MPEG-1 video stream. Moreover, network transmission errors are handled by TCP so that we don't have to deal with data errors at the application programs. For windows95, the client buffer is organized into a circular buffer of 40 blocks, each of $1400 * 6$ Bytes, where 1400 is also the payload size. The client buffer is pre-filled in order to regulate network delay jitters.

2.2 Storage Subsystem

In following, we introduce storage subsystem design of ASIS VOD system.

2.2.1 Disk layout and meta data

In our current implementation, a physical storage, i.e., a hard disk, is divided into three sections as shown in figure 2. In the first section, information on the entire storage is stored, e.g., number of video streams stored in this device, and meta information describing each stored video stream. In the second section, we allocate a contiguous region for each stream for storing the video stream, its fast forward version, and an index table storing the location of a particular frame in these two streams. Each contiguous region associated with a stream is said to be a file. Let N_{stream} denotes maximum number of streams stored in the disk, S_{meta} denotes the total size of meta information, S_{stream} denotes maximum size of a file, then we

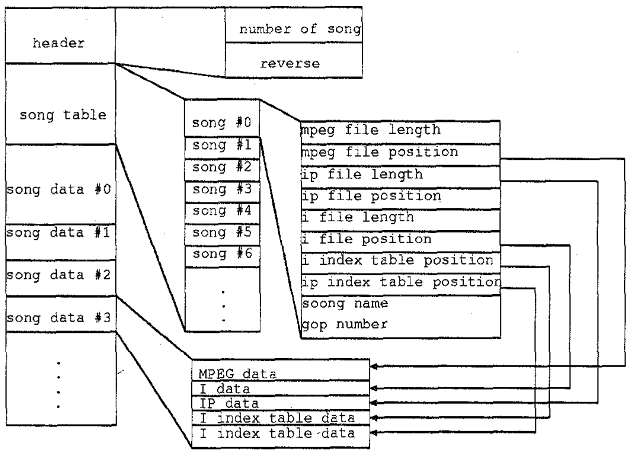

Figure 2. Data Layout.

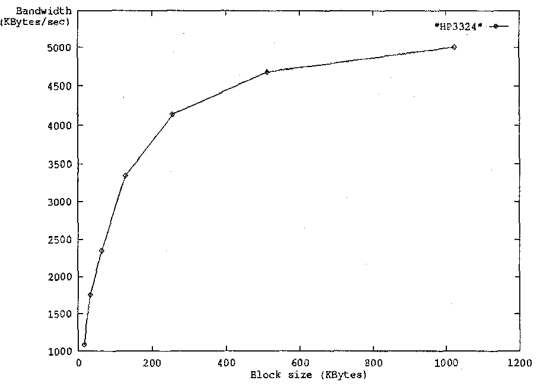

Figure 3. Disk Bandwidth with Different Buffer Size.

have: $N_{stream} * S_{stream} + S_{meta} \leq S_{disk}$, where S_{disk} is the size of the disk.

2.2.2 Block size and buffer size

The server retrieves data from disk through Linux raw device interface into a buffer, called server buffer. Data in the server buffer are then sent to the corresponding client via a TCP/IP connection. In ASIS VOD, server buffer has two blocks for each VOD client so that data retrieved from hard disk is stored into the first buffer block while data in the second buffer block is being sent to the corresponding client, and vice versa. As in typical video file server designs [7], the size of a buffer block equals that of a disk block. Note that, as depicted in figure 3, disk input and output (I/O for short) throughput increases as block size increases. Thus, number of video streams which can be serviced concurrently also increases. The cost is an increase in buffer size and the required initial delay before starting to send a video stream.

