Proceedings, 6th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV), Zushi, Japan, April 1996, pp. 123-129.

VIDEO PUMP DESIGN FOR INTEROPERABILITY WITH SET TOP UNITS: THE CASE AGAINST SMALL PDUS.[1]

Stephen Jacobs and Alexandros Eleftheriadis

Department of Electrical Engineering
Image Technology for New Media Center
Columbia University
New York, NY 10027, USA
{sej,eleft}@ctr.columbia.edu

ABSTRACT

The deployment of video services to the home will continue to push the development of low cost Digital Set Top Units (STUs) for decoding MPEG video. One technique for reducing cost is to minimize the buffer space in the network interface. This means that the STU will only accept small packets. Video pump design for this situation is complex when the video pump is based on a general purpose workstation. Traditional timing mechanisms cannot be used in this situation due to the high resolution required by the small packets. A novel video pump design is presented which meets the strict STU requirements. Scalability issues are addressed and performance results are presented.

1. INTRODUCTION

One of the main concerns in developing video services is interoperability. Many institutions are working on one or more pieces of the video service puzzle [1, 3, 5, 7], i.e. Set Top Units (STUs), video servers, network protocols, etc. However, it is of the utmost importance that pieces of the puzzle from different vendors fit together in such a way that maintains quality of service for the end user.

As with the design of any system, it is possible to minimize the cost of the system as a whole by distributing the intelligence to the entities that are least prevalent in the system. In the video services scenario, this would mean giving more intelligence to the video pumps than to the STUs, since it is anticipated that there will be many more STUs than video pumps. However, if too much responsibility is given to the video pump, it could prevent general purpose high-end workstations from being used as video pumps. This would result in expensive special purpose hardware, which would most likely be much less programmable than a workstation. The added expense might preclude individuals or small companies from becoming video information providers while the lack of programmablility might mean that new services will be more difficult to disseminate quickly to the end user.

The interaction between the STU and the video pump is quite significant. One technique for both reducing STU cost and increasing video pump complexity is to minimize the buffer space in the network interface. However, this makes the video pump design much more complicated. The pump must send smaller packets more often. As the buffer space in the STU becomes smaller, it becomes more difficult for general purpose workstations to meet these requirements.

The ATM Forum also has been working on STU and video pump interaction. They have suggested that the unit of transmission, which is called a PDU, be N MPEG-2 Transport Stream packets. This suggestion is based on the fact that two MPEG-2 Transport Stream packets and a PDU trailer fit perfectly within 8 ATM cells. Our prototype STU requires $N = 2$, or 376 bytes.

In the next section, we give an overview of the environment in which our video pump is operating. This leads us to a discussion of specific problems of interoperability between video pumps and STUs. In Section 3 we present the design of our video pump. The methods used for maintaining quality of service guarantees are presented in Section 4. Section 5 provides a solution to the interoperability problem by using a nonconventional timing mechanism. This timing mechanism is the only one available to general purpose computers which can provide the resolution required by the

[1] This work was partially supported by the Army Research Office under grant DAAH04-95-1-0188.

STU. It will be shown that although this timing mechanism works quite well for a few streams, it creates problems for scalability. A solution to the scalability problem is proposed and experimental results are provided. Finally, some concluding remarks are given in Section 6.

2. VIDEO PUMP ENVIRONMENT

The video pump presented in this paper runs on a Silicon Graphics Onyx, which is a high-end general purpose multi-user workstation with 6 processors. The operating system is IRIX Release 5.3. The workstation has a file system which has 4 disks striped in parallel. It is connected to both an ATM LAN and an Ethernet LAN. On the client side there is an STU which accepts MPEG-2 transport streams over ATM Adaptation Layer 5 (AAL5). There is also a software decoder which accepts MPEG-2 transport streams over UDP/IP [1]. The ATM client will be the focus of this paper.

User level timing mechanisms on general purpose workstations are all based on signals. Signals are sent periodically to interrupt the CPU. Applications can use them to perform periodic actions. Normally, signals occur on the order of once every millisecond. The time between arriving 376 byte PDUs is 600 microseconds for a 5 Mbps high quality stream. The difference between what is available on general purpose computers and what is needed to support high bandwidth video streams using small PDUs, is the core of the problem that we address here.

