A Practical Methodology for Guaranteeing Quality of Service for Video-on-Demand

Javier Zamora, Stephen Jacobs,
Alexandros Eleftheriadis, Shih-Fu Chang and Dimitris Anastassiou

Manuscrit received February 3, 1997
The authors are with the Department of Electrical Engineering, Columbia University, New York, NY 10027-6699 USA (e-mail: javier@ee.columbia.edu).

Abstract

A novel and simple approach for defining end-to-end Quality of Service (QoS) in Video-on-Demand (VoD) services is presented. Using this approach we derive a schedulable region for a video server which guarantees end-to-end QoS, where a specific QoS required in the video client translates into a QoS specification for the video server. Our methodology is based on a generic model for VoD services which is extensible to any VoD system. In this kind of system, both the network and the video server are potential sources of QoS degradation. Specifically, we examine the effect that impairments in the video server and video client have on the video quality perceived by the end user. The Columbia VoD testbed is presented as an example to validate the model through experimental results. Our model can be connected to network QoS admission control models to create a unified approach for admission control of incoming video requests in the video server and network.

Keywords

Video-on-Demand, End-to-End QoS, ATM, Admission Control.

I. Introduction

Video-on-Demand (VoD) will become one of the most important services in high speed networks. Proof of this is the recent standardization efforts [1], [2] and the many recent publications [3], [4], [5], [6]. A key issue in any video service is to provide an acceptable Quality of Service (QoS) to the end-user. This QoS generally implies various aspects, such as the frame loss frequency, blocky effect, audio and video synchronization (lip synchronization), chroma stability (i.e. in NTSC display systems). Some of these parameters are not easily quantifiable since they depend on the subjective perception of the viewer. The QoS at the video client reflects how the original video stream has been delivered from the remote video server, where concepts of semantic transparency and time transparency [7] characterize the performance of video services over networks. Such services require specific constraints regarding the delay (time transparency), specifically the delay variation or jitter, experienced across the connection, as well as constraints regarding the rate of errors (semantic transparency), from video server to video client. Therefore, an important question for VoD is how to map a specific QoS required in the video client into a QoS specification for the video server and network.

We present a novel and simple approach for mapping QoS from video server to video

client by using a generic model which can accommodate any VoD system. We then provide an example of this methodology by applying it to the Columbia VoD testbed [8]. The goal of our model is to provide an end-to-end QoS. In other words, we are examining how impairments in the video server and network affect the quality of the video perceived by the end user. Since VoD is a point-to-point service, we provide QoS guarantees per individual stream rather than over an aggregate video traffic in the video server or network switch.

Our generic VoD model consists of three components: video server, network and video client. The video server model is based on the video pump architecture, network interface and operating system. The network model uses the concept of the schedulable region [9] which guarantees QoS with an efficient use of resources. The video client model includes the network interface, memory, and error concealment component. From the QoS point of view, each of these components has its own characteristic parameters. For instance, the video server's parameters are the bit rate, burstiness, and autocorrelation of the video streams, as well as the conditions in the system, such as number of video streams currently active and the impact of other processes running on the system. The performance in the network can be described by several parameters such as probability of loss and error, end-to-end delay, jitter, and burst tolerance. In the video client, frame loss, lip synchronization, blocky effect, picture distortion and chroma stability are the parameters that define the QoS perceived by the final viewer. It is very important to map the parameters from each of these domains into those of the video client, i.e., a certain jitter distribution will be translated to a specific frame loss rate.

We have to note that the scales in the three domains are different. The video server handles Protocol Data Units (PDUs) whose size can range from 376 bytes, per the ATM Forum [1] specification, to several kB; the Asynchronous Transfer Mode (ATM) network operates with 48-byte-payload cells; and the video client works in the frame domain with an average size of 200 kb for a 6 Mbps video stream. These differences in scale mean that the loss of one cell will cause the loss of an entire PDU. This same PDU loss could cause the loss of an entire frame or Group of Pictures, depending on the position of that PDU in the transport stream. Also, losing an I frame would be more detrimental than losing a

B frame. Therefore, not all losses have the same impact on the final QoS.

The organization of this paper is as follows. The generic VoD model is presented in Section II, where we describe the three basic components: video server, network and video client. In Section III the Columbia VoD Testbed is presented as an example of how a real system fits into the model of the previous section. Starting from several real-time video traffic measurements, Section IV identifies which of several QoS parameters provide a comprehensive description of the performance of a video server. Section V presents a novel mapping from the video server QoS parameters identified in Section IV to the video client QoS parameters. In Section VI, this mapping is used to define a video server schedulable region and the specific example of the Columbia VoD testbed is again used. In Section VII some concluding remarks are given.

II. A Generic Video-on-Demand Model

In this section we introduce the three basic components of a generic VoD system model: video server, network and video client. Our model provides a generic abstraction and captures the common elements that can be found in a any VoD system, avoiding elements that are dependent upon a specific implementation or platform.

A. Video Server

Fig. 1 shows the three main components of the video server model: video pump, network interface and operating system. A video stream is stored on a disk as a consecutive number of video packets (i.e., MPEG-2 transport packets), the function of the video pump is to read this video stream and write the video packets into the network interface. The video pump attempts to preserve the timing information inside the stored video. For instance, in the case of a Constant Bit Rate (CBR) video stream the video pump must deliver the video packets to the network according to the rate of the stream. An ideal video pump generates the same video traffic as that generated from a real-time encoder. Therefore the critical issue in a video pump is its ability to reproduce the original traffic pattern, which implies a coordination between disk accesses and network writings. One video pump engine is associated with each video stream. The output of each video pump contends for the network resource through the network interface. The mission of the network interface

is to resolve contention among video packets from different video pumps using a scheduler, to construct the PDUs with the video packets pumped out from a particular video stream, and to transmit those PDUs over the network.

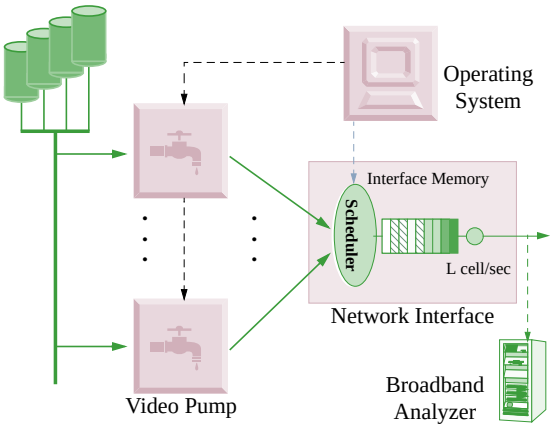

Fig. 1. Video Server Model.