3 System Scheduling

In this section, we are going to present the system scheduler of our video server. The video server must fairly schedule I/O requests for each client to reduce I/O processing delay jitters. In current implementation, we implement all the services in a single process to avoid unnecessary context switching. There is a downstream data channel and a upstream command channel for each VOD client. To schedule I/O and network access requests, we divide the time axis into service cycles of bounded length in time as shown in figure 4. During each service cycle, the scheduler first examines for each active VOD client if it has any empty buffer and issues I/O requests to read data from disk to fill the empty buffer. It then checks for arrival of user requests and availability of data to be sent through network. After disk-reading phase, it processes outstanding network I/O requests. The newly arrived events are appended to command queue. The commands in the command queue are processed at the end of a service cycle. The maximum number of VOD clients which can be admitted to the system is bounded by the length of service round[3, 15].

4 Implementation of Client Subsystem

To satisfy different environment, our clients are implemented upon Dos, Window95, and WWW. User can use plug-in, helper, and Java to integrate into WWW browser. Figure 5 shows the data and command flows between video server and a plug-in client. Netscape navigator first sends a request to Web server after the user selects a video, the server returns the information of this video to client. Client uses this information to locate a video server

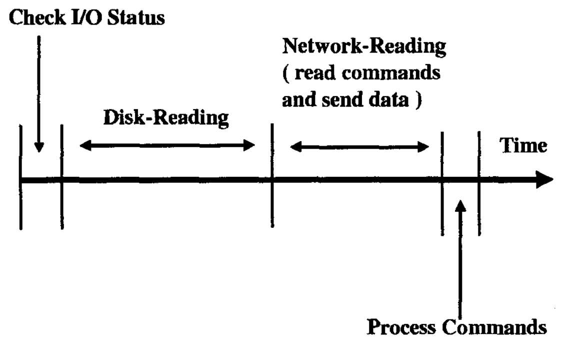

Figure 4. Service Cycle

and build both data and command channels. The video information and all commands are sent via command channel.

Our client has full VCR functions, such as play, pause, fast forward, and stop. Figure 6 shows the transition diagram. The diagram consists of eight states:

- Initial State means the client is ready and can only issue a Play command to start to play a stream and then the state changes to Normal State.
- Normal State means the client starts to display a normal stream or resumes a normal stream. User can issue Pause, Stop, or Fast-Forward command to pause, stop or fast forward the stream at this state.
- Normal & Change State means the client issues a Play command at the Fast forward state. So, the current stream changes to the normal stream instead of fast stream.
- Pause & Normal State is a state to which the client issues a Pause command at the normal state. The current steam is still the normal stream.
- Pause & Fast forward State is same as Pause & Normal State except the current state is Fast forward and the current stream is the fast stream.

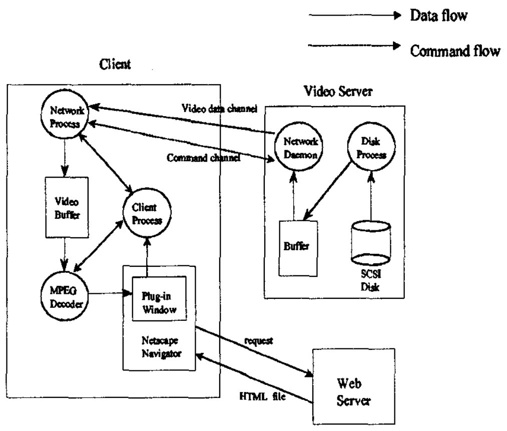

Figure 5. Data and command flows between server and client

- Fast forward state means the client resumes a paused fast stream.
- Fast forward & Change State means the client issues a fast forward command at the Normal state and changes the current stream to a fast stream.
- Stop State means the client stops the current stream. It will check if it still has any selected video. If so, it send the video number to server and display it. Otherwise, it enters the Initial State and waits for other requests.

Figure 7 shows the implementation of VCR function. For example, a Stop command first issues FMPPause and FMPStop calls to pause and stop the current stream, and then issues FMPClose to close this stream. After these steps, this stream is just closed completely, client issues SendCmd(Stop) to notify VOD server that this stream is stopped.