In the case of our workstation, interrupts can come more quickly, as fast as one every 500 microseconds. This number is a system tunable parameter which is set at boot-time and affects all processors. The signal resolution has a limit though. Each interrupt has substantial overhead which will degrade the performance of the processor if the interrupt frequency is too high.

Even though a 500 microsecond timer may seem like enough, it is not. For example, at 500 microseconds a rate of 6 Mbps can be supported, however a rate of 5 Mbps cannot because this rate requires interrupts every 600 microseconds and the interrupts are only coming at integer multiples of 500 microseconds. This is clearly unacceptable since a video pump must be able to support clients with differing bandwidth capabilities. A client may have a software decoder that can only decode up to a maximum bit rate or may only have access to a certain amount of bandwidth. In either case the client wants to specify and receive the maximum bandwidth it can handle [4].

There is a certain amount of processing overhead each time data is sent to the network. Each send command is a system call and each system call initiates a request for services from the device driver. If the send command is called for larger PDUs, then the overhead is amortized over a larger number of bits and the resultant overhead per bit is lower. This is another reason that larger PDUs are more attractive from the video pump perspective.

Nonetheless there are also drawbacks to large PDUs. When a PDU is received in error, the entire PDU is discarded. Although ATM optical networks tend to have low bit error rates on the order of 10^{-9}, they may lose cells due to congestion at switches along the way. In the case of large PDUs, more data is lost each time there is an error. This will have an increasingly detrimental effect on the video.

3. ARCHITECTURE

Each time a client requests a service from the video server, a new video pump object is created specifically for that client. The management and control of these objects is performed via the Client Object Request Broker Architecture (CORBA Rev. 1.3) [6]. CORBA is an advanced object-oriented Remote Procedure Call (RPC) facility. Each video pump object consists of 3 separate threads, as in Figure 1. The first thread is the Control thread. Its job is to interpret commands from the client, such as pause, resume, and stop. The second and third threads are responsible for moving the data and are each run on a separate processor. The Reader thread reads data from the disks and fills the common buffers according to a round robin schedule. Meanwhile, the Writer thread reads the buffers and sends the data out to the network, one PDU at a time.

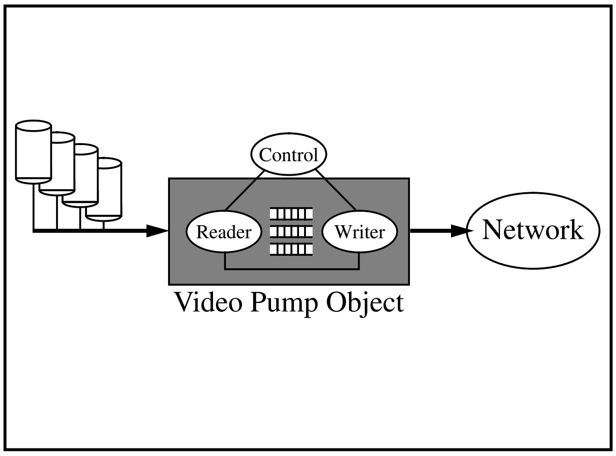

Figure 1: Video pump architecture.

The video pump object is separated into threads for both logical and performance reasons. From a logical perspective, each thread performs a task independent of the others and is therefore a separate entity. The more significant reason is performance. The small PDUs result in correspondingly small periods between PDUs. These indicate that the Writer thread is working the hardest. For this reason, it cannot suffer the added delays required for reading large blocks of data from the disk or for waiting for a client request.

The interaction between the threads is coordinated via interprocess communication, or more specifically, semaphores. For instance, when the Reader is getting data from the disk and filling one of the common buffers, it must first verify that the Writer is not currently sending data from that buffer by evaluating the semaphore. The interprocess communication also occurs when the Writer finishes the data in one buffer and must get new data from the next one. Unfortunately this means that each time the Writer needs a new buffer, it will be delayed in sending out the next PDU by the amount of time it takes to evaluate the semaphore; this takes about 200 microseconds. The alternative is to combine the Reader and Writer into one thread and suffer no interprocess communication delay. However, then the Writer would have to wait for the time it takes to actually read the data from the disk, which can be on the order of several milliseconds.