A video server could introduce some jitter into the traffic pattern of the stream, due to contention for common resources, i.e., network interface and disk access. It is more likely to happen however in the case of a software implementation where there are additional tasks in the operating system. For this reason, we model the video server as a virtual multiplexer, where several sources are accessing a multiplexing stage (network interface). In a software-based implementation, the interaction with other processes that use common resources such as the bus or the disk causes additional jitter to the video sources. This effect can be modeled as cross-traffic accessing the multiplexer with a priority (scheduler) dictated by the operating system.

A software implementation provides a high degree of interoperability and ease of deployment because it can be easily adapted to different platforms and networks, dramatically reducing the cost of such systems. For this reason, it is expected that many video servers will be based on software implementations. However, general purpose workstations do not have real-time operating systems and this has a detrimental effect on the traffic of the transmitted video stream.

This video server model allows us to consider the video server as the first switch of the connection. This has the advantage of simplifying admission control since the behavior

of the video server can be characterized as an additional switch of the connection with a specified QoS performance. The QoS degradation not only occurs in the network switches but also in the video server itself. The video server QoS degradation can be dominant in scenarios where the network utilization is low, as is in many private local ATM networks.

B. Network

With regard to the network model we use the approach described in [10], [11] which provides QoS guarantees with efficient use of resources. For this purpose the concept of schedulable region is introduced in [9], which defines the possible combination of loads for each traffic class under a QoS constraint (i.e., loss and delay). Although this admission control policy is different than that which is specified by the ATM Forum [12], it is possible to map the schedulable region into ATM Forum admission policy. This implies mapping the different service classes into network traffic classes. In Section VI-A, we extend this concept of schedulable region to provide an admissible region for the video server based on experimental results of QoS measurements.

C. Video Client

Fig. 2 shows a diagram of our Video Client Model. It consists of four sections, Reception, Demultiplexing, Decoding and Presentation. Reception comprises the network interface that receives the PDUs from the network. The buffer size of the network interface is on the order of several PDUs. In some digital set-top-boxes this size is only 376 bytes (8 ATM cells using AAL5). This size prevents the server from sending larger PDUs. The output of the network interface is connected to the smoothing buffer to eliminate the jitter between the PDUs. Generally a Phase-Locked Loop (PLL) driven by the smoothing buffer occupancy is used to smooth the accumulated jitter in the server and network, and the jitter caused by independent clocks used in the server and client [13]. The output of this buffer is an MPEG-2 transport stream which is demultiplexed into the corresponding video and audio streams. In the Decoding stage, the video and audio streams feed the Video Decoder and the Audio Decoder, respectively. Finally in the Presentation Stage the video and audio are played.

The whole system is controlled by two subsystems: Time Recovery and Error Conceal-

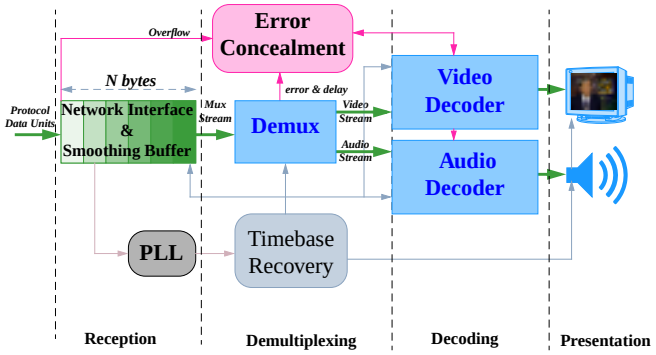

Fig. 2. Video Client Model.

ment. The Time Recovery subsystem generates different levels of synchronization (i.e. demux, decoding and presentation) from the PLL signal and the time stamps inside the incoming stream. The Error Concealment subsystem takes the appropriate actions (i.e., repeating previous frame) in the case of some error condition. Such error conditions can be caused by an overflow in the network interface buffer (loss of PDUs), underflow of the reception buffer causing loss of synchronization, and error in the received streams. Some parameters of this model can be scaled depending on the type of video client used. Memory constraints in digital set-top-boxes can be eliminated in software-based implementations. For instance, a PC network interface card has substantially more memory.

In the next section, the Columbia VoD testbed is described as an example implementation of a VoD system, and is used to show how the model can be applied to a real system as well as how design constraints can affect the overall QoS performance.

III. An Example Implementation: Columbia VoD Testbed

Columbia's VoD testbed is designed with advanced features of video storage, coding, manipulation, transmission and retrieval [8]. The main objective is to use this testbed as a platform for multimedia research and application development such as Columbia's Electronic News System, Digital Libraries, Interactive Video Courses on Demand (see Fig. 3). We use a broadband analyzer to measure performance parameters such as the delay jitter at different points in the network. Moreover, the broadband analyzer can

emulate network impairment conditions (i.e., cell losses) on real video services in the testbed.

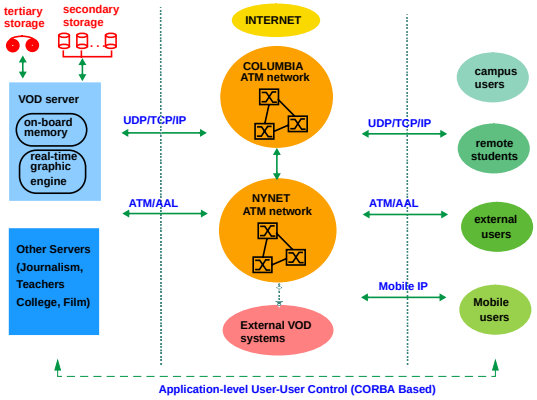

Fig. 3. Columbia's Multimedia Testbed.

A. Testbed Architecture

In Fig. 4, the Columbia VoD testbed architecture is shown. We identify the three components of the VoD model explained in Section II.

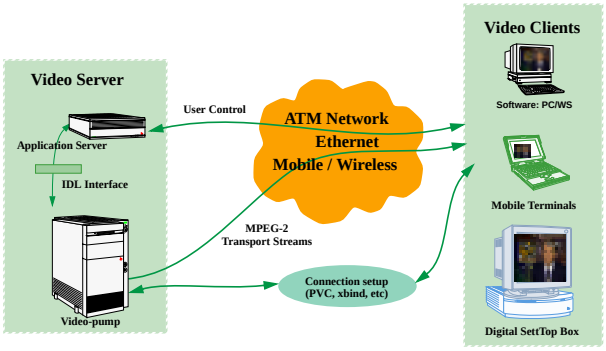

Fig. 4. Columbia's VoD Testbed Archictecture.

A.1 Testbed Video Server

In general there will be many more video clients than video servers. Therefore, it is beneficial to give more intelligence to the video server than to the video client. However,

if too much responsibility is given to the video server, it could prevent the use of general purpose workstations for this purpose. This would result in expensive special purpose hardware, which is usually much more difficult to program. The added expense precludes individuals or small companies from becoming video information providers. The lack of programmability means that new services will be more difficult to implement and deploy.