5 Implementation of Fast Forward Operation

There are several ways to implement fast-forward operation. Some mechanisms supporting

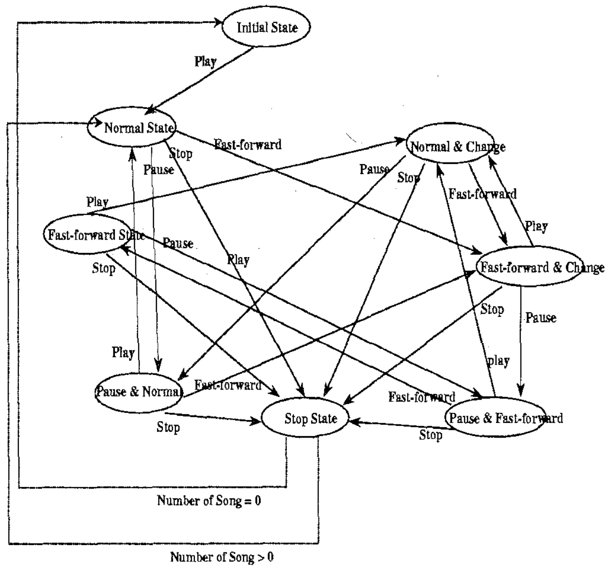

Figure 6. The State Transition Diagram of VCR

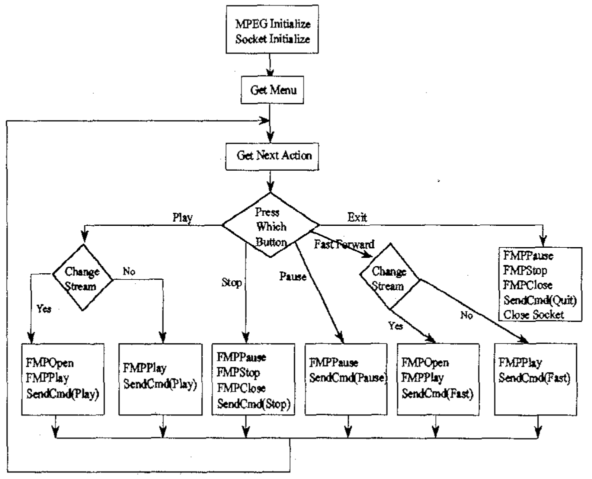

Figure 7. Implementation of VCR functions.

fast forward requires more system bandwidth than normal play that makes system bandwidth control difficult. We propose two methods which do not increase disk and network bandwidth requirements.

First, we could skip some MPEG frames. For instance, in order to support triple rate, we could skip two frames for every three frames. However, it is not suitable for MPEG stream. An MPEG stream consists of I, P, and B frames. An I frame is intra-encoded, i.e. encoded without referencing other frames; a P frame is predicted from a previous I or P frames; and a B frame is bi-directionally predicted from a previous and a later I or P frame. A sequence of frames is called a GOP (Group of Pictures). Though B frames can be discarded without hampering decoding of other frames. Once an I frame is discarded, the decoder will fail to decode the entire GOP. In MPEG format, GOP is the basic unit of random access. Skipping GOPs instead of frames to implement fast forward operation is proposed in[3]. However, visual quality of this scheme is not quite satisfactory. To overcome this problem, we propose a method which has better video quality than skipping GOPs. That is we can either discard all the B frames or we could even further discard all P frames. The server depends on the fast rate to adjust the number of frames sent per unit time. For instance, the normal MPEG-1 stream displays 30 frames per second. The twice rate can be achieved by only displaying 15 frames per second. The equation of the fast forward rate can be defined as

$$R = \frac{fps}{nfps} \times \frac{ng}{g}$$

where

R :	fast-forward rate
fps :	frame per second for current playout
g :	number of frames in the current GOP
$nfps$:	number of frames per second at normal speed
ng :	number of frames in normal GOP

And we show some results of the equation in Table 1.