4. RESOURCE RESERVATION

The development of economically feasible video services will be essential to the evolution of the industry. A system which provides a video service must, by definition, provide guarantees on end to end quality of service. These guarantees can only be met if the underlying resources provide guarantees on their performance. Therefore, one purpose of a video server is to coordinate the usage of these resources.

IRIX Release 5.3 has several features for providing quality of service guarantees to applications. These guarantees are provided by isolating the resource for a specified period of time. When there is no operating system level provision for isolating the resource, other user-level best-effort arrangements must be made. The resources used by the video pump are CPU, memory, system bus bandwidth, disk bandwidth, and network bandwidth.

4.1. CPU

A process is allowed to isolate an entire processor such that no other regular process can run on it. This is essential to guarantee that the video pump will not be competing with other processes for time on the CPU. Multiple video pumps could also share a single processor.

4.2. Memory

Each video pump uses a small portion of memory. To minimize random delays, the program text and data segments may be locked into physical memory. Locking prevents any information from being paged out to disk. Paging happens when other user processes or system processes need to be in physical memory and there is not enough physical memory available to hold all processes.

4.3. System Bus Bandwidth

This is actually the most critical area because there is no control over it. If other users or system processes are running they may require significant access to the system bus bandwidth. This will deplete video pump resources. Also, no guarantees or reservations can be made.

4.4. Disk Bandwidth

There are no operating system level guarantees for allocating disk bandwidth in IRIX. Nonetheless, there are ways of minimizing the likelihood that a video pump object is unable to provide a specified quality of service due to a lack of disk bandwidth. The file system supports striped disks. This means a portion of the data in a file is placed on each disk so that accessing the file requires accessing all of the disks simultaneously. Disk bandwidth is thus increased in proportion to the number of striped disks.

However, there may also be jitter at the output of the disks since they are being accessed by other video pump objects, too. This jitter is smoothed by having each video pump fill two buffers with 250 milliseconds of data each before delivery to the network can begin [2]. These buffers will smooth up to 500 milliseconds of disk jitter. If the striped disks' throughput is 200 Mbps and the stream being served is 5 Mbps, it will take approximately 10 milliseconds to fill the buffer once the request is acted upon by the disk, depending on seek and rotation times. If we assume round robin scheduling then even if there are 50 other streams operating simultaneously, each one will have access to the data before their buffers empty. This user level guarantee is obtained by assuming worst case delays for seek, rotation and transfer time.

4.5. Network Bandwidth

Currently our ATM LAN does not support resource reservation and does no admission control. Nonetheless, it can be considered an isolated resource since it is used for experimental purposes.

5. HIGH RESOLUTION TIMERS

As discussed earlier, traditional timing mechanisms based on signals cannot be used for a video pump which must use small PDUs because of the high resolution required. For this reason, new timer objects had to be created. The only mechanism which could provide the resolution required by the STU is simple busy-wait timers. Busy-wait timers are a last resort since they consume extra CPU cycles. At the beginning of a period the timer is set to the appropriate expiration time. After the data has been sent out for that period, the pump waits for the timer to expire. During this waiting the timer object is continually looping and monitoring the time. It will only stop looping when it discovers that the time is greater than or equal to the expiration time.

IRIX provides a way of mapping the hardware clock into memory, thereby obtaining a clock which has an accuracy of 21 nanoseconds. Of course this level of accuracy cannot be fully exploited because of the time it takes to actually read the value. Nonetheless, the timers can measure times from 60 microseconds and up.

The obvious problem with busy-wait timers is that they will use the processor at times when ordinarily it could be released to do other things, such as running a different video pump. One possible solution is to have only one processor suffer the penalty of using a busy-wait timer by running a timer daemon on that processor. Video pumps would register themselves with this daemon and would be put to sleep and woken up by it when their period ended and began, respectively. However, the overhead due to interproccess communication is too high at about 200 microseconds, which would be incurred each time a PDU is sent.