Our server is implemented in software using a general purpose UNIX workstation. Specifically it includes a Silicon Graphics ONYX 6-processor system, running IRIX 5.3. All the video material is stored in an array of disks in compressed form. We use MPEG-2 software encoder or real-time hardware encoder to compress the video sequences, prepare the video elementary streams as well as MPEG-2 transport streams [14].

Our video server has the capability to pump both CBR and VBR video streams. A video pump is implemented as a software object. Each time a client requests a stream from the video server, a new video pump object is created for that client. Each video pump object consists of 3 separate threads, as shown in Fig. 5. The first thread is the Control thread. Its job is to interpret commands from the client, such as play, pause, resume and stop. The second and third threads are responsible for moving the data from the disk to the network interface. The Reader thread reads data from the disks and fills the shared buffers according to a round robin schedule. Meanwhile, the Writer thread reads the buffers and sends data out to the network interface, one PDU at a time.

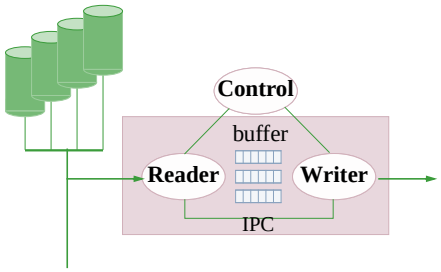

Fig. 5. Video Pump Object.

Our video server uses an absolute timer scheme to send out the PDUs according to a specific interarrival pattern. At time t_0 the interarrival time is set to a period, T, between two consecutive PDUs. The first PDU is sent and the pump waits for the timer to expire. The timer, t_e, will only expire when it is equal to $t_0 + T + \Delta t$, where Δt is the time by

which the timer overshoots the expiration time. At this point one PDU is sent out and the timer is set to $t_e - \Delta t + T$, to negate the effect of the timer overshoot.

From the perspective of the video client, absolute timers enforce a constant average bit rate. This reduces the potential for underflow in the decoder in the absence of a PLL mechanism. If one PDU is sent out late with a delay of Δt, the absolute timer will compensate for this delay by trying to send out the next PDU with a period $T - \Delta t$. An advantage of using absolute timers is their ability to mask the inconsistencies of the resources that are logically below them. For instance, if the time it actually takes to send out a PDU has a large variance, an absolute timer attempts to compensate for this. If the network interface delays the sending of a PDU, the absolute timer will make the period for the next PDU smaller. The variance of the underlying resource is therefore reduced since it is being corrected by the changing period in the timer. However, this also introduces jitter and PDU clumping as will be seen in Section IV-B.

For the network interface, we use *ForeRunner* SBA-200 ATM VMEBus Adapter which resolves the contention among the outputs of the different active video pumps. The output of the adapter uses a TAXI line as the physical interface. Since our implementation is software-based, the operating system affects with the performance of the video pump objects and the network interface. For instance, the interaction between the threads is coordinated via interprocess communication (IPC), or more specifically, semaphores. When the Reader is getting data from the disk and filling one of the common buffers, it must verify that the Writer is not currently sending data from that buffer by evaluating the semaphore. The IPC also occurs when the Writer finishes data in one buffer and must get new data from the next one. This implies that each time the Writer needs a new buffer, it will be delayed by the amount of time it takes to evaluate the semaphore; this takes about 200 microseconds in our system. This delay is periodic and it is a trade-off between the size of the buffer and the rate of the video pump. For example, for 5 Mbps and a buffer size of 156 kB, the frequency of the IPCs which is due to switching buffers is 4 per second. Increasing the buffer size decreases the frequency of the IPCs, but increases memory costs. This phenomenon can be observed in Fig. 6, where we can see 200 microsecond periodic peaks. This Figure is a plot of PDU interarrival times for a 2.5 Mbps CBR stream.

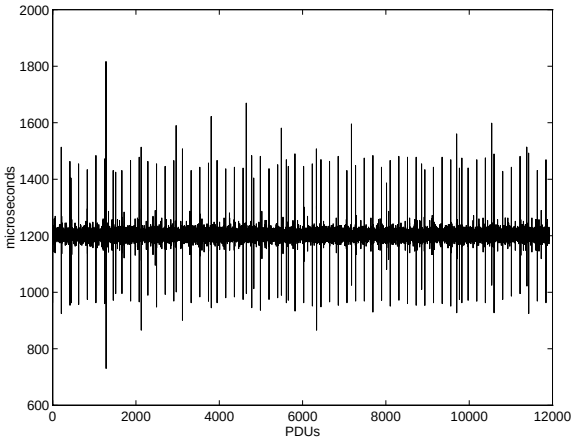

Fig. 6. PDU interarrival times for 2.5 Mbps video stream measured at the output of the video server.

A.2 Testbed Network

The transport network infrastructure of the Columbia VoD testbed includes ATM and Internet as the core technologies. The ATM network consists of a ATM Local Area Network (LAN), which is interconnected to the campus-wide ATM network. This campus ATM network is also connected to an ATM Wide Area Network (WAN), which provides high-speed connections to other VoD testbeds on the East Coast.

For the ATM case we are using **xbind** (Open Signalling Architecture) to control and manage the connections with QoS guarantees and efficient use of resources [15]. **xbind** is based on the schedulable region concept described in Section II-B. Users on the campus access the video servers directly through the native ATM network (using AAL5) or via Internet (i.e., over Ethernet or wireless).

An important feature is how the user can search for and select a specific video stream and control the stream. For this purpose we have an application server that communicates with the client through a CORBA-based interface using DSM-CC [16], [17] User-to-User primitives.

A.3 Testbed Video Clients

We have three types of video clients. The first type uses digital set-top-boxes with direct ATM connections (DS3 lines). These clients can decode up to 15 Mbps MPEG-2 TS streams. The output of the decoders are connected to TV monitors, providing better

quality than regular VCRs and displaying video at 30 frames per second. The second type uses personal computers and workstations with our in-house software decoder. The software implementation provides a high degree of design flexibility but with some processing constraints, such as frame rate and resolution, since it has to demultiplex and decode video and audio components received through the network. Finally, we have mobile terminals, using a hardware decoder (*IBM Thinkpad 760CD* laptop) and mobile IP protocol.

B. Design constraints

The interaction between the video client and the video pump is quite significant. One technique for reducing video client cost is to minimize the buffer space in the network interface intended for jitter removal. However, this makes the video pump design much more complicated. The video pump must send smaller packets more often, in order to bound the jitter introduced in the video packets conforming such PDU. As the buffer space in the video client becomes smaller, it becomes more difficult for general purpose workstations to meet these requirements [18]. For instance, for a PDU size of 376 bytes and an MPEG-2 transport stream of 6 Mbps (500 microseconds interval), the network interface in the video server has to handle 2000 interrupts per second for each stream.