One important goal of our implementation is keeping the bandwidth under 1.5 MBytes/sec, the bandwidth of MPEG-1 stream. As shown in Table 1, the bandwidth can be met this requirement as the rate under twice. To provide a higher fast forward rate, we can skip P frames in advance. The equation of fast forward rate can be redefine as:

Rate	×1	×1	×2	×3
fps	30	10	20	30
stream	normal	IP-only	IP-only	IP-only
g	15	5	5	5
time(sec)	202.47	203.41	101.70	80.05
BW(KBytes/s)	174.65	88.48	196.96	224.82

Table 1.

$$R = m * \frac{fps}{nfps} * \frac{ng}{g}$$

where $(m-1)$ denotes number of discarded frames. Table 2 shown the result according this equation.

Rate	×1	×4	×5	×6
fps	30	8	10	12
stream	normal	I-only	I-only	I-only
g	15	1	1	1
time(s)	202.47	50.99	40.82	34.01
BW(KBytes/s)	174.65	125.71	157.02	188.47

Table 2.

The disadvantage of skipping frames is a waste of disk space to store the I/P frames. Thus, we proposed the second way to *re-encode* stream of the selected frames. Using this method, we don't have to adjust the number of video frames to play at the decoder while satisfying the bandwidth constraint. And the size of the re-encode file is much less than I/P frame only and original files. Table 3 shows the size of re-encode file according to the normal and I/P only files when the fast rate is 6.

File	1	2	3	4
normal file	45420256	46373096	44516220	49187460
IP only file	30190791	30732436	28271888	32845609
re-encode file	7598596	8047453	7444970	8227849

Table 3.

6 Experiments and Experiences

In this section, we are going to present how the VOD network topology is configured, i.e., how the parameter N_c and N_m as mention in subsection 2 are computed. The tools we used are *ttcp*, *top*, and network analyzer by Network General. *ttcp* is a public domain program. We use it to generate end to end traffic to measure the capacity of an IP-based network. *top* is a command available in most UNIX operating systems. It is used to on-line monitor resource utilization, CPU utilization in specific, of a UNIX system. Using these tools, the maximum bandwidth of TCP traffic from the server to clients on a 10BaseT Ethernet segment is found to be 6.3 Mbps. And the capacity between two Pentium Pro200 running Linux connected by a 100BaseT fast Ethernet segment is around 68Mbps. We thus choose both N_c and N_m to be 4, and estimate that the server should be able to handle up to 40 interactive clients simultaneously.

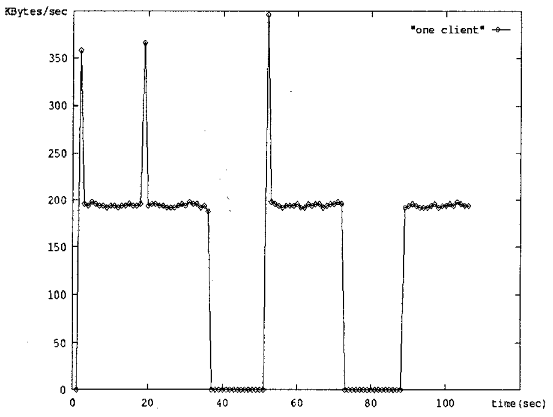

Figure 8. Network behavior of one client.

Now, let's take a look at Figure 8 a typical trace of data flow of a single interactive VOD client as measured by the network analyzer. The current implementation supports six interactive commands, i.e., movie selection, , movie deletion, play, stop, pause, and fast forward. The spikes on the plot corresponds to buffer preloading after each VCR request from the client except for pause function. The flow becomes idle when the video stream is paused by the user. It is also interesting to notice that the flow rate remains almost constant except

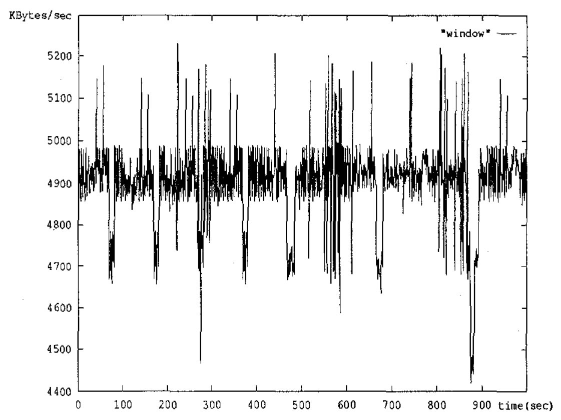

Figure 9. Network behavior of 23 clients.

for the above events.