5.1. Accuracy

An additional short-coming of having to use these timers is that they are accurate only to several microseconds. A 5 Mbps stream needs a period of 601 microseconds. If the closest we can get to this is 603 microseconds, then we will be sending too slowly. The STU will be expecting 5 Mbps as indicated within the MPEG stream, but we will be sending 4.98 Mbps. Eventually this drift will accumulate and cause blanks to appear on the display due to buffer underflow in the decoder.

The above scenario assumes that there is no Phase Locked Loop (PLL) in the STU. The PLL modulates the system clock of the decoder based on the buffer occupancy in the network interface. Slight differences in rate can be accommodated by the STU if it implements a PLL. Our prototype runs in open loop mode so that it cannot tolerate deviations from the actual rate. This is another example of how oversimplifying the decoder leads to a much more complex video server.

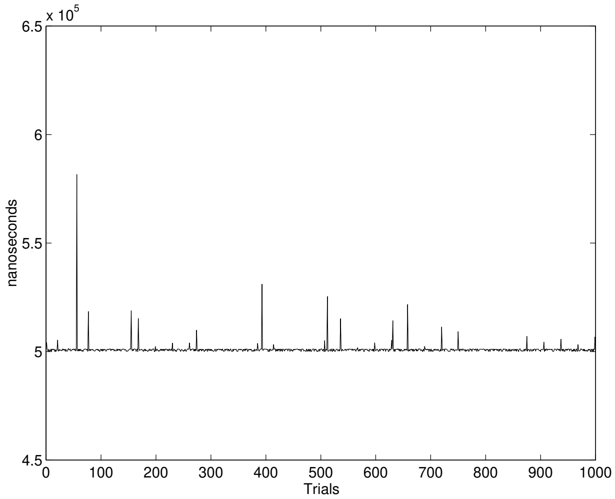

Figure 2: Characterization of the timers. Actual time waited by the timer over 1000 trials.

A characterization of the timing mechanisms independent of the video pump is given in Figure 2. The timer was repeatedly set for a 500 microsecond period. The points in the graph show the actual time waited by the timer for 1000 such trials. Clearly there is some deviation from the average. The average time waited is 501 microseconds while the standard deviation is 3 microseconds.

Since the timers are not accurate, Long Term Rate Adaptation (LTRA) must be performed. The average throughput is calculated periodically by dividing the number of bytes sent by the amount of time it has taken to send them. If the throughput is too low, then the period is decreased; if the throughput is too high, the period is increased. This adaptation creates small fluctuations around the desired throughput which ensure that the accumulated jitter does not reach an intolerable level.

Figure 3 is a diagram of the set-up for generating experimental results used in this paper. The HP is a stand alone Broadband Analyzer workstation with special purpose hardware for network analysis. It provides timing data which has a 10 nanosecond resolution. It also reassembles ATM cells into AAL5 PDUs and even into MPEG-2 Transport and Packetized Elementary Streams.

The data was sent from the video pump to the Broadband Analyzer through a single ATM switch. The Broadband Analyzer stores the data in memory as it

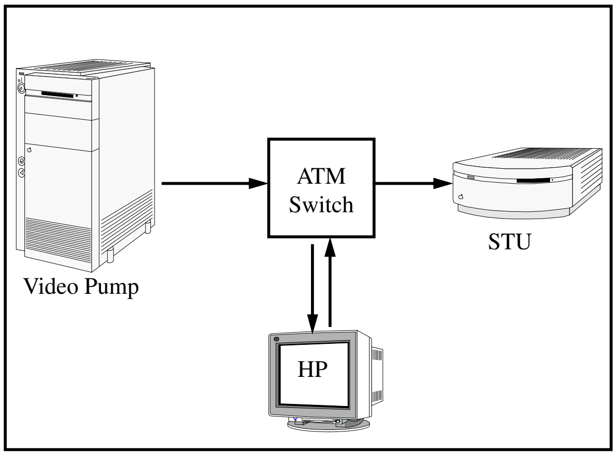

Figure 3: Experimental configuration for HP Broadband Analyzer.

receives it. Once data collection is done, it produces a list of arrival times of PDUs. We then process this list to obtain the interarrival times of the PDUs and various statistics, such as the probability mass function, the mean, the variance, and the coefficient of variation.