MPEG-2 transport streams are broken into transport packets which are 188 bytes long. Taking into account the previous considerations, the ATM Forum [1] has specified that, for MPEG-2 transport stream transmission over ATM using AAL-5, the PDU size will be $188N$ bytes ($N = 1, 2, \ldots$). An AAL-5 PDU is allowed 48 bytes of payload per ATM cell. However, the PDU also has a trailer of 8 bytes which also must fit into this payload. If the PDU payload does not use all of the ATM cell payload, the unused bytes are wasted. For optimal throughput and so that no bytes are wasted we solve the following equation: $188N + 8 = 48M$, where N is the number of transport packets and M is the number of ATM cells. Integer solutions to this equation occur at $N = 2 + 12i$, for $i \geq 0$. Therefore the minimum PDU size is for the case of $N = 2$ or 376 bytes per PDU. Some digital set-top-boxes only admit this small PDU size for memory cost constraints. This is the case for our digital set-top-box.

As we mentioned before, the use of a small PDU has an adverse effect on the performance of the video server. There is a certain amount of processing overhead each time a PDU 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. Fig. 7 is a graph of PDU size vs. coefficient of variation for the interarrival time (see Section IV-A.1) for a rate of 2.5 Mbps. As can be seen, the smaller PDUs have a noticeably larger variance. This is due to the increased number of requests sent to the device driver.

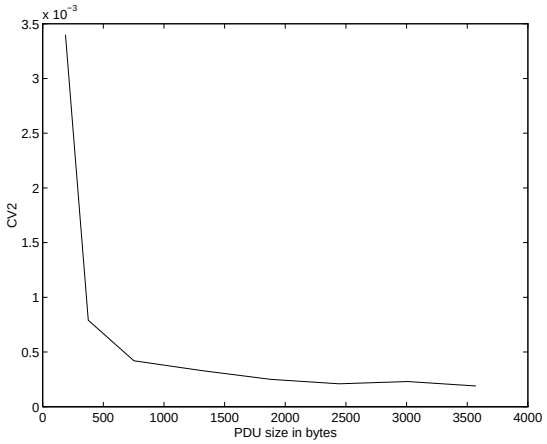

Fig. 7. CV^2 of PDU interarrival times vs. PDU size for a 2.5 Mbps video stream.

For this reason, the use of a small PDU size as mandated by our hardware video clients, in our video server is translated into a limit in scalability (i.e. maximum number of simultaneous video streams). This limitation is overcome, when the video client does not have such memory constraints. This is the case for our software video clients and any client that resides on a general purpose computer.

IV. Selection of Video Server QoS Parameters

In this section, we first define the QoS metrics of the video server. We then apply these to several experimental results obtained from our testbed in order to determine which metrics most accurately characterize a video server.

A. Video Server QoS Parameters

The most important feature in evaluating the performance of a video server is to study the output traffic pattern. The deviation from the theoretical video traffic pattern will

indicate the QoS degradation. For this reason, we are interested in studying the delay distribution of the PDU interarrival process associated with a video stream. There are many measures for evaluating performance and jitter. The performance metrics used in this paper refer to the transport of PDUs, since it is the information unit that the video client handles. The PDU metrics can be directly translated to cell metrics when the video server's network interface sends equispaced cells over a PDU period. Unfortunately this is not always the case in commercial ATM network adapters. We divide the PDU metrics into the ones associated with delay distribution moments and others associated with traffic control.

A.1 Delay Distribution Moments

From the PDU interarrival distribution, we can calculate three basic parameters associated with the first three moments of the distribution: mean, variance and skewness. The simplest measures are the mean, μ, and the variance, σ^2, which together form the coefficient of variation, $\mathrm{CV}^2 = \sigma^2/\mu^2$. The mean is the most basic measure for a CBR video server as it indicates whether it is pumping at the correct rate. The variance measures how consistently the server is pumping this rate, while the CV^2 provides similar information about the burstiness of the traffic. These measures are based on the first and second moments; for the third moment we use the skewness defined as $\nu = E[(X-\mu)^3]/\sigma^3$, which is a measure of symmetry of the delay distribution and indicates the uniformity of the delay variation.

A.2 Traffic Control Metrics

A second group of metrics is related to traffic control. In this case, we are more interested in studying the compliance of the video server output with a network contract (i.e. peak rate, burstiness, etc.). Examples of useful measures are the 1-Point PDU Delay Variation (PDV) and the Generic Cell Rate Algorithm (GCRA) which is equivalent to the continuous state leaky bucket algorithm [12]. The PDV for PDU k, d_k, is defined as

$$d_k = c_k - a_k \tag{1}$$

where c_k is the PDU's reference arrival time and a_k is the actual arrival time. The reference arrival time is defined as follows

$$c_{k+1} = \begin{cases} c_k + T & \text{if } c_k \geq a_k \quad \text{(early arrival)} \\ a_k + T & \text{otherwise} \quad \text{(late arrival)} \end{cases} \tag{2}$$

where T is the period which corresponds to the desired rate. The above method tries to eliminate the effects of PDU gaps and provides a measurement solely of PDU clumping, since a clumping in the PDU traffic pattern implies a violation of the peak rate specified in the network contract.

As a measure for video services, the PDV is problematic. It is very sensitive to the average rate. For instance, if the video pump is delivering a 5 Mbps video stream, using a PDU size of 376 bytes, with a period that is slightly smaller than the nominal period T by $\Delta T = 1$ microsecond in the period, the PDV could accumulate to one second over a period of less than 10 minutes:

$$\begin{aligned} a_0 &= c_0 \\ a_i &= a_{i-1} + (T - \Delta T); \quad \forall i \geq 1 \\ c_i &= c_{i-1} + T \qquad\qquad \forall i \geq 1 \\ d_i &= d_{i-1} + \Delta T = i\Delta T \quad \forall i \geq 1 \end{aligned} \tag{3}$$

However, this is not the case if the rate is slightly lower. In that case $d_i = -\Delta T$, for all values of $i \geq 1$. In practical systems, a slightly different rate could be compensated for by the video client PLL and therefore should not cause such dramatic effect in performance metrics.

The GCRA(T,τ) defines a leaky bucket running at a rate of $1/T$ with a tolerance of $100\tau/T$ and is a second-order statistic that measures the burst tolerance. If the tolerance is 0%, we only admit the PDUs that are not violating the network contract, i.e. PDUs whose interarrival time is greater than or equal to the nominal period T. By increasing the value of τ we can admit more bursty PDU traffic patterns. A less bursty source will have more PDUs admitted since it requires less tolerance in the leaky bucket to accomodate all the PDUs. It is important to note that two video streams with the same average bit rate can have totally different GCRA performance depending on how their PDUs are distributed over time.