When all our 40 VOD clients operate simultaneously, a typical trace of the aggregated network flow as measured from one 100 BaseT fast Ethernet segment at the server side is depicted in figure 9. We configure these clients on two fast Ethernet segment, one is 23 and other is 17. The maximum bandwidth is about 42.3 Mbps and the average is 39.7 Mbps. When the users are not sending interactive requests to the server, the variance of the flow rate is only 5%. Again, it verifies our previous assumption that TCP could deliver a rather smooth flow rate over a dedicated network.

7 Conclusions and Future Work

In this paper, we present our experience in designing and implementing a TCP/IP based VOD system using PCs and off-the-shelf equipments over a dedicated network. We show that in order for TCP to support a real-time application, there are two major design issues. First, the network must be properly configured. In other words, admission control is performed at system configuration time such that traffic at any spot of the entire network system will never exceed its capacity. Second, we have to guarantee the timing behavior of the application. In ASIS VOD system, this is done by analyzing the behavior of the scheduler and by inserting buffers of proper sizes at proper locations in the system. In our current implementation, a 400 KBytes buffer is allocated to each active VOD session at the server and a 330 KBytes buffer is allocated at each DOS client. The client buffer is used to preload video data such that network delay jitters are regulated and for the client to respond to an incoming packet is accommodated.

We also present our design and implementation to fast forward a video stream, and performance. Now, we had integrated this research into a digital library system. Data is going to be digital, user can retrieve paper and information from a digital library whenever he would wants to read and wherever he is. For example, user can use our VOD service to view a special talk.

Despite of the general belief that TCP/IP is not suitable for real-time multimedia applications, based on our result, we are rather announcing that TCP/IP can still operate efficiently over a dedicated network. In addition, we also present that client buffer is an important factor in designing VOD system. A larger one may cause long response time but can tolerate more retransmission delay. In our prototype, the 330KBytes buffer may cause about 2 sec delay but this is faster than accessing CDROM, and it can strongly tolerate TCP retransmission caused by the TCP delay acknowledgment, 200msec. Finally, we proposed our implementation of fast forward which enable our system to support interactive function without increasing system bandwidth.

However, TCP is still not suitable for real-time multimedia application over Internet due to there exists so much other traffic. There exists thus several special protocols such as VDP, ST-II, and RTP [5, 14, 1] for real-time transmission over Internet. As continuing work, we are implementing ASIS VOD v2 over Internet which will based on UDP/IP and RSVP protocol which can reserve the resource on the routing of video stream and can thus guarantee QoS requirement of real-time multimedia applications.

However, in a complicated network environment, there is no guarantee that network bandwidth is always under-utilized. When network is congested, data flow carried by TCP connection will most likely to be a victim in competing for network bandwidth. It thus required a more robust rate-controlled protocol to transport a real-time continuous media stream. Furthermore, resource reservation protocols are also necessary in order prevent the network from being congested. In the future, we will study the design of transport and resource reservation protocols to support continuous media applications using ASIS VOD as an intensive test suite.

References

[1] I. Busse, B. Deffiner, and H. Schulzrinne. Dynamic QoS control of multimedia application based on RTP. *Computer Communications*, January 1996.

[2] Ray-I Chang, Meng-Chang Chen, Jan-Ming Ho, and Ming-Tat Ko. Optimizations of stored VBR Video Transmission on CBR Channel. In *SPIE VV'97*, 1997.