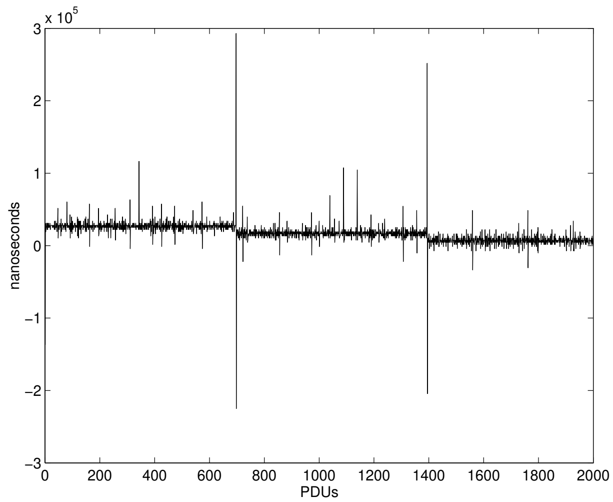

Figure 4: Deviation from the average PDU interarrival time for a 5 Mbps stream measured at the output of the video pump through one ATM switch.

Figure 4 is a graph of interarrival times of consecutive PDUs with the actual average interarrival time subtracted. The stream was sent at 5 Mbps. The standard deviation of the interarrival times is 21 microseconds. This is larger than the 3 microsecond standard deviation for the timers operating alone, as shown back in Figure 2. This difference is due in part to the sharp peaks at 700 and 1400 PDUs. The peaks are caused by the calculation of the LTRA as well as the interprocess communication overhead for switching buffers as mentioned in Section 3. As was mentioned there, this delay is much better than if the Reader thread and Writer thread were merged into one because then the delay would be that of reading from disk, which would be several orders of magnitude higher.

Immediately after the peak comes a trough. The pump attempts to make up for lost time since it missed the deadline on the previous PDU. The jitter introduced by these peaks and troughs is acceptable since the average bit rate remains constant over a small period of time.

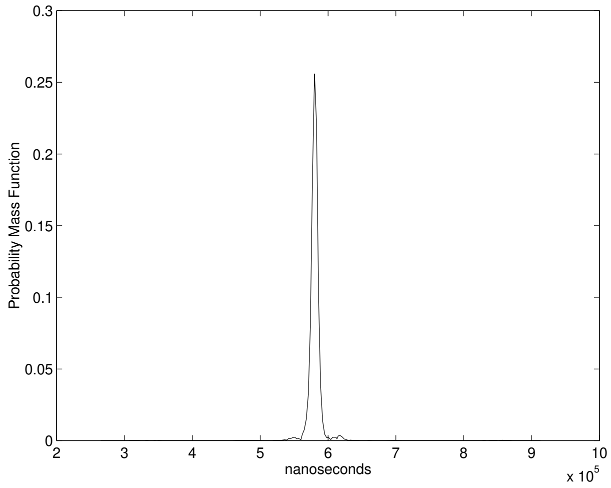

Figure 5: Probability Mass Function for the interarrival times.

The subtle staircase nature of the graph is due to the LTRA determining that the current rate is too slow and that it must decrease the interarrival time to increase the throughput. Figure 5 shows the Probability Mass Function (PMF) for the interarrival times of Figure 4. The ideal PMF would be an impulse of height one, indicating that the interarrival times were always exactly the same. In our case, we have one main peak and two side lobes which correspond to the peaks and troughs mentioned previously.

5.2. Scalability

Scalability is another problem for busy-wait timers and small PDUs. Each video pump object will use 100% of a processor since the pump is not asleep while waiting. This is clearly not an efficient use of that resource. When two pumps are run on the same processor the operating system's scheduler gives the processor to each pump for a 30 millisecond quantum [8]. This is not a constant for the scheduler, but an experimentally obtained value. This means that a pump will not be able to send out any data for 30 milliseconds. Since the average rate must remain constant, the pump must deliver data more quickly during the time it occupies the processor.