The performance of a software-based video server can also be affected by other processes that are running on the workstation. Most of the resources such as the CPU, memory, disk bandwidth, and network bandwidth can be isolated using operating system level or user level mechanisms. Processes which require a substantial amount of bandwidth from the bus can cause erratic video pump performance.

B. Experimental Results

In this section, we apply the metrics defined in Section IV-A to experimental results obtained from the Columbia VoD testbed in order to identify which metrics are more effective in characterizing a video server. A video server must be able to support clients with largely varying bandwidth requirements. For this reason, we test our video server over a broad range of bit rates, 800 kbps (for software decoders), 2.5 Mbps (VCR quality), 5 Mbps (digital TV quality), and 10 Mbps (HQ-TV). In all cases, our video streams are MPEG-2 transport streams encoded from the movie *Robin Hood*. Each stream has a minimum duration of 15 seconds, or equivalently 50,000 PDUs for 10 Mbps (376 byte PDUs). All measurements of the video server performance are made using an HP Broadband Analyzer, which is connected to the ATM switch. We begin testing the video server for a single CBR video stream. Afterwards we test the video server under different cross-traffic scenarios in order to model the effect of contention for resources as mentioned in Section II-A. Finally we identify metrics defined in Section IV-A that best characterize the video server performance.

B.1 Performance of the Delay Distribution Moments

For each of the rates (800 kbps, 2.5 Mbps, 5 Mbps and 10 Mbps) the video server maintains the average rate of a single CBR video stream to within 0.006% of the desired rate, with 10 Mbps being this worst case. The standard deviation is relatively constant over the different rates too, ranging between 30 and 50 microseconds (Table I).

Fig. 8 shows the probability mass function for a 2.5 Mbps video source, where the side lobes correspond to the effect of the delay introduced by IPC. The right lobe is due to the PDU which is delayed by IPC and the left lobe to an early-arriving PDU, which is an attempt by the absolute timer to compensate for the previous delay. These simple

TABLE I

First, second and third moments for a single CBR video source.

CBR video (kbps)	μ (kbps)	σ (μs)	CV^2	ν
800	800.00	51.77	0.0001	0.3780
2500	2499.99	31.99	0.0007	1.1499
5000	5000.08	30.18	0.0025	1.5235
10000	9999.37	30.84	0.0105	5.2072

measures do not provide enough information in this case, because, as will been seen, the video pump actually performs more consistently at lower rates and this cannot be seen from the mean, standard deviation or CV^2. The timer has a larger impact on streams with higher rates than on streams with lower rate. This is made obvious by the third moment statistic, skewness, which increases with the rate (Table I). The skewness indicates that the probability mass function becomes less symmetric and heavier on the side of long delay.

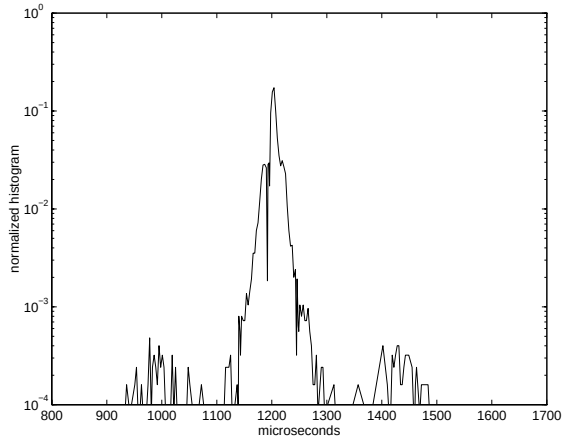

Fig. 8. Probability Mass Function for PDU interarrival times of a 2.5 Mbps video stream.

B.2 Traffic Control Performance

Fig. 9 shows the PDV evolution for the times between arriving PDUs for a 2.5 Mbps stream. The staircase shape is caused by the way the reference and arrival time is defined

in (2). Due to the use of absolute timers, when one PDU is delayed by Δt, the next one will be sent with a period which is decreased by Δt. This causes a PDU delay, followed by a PDU clumping. When the large delay is incurred, the PDV goes negative and the subsequent reference arrival time, c_{k+1}, is updated using the actual arrival time, a_k. If d_{k-1} is the PDV just before the delay and d_k is negative, due to the delay, then the PDV for the clumping is, $d_{k+1} = d_{k-1} + |d_k|$.

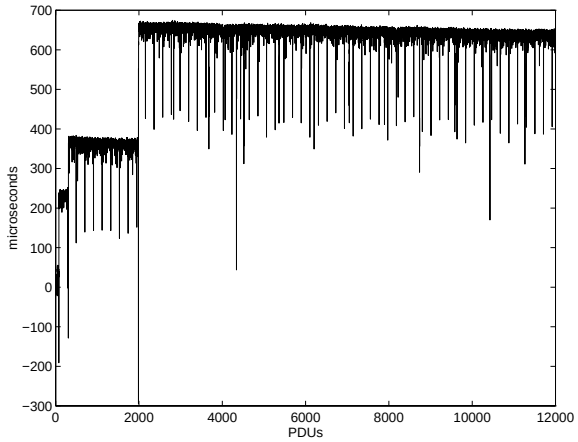

Fig. 9. PDU Delay Variation for PDU interarrival times of a 2.5 Mbps video stream.

As the rate increases, burstiness increases too. This can be seen in Fig. 10 which is a graph of the GCRA curves for each of the rates. Clearly, the lower bit rates have a much higher percentage of admitted cells than the higher rates. This is because the higher rates are burstier and the PDUs are often more clumped together.

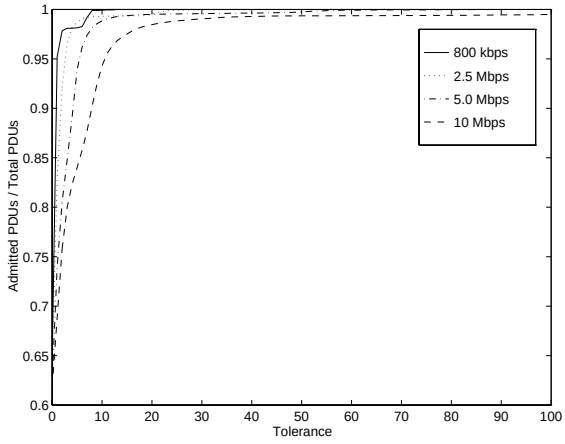

Fig. 10. GCRA for varying CBR rates.

Since the video pump buffers are of fixed size, regardless of rate, the number of IPCs between PDUs increases with decreasing rate. For the 800 kbps source, we observe a plateau that is caused by the increased frequency of the IPC at lower rates. In this case, more tolerance is needed to absorb this jitter.