[3] Ming-Syan Chen, Dilip D. Kandlur, and Philip S. Yu. Storage and retrieval methods to support fully inteacvtive playout in a disk-array-based video server. In *In Proceedings of ACM Multimedia*, volume 14, September 1994.

[4] Shenze Chen and Manu Thapar. Zone-bit-recording-enhance video data layout stategies. In *Processing of MASCOTS'96*, pages 29–35, Febrary 1996.

[5] Zhigang Chen, See-Mong Tan, Roy H. Campbell, and Yongcheng Li. Real-Time Video and Audio in the World Wide Web. *World Wide Web Journal*, 1, January 1996.

[6] Shahram Ghandeharizadeh, Douglas J.Ierardi, Dongho Kim, and Roger Zimmermann. Placement of data in multi-zone disk drives. In *Processing of the Second International Baltic Workshop on DB and IS*, June 1996.

[7] Ming-Huang Lee, Meng-Chang Chen, Jan-Ming Ho, and Ming-Tat Ko Yen-Jen Oyang. Design a Read/Write Multimedia On-Demand File Server. In *IS&T/SPIE Symposium on Electronic Imaging: Science & Technology*, 1996.

[8] Yasushi NEGISHI, Kiyokuni KAWACHIYA, and Kazuya TAGO. A portable communication system for video-on-demand applications using the existing infrastructure. In *Infocom96*, October 1996.

[9] J. Postel. *User Datagram Protocol*, IETF RFC 768, available by anonymous FTP from ds.inetnic.net, August 1980.

[10] J. Postel. *Internet Protocol*, IETF RFC 791, available by anonymous FTP from ds.inetnic.net, September 1981.

[11] J. Postel. *Transmission Control Protocol*, RFC 793, available by anonymous FTP from ds.inetnic.net, September 1981.

[12] A.L. Narasimha Reddy and James C Wyllie. I/O issues in a multimedia system. *IEEE Computer*, 27:17–28, March 1994.

[13] Chris Ruemmler and John Wilkes. An introduction to disk drive modeling. *IEEE Computer Magazine*, March 1994.

[14] Casner S., Lynn C., Park P., Schroder K., and Topolcic C. *Experimental Internet Stream Protocol, Version 2 (ST-II)*, October 1990.

[15] Philip Yu, Mon-Song Chen, and Dilip Kandlur. Design and analysis of a grouped sweeping scheme for multimedia storage management. In *Proc. Third International Workshop on Network and Operating System Support for Audio and Video*, 1992.

Jan-Ming Ho received his Ph.D. degree in electrical engineering and computer science from Northwestern University in 1989. He received his B.S. in electrical engineering from National Cheng Kung University in 1978 and his M.S. at Instititute of electronics of National Chiao Tung University in 1980. His research interests include realtime operating systems with applications to continuous media systems, e.g., video on demand and video conferencing, computational geometry, combinatorial optimization, VLSI design algorithms, and implementation and testing of VLSI algorithms on real designs.

Ming-Tat Ko received a B.S. and an M.S. in mathematics from National Taiwan University in 1979 and 1982, respectively. He received a Ph.D. in computer science from National Tsing Hua University in 1988. Since then he joined the Institute of Information Science as an associate research fellow. Dr. Ko's major research interest includes the design and analysis of algorithms, graph algorithms, real-time systems and computer graphics.

Dr. Meng Chang Chen received the B.S. and M.S. degrees in Computer Science from National Chiao-Tung University, Taiwan, in 1979 and 1981, respectively, and the Ph.D. degree in Computer Science from the University of California, Los Angeles, in 1989. His current research interests include multimedia database systems and knowledge base systems, knowledge discovery and knowledge representation.

De-Jen Lu received his M.S degree in Computer Science from National Chiao-Tung University. Now, he is a Ph. D. student of Computer Science of National Chiao-Tung University. His research interests include realtime operating systems with applications to continuous media systems, and network protocols.