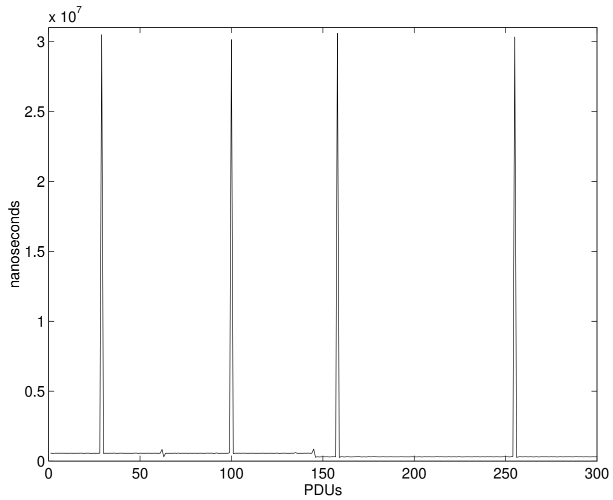

Figure 6: Interarrival times for a video pump operating at 5 Mbps while another video pump is running on the same processor.

The problem is solved by using short term rate adaptation (STRA). If a video pump senses a sudden and severe drop in average throughput, it automatically changes the rate at which it is trying to pump. If the throughput drops significantly enough, it is possible that STRA will not respond until after the decoder's buffer empties, causing a momentary blank screen. In Figure 6, STRA occurs just before the 150th PDU where there is a sudden drop off. The dropoff in interarrival times corresponds to sending the data at a higher rate. Before the dropoff the average interarrival time is about 570 microseconds; after the dropoff the average time is 280 microseconds.

The sharp peaks are the times when the video pump has been descheduled by the operating system to run the other video pump. The peaks show a 30 millisecond interarrival time, which is of course the time the pump is asleep. The average throughput from one peak to the next is still the desired throughput. However, the rate at which the pump is operating is much higher than this average to compensate for the 30 milliseconds of delay.

Let R_{f_i} be the rate at which the video pump is sending data during interval i. The video pump is attempting to maintain a rate of R_o so when the pump first starts,

$$R_{f_0} = R_o. \tag{1}$$

Now when another video pump starts on the same processor, the actual throughput for the first one will drop to $\rho < R_o$. The effort, E, is defined as the desired rate divided by the actual throughput,

$$E = \frac{R_o}{\rho}. \tag{2}$$

E indicates how much harder the video pump must work to obtain the desired rate. To obtain an actual throughput of R_o, the video pump must pump at the faster rate,

$$R_{f_{i+1}} = R_{f_i} E = \frac{R_{f_i} R_o}{\rho}. \tag{3}$$

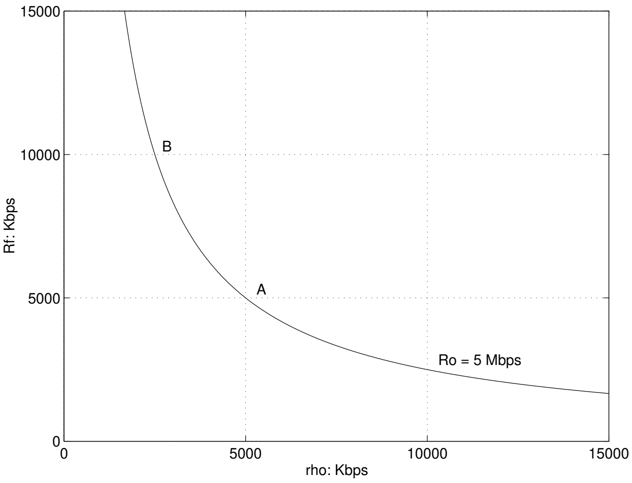

Figure 7: Example R_f curve. R_{f_1} vs. ρ for $R_o = 5$ Mbps and $R_{f_0} = 5$ Mbps.