B.3 Cross-Traffic Scenario

As mentioned earlier, we model the video server as a virtual multiplexer. The competition for common resources among video pumps can be modeled as cross-traffic and the origin of this cross-traffic can be considered as the aggregate traffic of other video sources. Table II gives the values of the CV^2 and the average PDV for a CBR 800 kbps video source and for three types of cross-traffic: deterministic, exponential and hyperexponential.

TABLE II

QoS PARAMETERS FOR AN 800 KBPS STREAM UNDER DIFFERENT TYPES OF CROSS-TRAFFIC.

Type of Cross-traffic	CV^2	1-point PDV μs
Deterministic	0.003189	1018
Poisson	0.033041	3790
Hyperexponential	0.108488	8358

The average bit rate of this cross-traffic is also 800 kbps and the latter two were generated using a gamma distribution with CV^2_{cross} values of 1 and 3, respectively. We observe that the jitter introduced in the video source is greater when the background traffic has a higher degree of burstiness, with a severe degradation when the background traffic is hyperexponential.

The average rate of the cross-traffic also has an impact on the QoS of the video stream. Fig. 11 shows the GCRA for a target CBR 800 kbps video source when it is multiplexed with cross-traffic of average rates of 800, 2500 and 5000 kbps and exponential distribution. For a higher cross-traffic average bit rate, we see a better performance for the jitter. This result is consistent with the analysis of M+D/D/1 [19] and simulation of video sources in a multiplexer [20] where the higher bit sources always experience more jitter, since the probability of contention with other sources is higher. On the contrary, for a fixed

average rate cross-traffic, the higher the target CBR video source is, the higher the jitter is. Table III shows the CV^2 for these two possible scenarios.

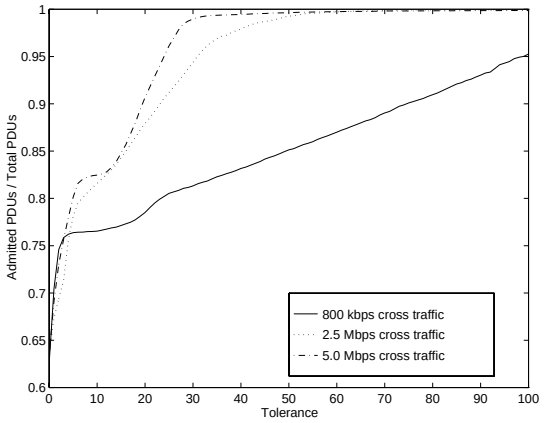

Fig. 11. GCRA for a CBR 2.5 Mbps with varying rate exponential cross-traffic.

TABLE III

CV^2 FOR A CBR TARGET VIDEO SOURCE UNDER EXPONENTIAL CROSS-TRAFFIC.

Exponential Cross-Traffic (kbps)	CBR-Video (kbps)	CV^2
800	800	0.033041
800	2500	0.242368
800	5000	0.576097
800	2500	0.242368
2500	2500	0.025269
5000	2500	0.018787

From the results shown in the previous sections we can conclude that the delay distribution moments measures such as the mean, variance, coefficient of variation and skewness, do not give enough information about the performance of the video server. On the other hand, the 1-point PDV is too sensitive to the momentary fluctuations of the video streams as well as the average rate. It also does not provide a stationary measure of the perfor-

mance of the video server over a long period of time and therefore does not accurately characterize the performance of a video client. We can see that a single clumping around PDU 2000 (Fig. 6) is translated into a jump in the PDV measure for the following PDUs (see Fig. 9). The GCRA captures the second-order statistics of the video stream or how the PDUs are distributed in bursts over time. It provides a measure of the behavior from an ideal video server over the connection time. For these reasons, the GCRA is the most informative measurement for end-to-end QoS at the video client and a good indicator of the video server performance.

V. QoS Mapping from Video Server to Video Client

Now that we have identified the GCRA as the most comprehensive QoS parameter for characterizing the video server performance, we map the GCRA at the video server into an equivalent QoS parameter at the video client. As mentioned earlier, this task is not trivial because of the different scales handled, the varying effect of errors on the information in the corrupted PDU, and the subjective criteria of the end user. One of the most significant parameters for the video client is PDU loss. For this reason we use the network interface buffer overflow as an indication of PDU loss and therefore QoS degradation. Although this parameter alone cannot fully characterize the QoS at the video client, it is certainly a significant indicator of QoS degradation.

Subjective tests have been performed which indicate that a probability of PDU loss for MPEG-2 decoders on the order of 10^{-3} is acceptable [21]. These test have been performed using digital set-top-boxes with ATM connections, where the set-top-box has no error concealment capabilities. We have reproduced those tests by injecting jitter, using a Network Impairment Emulator in our Broadband Analyzer, into the traffic pattern and observing the results on our digital set-top-box video clients. The frame loss and blocky effect is observed to be tolerant to an expert user at the PDUs loss rate of 10^{-3}. Fig. 12 shows two frames from the MPEG-2 transport stream *Robin Hood* encoded at 2.5 Mbps. The left frame corresponds to the case where the PDU loss is $P_n < 10^{-3}$ ($P_n \simeq 10^{-4}$), whereas the right frame corresponds to the case where the PDU loss is $P_n > 10^{-3}$ ($P_n \simeq 10^{-3}$). Clearly, the right frame shows a significant decrease in the subjective QoS. It is important to note that the effect of PDU loss depends on the the

encoding technique (MPEG-2) as well as the error concealment capabilities of the decoder.

(a)

(b)

Fig. 12. Jitter effect on the subjective QoS. (a) Robin Hood frame with a PDU loss, $P_n < 10^{-3}$. (b) Robin Hood frame with a PDU, $P_n > 10^{-3}$.

For a given PDU loss rate, the higher the rate of the video stream, the higher the absolute number of PDU are lost per second. However the most critical case is when losses occur in the header information (i.e. sequence, group of pictures). The occurrence of these PDU losses is practically independent of the rate of the video stream since the number of PDUs containing this header information is the same regardless of the rate. We have empirically verified this fact by confirming that the subjective QoS is similar for different rates and a constant PDU loss.

Now that we have identified a relationship between the subjective QoS and a PDU loss

rate at the video client, we map the GCRA performance at the video server with the PDU loss probability (i.e., overflow in the network interface) at the video client to make the connection from video server performance to video client subjective performance.

The GCRA(T, τ) algorithm is equivalent to the continuous-state leaky bucket algorithm [12]. It can be modeled as a finite capacity buffer of size $T + \tau$. Every time a PDU is conforming, the buffer is incremented by T units of content and the buffer is continuously drained at a rate 1 unit of content per unit of time. A PDU is admitted on arrival if and only if there is enough room in the buffer to accommodate it.

