Experience in designing a TCP/IP based VOD system over a dedicated network

Der-Jen Lu, Yu-Chung Wang, Jan-Ming Ho, Ming-Tat Ko and Meng-Chang Chen
Institute of Information Science
Academia Sinica
11529 Nankang, Taipei, Taiwan, ROC

Abstract

In this paper, we address the problem of designing a TCP/IP based VOD system over a dedicated network. Despite of the general belief that TCP/IP is not suitable for real-time multimedia applications, after testing in laboratory environment for 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 small amount 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.

1 Introduction

With the rapid growth in micro-electronics and network technologies, digital media service over network such as video on demand (VOD for short) 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, 9, 10, 6] and the integration of multiple disks to increase storage capacity and I/O throughput. In the case of CPU and I/O bus, the most powerful CPU and bus architectures such as PCI are used. As for network, because transmission of media data is timing critical, most VOD systems use UDP/IP with some QoS mechanisms [1, 8] to provide guaranteed real time transmission. However, it takes a long time for a new protocol to become popular [8] and it usually requires modifications to the underlying operating system. In this paper, we address the problems of designing a TCP/IP based VOD system over a dedicated network. TCP/IP, a de facto network protocol standard, provides a reliable connection with flow control and error control mechanisms. Although, many researchers claim TCP/IP to be 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 updating its window size to prevent its flow from overloading network traffic. 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, video server is the major source of packets flowing in the network. There are no other traffic sources competing for network bandwidth except acknowledgment packets generated from each VOD client and signaling packets generated occasionally in network components. Thus, the major locations where packet loss might occur are either at the network where packets might collide with 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 retrieve data from mass storage and for a VOD client to pull packets from the server. Our system, namely ASIS VOD v1, is designed to meet these requirements. Though it it not necessary for the VOD system to support reliable transmission. Fortunately, it takes about 200msec to retransmit a lost packet. Furthermore, we have stored 330 KBytes of video data in client buffer, we thus have about 2 seconds to wait for retransmission of a lost packet to complete. 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 the system scheduling is present in section 3. In

0-7803-4371-9/97/$10.00 © 1997 IEEE

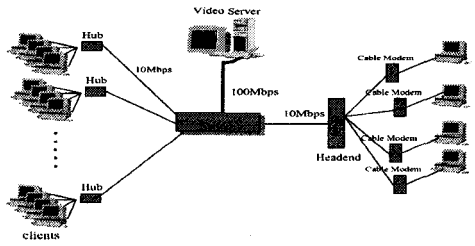

Figure 1. ASIS VOD System Architecture.

section 4, we present our experimental results. Concluding remarks are given in section 5.

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. It contains of an ultra-wide SCSI 2 adaptor card, AHA3940 by Adaptec, and a 100BaseT network adapter card, Intel Pro100B. The clients are either PC-486 running DOS or PC-Pentium 133 running Windows 95. The MPEG decoder is made by Sigma Designs and the 10BaseT network cards are 3C509 by 3Com. The video server connects to a fast Ether Switch, NBASE MEGA Switch NH208, via 100BaseT Ethernet. NH208 has two 100BaseT ports and six 10BaseT ports. Each 10BaseT port of the 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 later. One of the 10BaseT ports of NH208 connects to a cable headend, consisting of an LCT and an LCb by Bay Network, then distribute to N_m cable modems at remote sites, where N_m is also a constant.

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 in 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 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 encoder. 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 retransmission 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 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.

The client buffer is organized into a circular buffer of 40 blocks, each containing $1400 * 6$ Bytes, where 1400 is also

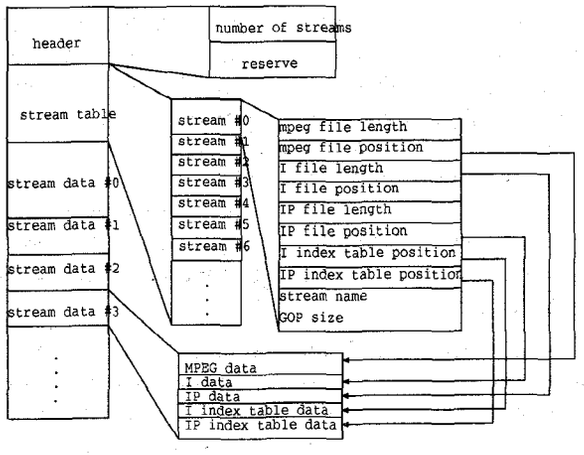

Figure 2. Data Layout.