Figure 7 is a graph of Equation 3. Point A is where the pump is attempting to deliver at 5 Mbps and the throughput is also 5 Mbps. In other words, only one pump is running on the processor. When another pump starts on the same processor, the scheduler runs each pump for 30 milliseconds. This has the effect of cutting the first pump's CPU time in half. The operating point then shifts to B on the graph. The throughput has dropped to 2.5 Mbps and the effort, E, is 2. This signifies that the pump must deliver data twice as fast. Now when the pump is active it must send data at $R_{f_1} = 10$ Mbps. If the pump knew in advance that it would get only 50% of the processor, it could simply send the data twice as fast and these calculations would be unnecessary. However, the only information available to the pump is the rate at which it is attempting to send and the actual throughput.

When one of the video pumps running on the single processor exits, the throughput of the remaining video pump will increase substantially. R_o is still 5 Mbps. However, now the throughput, ρ, jumps to 10 Mbps and $E = 1/2$. R_{f_2} is calculated to be 5 Mbps, indicating a return to our original rate. If the size of the PDU were much larger, such that the period was greater than the operating system scheduler quantum, STRA would not have to be performed at all. This is because the idle time between PDUs could be freed and used to schedule other video pumps.

6. CONCLUSION

STUs which require the use of small PDUs make video pumps based on general purpose computers more difficult to design. The problems with using small PDU sizes are many. Efficient timing mechanisms for small PDUs do not exist. Therefore, the only solution is to use busy-wait timers. These timers are problematic too, because of their lack of accuracy, their inability to scale well and their inefficiency.

One solution to this might be a programmable network interface card which could receive an entire frame of data from the video pump. The card could then be programmed to deliver that frame at a given rate. This relieves the video pump of the intensive timing responsibilities. However, until there are such network cards available, some compromise will have to be made between video servers and STUs.

Small PDUs require substantially more overhead per bit to deliver. Larger PDUs would mean less overhead and therefore a higher utilization of the CPU resource. Currently, each processor can only support an aggregate throughput of 12 Mbps due to the intense workload in having to deliver it in 376 byte PDUs. This means a maximum of 2 MPEG-2 streams per processor or 8 MPEG-1 streams.

If STUs continue to have such strict requirements, video pumps will have to be built using special purpose hardware to be scalable. This works against the basic premise of interoperability where different platforms and architectures can work together to provide a useful service to the end user.

Acknowledgments

The authors would like to thank Trevor Dyck of HP-IDACOM for providing us with the MPEG-2 support software for the HP Broadband Analyzer.

7. REFERENCES

[1] S.-F. Chang, A. Eleftheriadis, and D. Anastassiou, *Development of Columbia's Video on Demand Testbed*, Signal Processing: IMAGE COMMUNICATION, Vol. 8, No. 3, April 1996, Special Issue on Video on Demand.

[2] Asit Dan, et al., *Buffering and Caching in Large-Scale Video Servers*, 1995 IEEE COMPCON: Technologies for the Information Superhighway, Mar 5-9, 1995, pp. 217–224.

[3] Daniel Deloddere, et al., *Interactive Video on Demand*, IEEE Communications Magazine, May, 1994, pp. 82–88.

[4] Alexandros Eleftheriadis and Dimitris Anastassiou, *Meeting Arbitrary QoS Constraints Using Dynamic Rate Shaping of Coded Digital Video*, Proceedings, 5th International Workshop on Network and Operating System Support for Digital Audio and Video, Apr. 18–22, pp. 95–106.

[5] Vivek Nirkhe and Mark Baugher, *Quality of Service Support for Networked Media Players*, 1995 IEEE COMPCON: Technologies for the Information Superhighway, Mar 5-9, 1995, pp. 234–238.

[6] Object Management Group, *The Common Object Request Broker: Architecture and Specification*, Revision 1.3.

[7] K. K. Ramakrishnan, et al., *Operating System Support for a Video-on-Demand File Service* , Multimedia Systems 1995, No. 3, pp. 53–65.

[8] Raj Yavatkar and K. Lakshman, *A CPU Scheduling Algorithm for Continuous Media Applications*, Proceedings, 5th International Workshop on Network and Operating System Support for Digital Audio and Video, Apr. 18–22, pp. 223–226.