The primary difference between the network interface and GCRA(T, τ) is the granularity, because the network interface is dimensioned in units of PDUs. Every time a PDU arrives it is served with a service time equal to the ideal interarrival time. We are assuming that the service time of the client's network interface is driven by the rate of the video stream. Under this condition, if P_n is the probability of PDU overflow in the network interface with a memory of n PDUs, we can establish the following relationship:

$$\text{GCRA}(T, nT) = 1 - P_n, \;\; n = \; 0,1,2..., \tag{4}$$

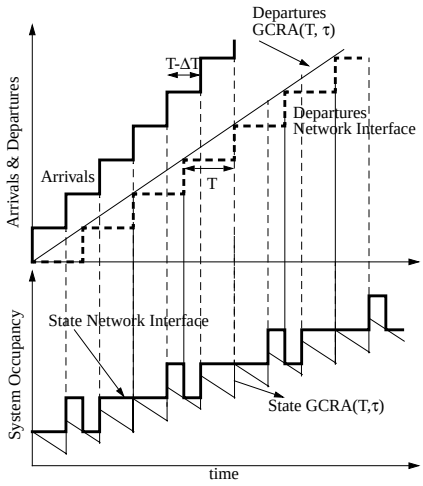

Fig. 13. Relationship of GCRA(T, τ) with video client network interface buffer occupancy.

This relationship can be justified by Fig. 13, where a source with a slightly higher bit rate than the nominal is analyzed with both the GCRA(T,τ) and the network interface state. We notice that the GCRA(T,τ) is always less than the network interface occupancy and is equal for each $\tau = nT$. Therefore, for the intermediate values of τ, we have the following relationship that is based on the monotonicity of the function GCRA(T,τ):

$$1 - P_n \quad \leq \text{GCRA}(T, T(n+\alpha)) \quad \leq 1 - P_{n+1}$$
$$, n = 0, 1, 2 \ldots; 0 \leq \alpha \leq 1. \tag{5}$$

From the previous analysis, a network interface with a capacity of 1 PDU will have a tolerance of 100%. This tolerance is related to the maximum burst size in any time interval of size T. For a $\tau = nT$, the maximum burst the system can tolerate in any interval T will be equal to $n + 1$ PDUs. However this burst size is a maximum; two consecutive smaller bursts could lead to overflow as well, since the GCRA(T,τ) is a second order statistic with memory.

In the next section, the relationship established in (5) will be used to define a metric that jointly captures the QoS constraints both of video server and video client.

VI. Schedulable Region

We now make use of our previous results in mapping the video server QoS into the video client QoS by defining a schedulable region. This region indicates the set of admissible combinations of traffic types which will guarantee a certain level of QoS for each individual stream. It is important to make the distinction between guaranteeing QoS for the aggregate traffic and guaranteeing QoS for each stream. A guarantee for the aggregate traffic only provides an average QoS for the set of video streams. The QoS for a particular stream could likely degrade below the guarantee for the aggregate traffic. Our definition of QoS is on a per-stream basis and specifies a minimum QoS that a stream will encounter.

A. Definition of Schedulable Region

Let us assume I is the number of possible types of video streams at the video server. Each type of video stream is characterized by the pair (R_i, ϵ_i). The first component, R_i, is the average bit rate for video streams of Type i, with $i = 1, \ldots, I$. The second component, ϵ_i, is the acceptable PDU loss probability at the video client for video streams of Type i. Let us also define $v_{i,j}$ as the jth video stream of Type i that is simultaneously pumped. The schedulable region, $S \in \mathbf{N}^I$, is defined as the set of all possible compliant combinations of simultaneously pumped video streams. We say that a combination of

video streams, $\{v_{1,1}, v_{1,2}, \ldots, v_{1,\Phi_1}; v_{2,1}, \ldots, v_{2,\Phi_2}; \ldots; v_{I,1}, \ldots, v_{I,\Phi_I}\}$, expressed as the $\mathbf{N}^I$ vector $\mathbf{\Phi} = (\Phi_1, \Phi_2, \ldots, \Phi_I)$, where Φ_i is the number of pumped video streams of Type i, is compliant and belongs to S if it conforms to the following conditions:

$$\mathbf{\Phi} \in S \Longleftrightarrow \begin{cases} \sum_{i=1}^{I} \Phi_i \leq C \\ r_{i,j} = R_i \\ P_{n,i,j} < \epsilon_i \\ \forall i,j \qquad\qquad 1 \leq i \leq I, 1 \leq j \leq \Phi_i \end{cases} \tag{6}$$

where C is the number of processors allocated to the video server, $P_{n,i,j}$ is the probability of PDU loss at the video client for the jth video stream of Type i with buffer space for n PDUs in the network interface. The first constraint in (6) implies that the video server can only serve a maximum number of video streams according to its processing capacity. In our software architecture it is possible to run several video pump objects on a single processor. In this case, the operating system scheduler acts as the multiplexer. The scheduler task-switches the video pumps into and out of the CPU so that only one pump is active at a time and each pump receives a slotted period of time to operate. This results in the scheduler performing as a TDMA multiplexer. Our interest here is to demonstrate the multiplexing performance of the video pump and its interaction with the network interface. Therefore, we restrict our attention to video pump objects running on separate processors. The second constraint in (6) enforces that video streams are being pumped at the right rate to avoid severe underflow situations at the video client (i.e. $r_{i,j} < R_i$). Finally the third condition ensures a compliance with a network contract, such as GCRA, which is mapped into a video client QoS parameter, $P_{n,i,j} < \epsilon_i$, as was seen in (5).

B. Methodology and Example of Schedulable Region

In order to calculate the schedulable region, S, for a specific video server, we use the following methodology. First, we define how many types of video streams to consider. There is a trade-off between the unlimited number of service classes and the reduced number of traffic classes that the network can handle. For a given processing capacity of the video server, C, and a given number of types of video streams, I, there are C^I possible vectors or video streams combinations. Only a subset of these vectors are compliant with

conditions on S in (6). Each video stream from a particular vector is individually traced using a Line Interface (LIF) from the broadband analyzer. The captured trace is analyzed using a Traffic Monitor, which checks the average bit rate, $r_{i,j}$, and the GCRA. Only if all the video streams in a vector are compliant with the conditions in (6), can that vector be contained in S. We have to note, that it is not necessary to check all the C^I combinations. For instance if $(\Phi_1, \Phi_2, \ldots, \Phi_I) \notin S$, it will imply that $(\Phi'_1, \Phi'_2, \ldots, \Phi'_I) \notin S$ for all $\Phi_i \leq \Phi'_i \leq C$. For this reason, choosing the initial vectors carefully can save many calculations for the schedulable region, S.