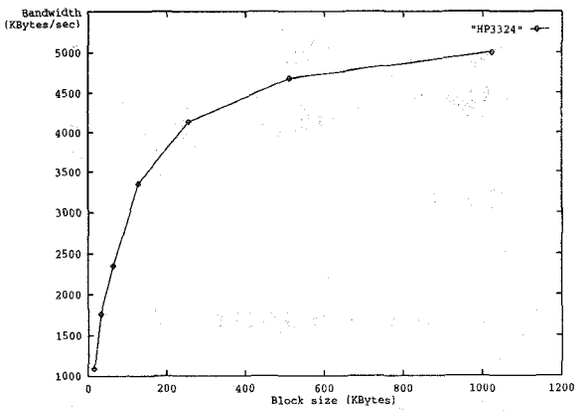

Figure 3. The disk bandwidth with different buffer size.

the payload size. The client buffer is per-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 storing the video stream, its fast forward version, and an index table in order to index between 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 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 a 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 I/O 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 avoid I/O processing delay jitters. In current implementation, we use a single process to serve all clients. There is one downstream data channel and 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 examine for each active VOD client if either one of its buffer blocks is empty. It then checks for the arrival of a user request and availability for more data to be sent to network without having to block the server process. Then, it enters the phase for each VOD client if necessary. After disk-reading phase, it processes outstanding network I/O requests. Each newly arrived event is appended to command queue. The commands in the command queue are processed at the end of a service cycle. Maximum number of VOD clients which can be admitted to the system can be computed [3, 11].

4 Experimental Results

In this section, we are going to present how the VOD network topology is configured, i.e., how the parameters N_c

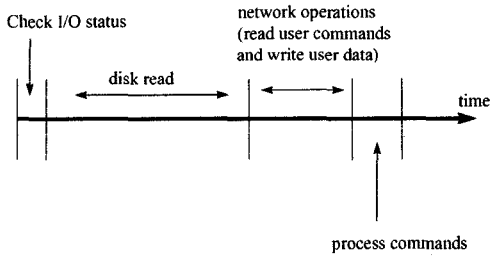

Figure 4. Service Cycle.

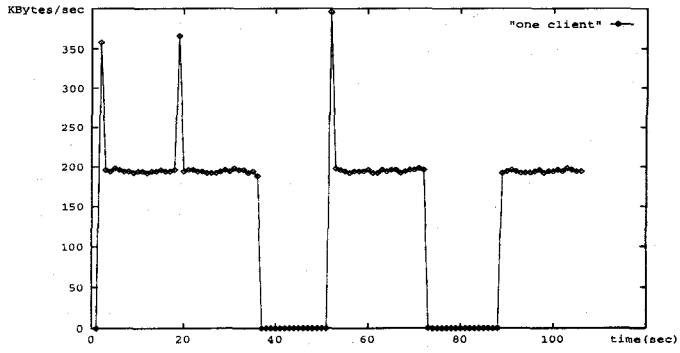

Figure 5. Network behavior of one client.

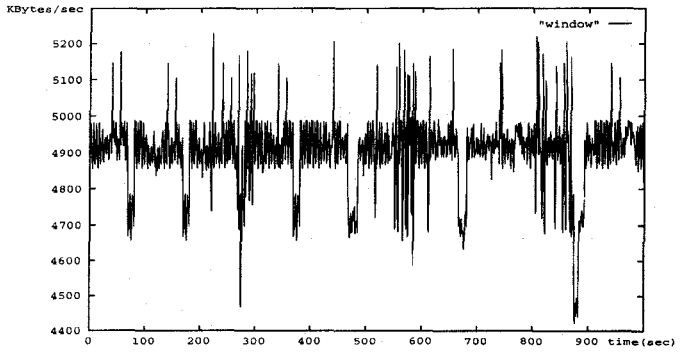

Figure 6. Network behavior of 23 clients.

and N_m as mentioned 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 N_c to be 4, and estimate that the server should be able to handle up to 40 interactive clients simultaneously. Note that the parameter N_m can be determined using similar scheme, as is set to 4 in our system.

At first, we used a 3C595 fast Ethernet adapter card by 3Com on the video server. We are quite surprised to observe jittery playback when the system drives more than 12 VOD clients. The problem is that 3C595 network adaptor card consumes too much CPU bandwidth. As a result, CPU utilization is 96% when the server drives 12 clients. After replacing the network adaptor card by an Intel PRO/100B, CPU utilization decreases to 16%. The lesson is, hardware components could be a critical factor in implementing a high performance VOD service.

Now, let's take a look at Figure 5, 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 command. 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 for the above events.

A typical trace of the aggregated network flow when all our 23 VOD clients operate simultaneously, as measured from the 100 BaseT fast Ethernet segment at the server side is depicted in figure 6. The maximum bandwidth is about 41 Mbps, the minimum is 34 Mbps and the average is 39 Mbps. When the users are not sending interactive requests to the server, the variance of the flow rate is only 4%. Again, it verifies our previous assumption that TCP could deliver a rather smooth flow rate over a dedicated network.

5 Conclusions

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 the capacity at the specific spot. 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 VOD client. The client buffer is used to preload video data to regulate the delay jitters.

We also present our design and implementation to fast forward command processing, and preliminary system performance data. Detailed performance data to give a microscopic view of the behavior of ASIS VOD system, including CPU utilization, I/O utilization and response time, and network overhead due to TCP acknowledgments and packet retransmission, will be given in the final paper.

However, in a complicated network environment, there is no guarantee that network bandwidth is always underutilized. When a network is congested, a data flow carried by a TCP connection will most likely to be a victim in competing for network bandwidth. It thus requires a robust rate-controlled protocol to transport a real-time continuous media stream. Furthermore, resource reservation protocols are also necessary in order to prevent the network from being congested. In the future, we will study the integration 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] A.L. Narasimha Reddy and James C Wyllie. I/O issues in a multimedia system. *IEEE Computer*, 27:17–28, March 1994.

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

[11] 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.