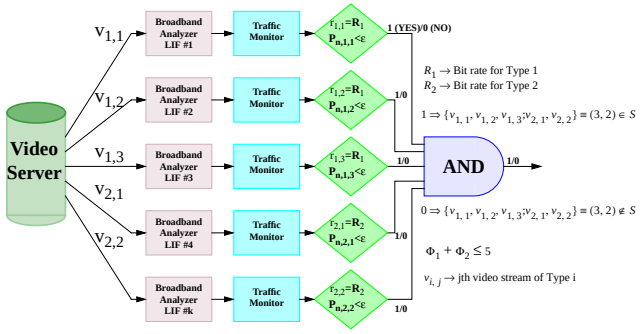

Fig. 14. Procedure to obtain the schedulable region for two type of traffic classes at the video server. Example for the video stream combination $\{v_{1,1}, v_{1,2}, v_{1,3}; v_{2,1}, v_{2,2}\} \equiv (3, 2)$.

Following this methodology, we calculate S for the Columbia video server. In this example, we have chosen two traffic classes, Type 1, with $R_1 = 800$ kbps and Type 2, with $R_2 = 2.5$ Mbps. The QoS constraint is $\epsilon = \epsilon_1 = \epsilon_2 = 10^{-3}$, according to the discussion in Section V. The processing capacity in our server is $C = 5$ (ONYX 6-processor system, resticting one video pump per processor). Fig. 14 shows how the video stream combination $\{v_{1,1}, v_{1,2}, v_{1,3}; v_{2,1}, v_{2,2}\}$ or vector $(3, 2)$ is checked for its inclusion in S. We can trace all the video streams simultaneously or in subsequent measurements depending on the number LIF available in the broadband analyzer, since it is possible to reproduce the same scenario in the video server. Fig. 15 shows several schedulable regions for these two types of video streams. Each region corresponds to different PDU buffer sizes, n, in the network interface. In this case we use a PDU size of 376 bytes. The higher the buffer is, the higher GCRA tolerance is and more video streams combinations are compliant with

(6). We also observe that increasing the PDU size above a certain value does not enlarge S. At that value, all the potential jitter has been absorbed by the network interface and the limitation is imposed by the video server with the second condition in (6).

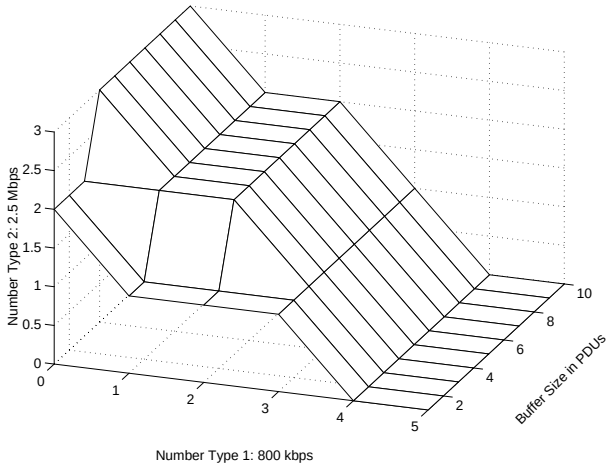

Fig. 15. Schedulable region for Type 1 and Type 2 traffic vs. video client network interface buffer size for $P_n < 10^{-3}$ in the video client.

Note that the calculation of S is an off-line procedure and must only be performed once for each video server-video client combination. We know beforehand the characteristics of the video streams stored in the video server. Therefore we can have as much accuracy (i.e., different types of video streams) as we desire. It is convenient however to have a similar number of service classes for the video server as traffic classes for the network. In this way, the admission control can be performed in a more straightforward manner, since video server and network are using similar metrics. Moreover, a commercial VoD system will likely have just a few traffic classes, actually maybe only one (i.e., all the streams encoded at a nominal rate).

Although admission control procedures are outside the scope of this paper, we describe two different scenarios where S can be applied. The first scenario is when the video server and the video client are connected to the same ATM LAN. In this case, the main source of QoS degradation is the video server since the low utilization of such networks prevents the switches from building up significant congestion. Therefore, a direct video server to video client mapping can be applied considering a transparent network which is not introducing QoS degradation. We have empirically verified this fact in the Columbia VoD testbed.

We have simultaneously traced the traffic pattern at the output of the video server and at the input of the video client, and no significant differences (i.e., GCRA performance) have been detected, for even a moderately loaded ATM switch.

The second scenario is when the video server and the video client are interconnected through a virtual path in a ATM WAN. In this case, the switches along the path introduce QoS degradation, since the utilization of the network is higher than in the ATM LAN case. Let us assume that the virtual path consists of an interconnection of L switches. We can associate with each switch a schedulable region, S^l, with $1 \leq l \leq L$. As long as the traffic classes and QoS constraints considered in each switch were the same ones as in the schedulable region of the video server, S^0, we can assume an equivalent schedulable region, $S^{eq} = \min_{0 \leq l \leq L} \{S^l\}$ for admission control purposes. The equivalent QoS constraint, which is the one to map in the video client, will be $\epsilon^{eq} = \sum_{l=0}^{L} \epsilon^l$, which is a conservative bound. More sophisticated admission control procedures can be applied to have a more efficient use of resources (i.e. schedulable region with more than one QoS constraint per type of video stream). The use of one technique or another is a trade-off between the cost of bandwidth in the network and the cost of processing in the network nodes.

VII. Conclusions

We have presented a novel and simple approach for mapping end-to-end QoS in VoD services. Using this approach, we have derived a schedulable region for a video server which guarantees end-to-end QoS, where a specific QoS required in the video client translates into a QoS specification for the video server.

We have used a methodology that is based on a generic VoD model. This model consists of three basic components: video server, network and video client. The model is an abstraction and captures the common elements that can be found in any VoD system, avoiding specific elements that are particular to a specific implementation or platform. The video server model allows us to consider the video server as the first switch of the connection. This has the advantage of simplifying admission control, since the behavior of the video server can be characterized as an additional switch of the connection with a specified QoS performance. This approach can be applied to both software-based and hardware-based video servers.

The Columbia VoD testbed is used as a practical implementation example of how our VoD model can be applied as well as how design constraints can affect the overall QoS performance. We have seen that the choice of a small PDU size for memory constraints in the video client has a detrimental impact on the scalability of the video server. These kinds of experimental results from the Columbia VoD testbed are the first ones from a VoD system presented in technical literature.

We have defined the QoS metrics of the video server, and have applied them to several experimental results obtained from the Columbia video server. From this analysis, we have concluded that the moments from the delay distribution do not give enough information about the performance of the video server. On the other hand, a traffic control measure like the PDV does not provide a stationary measure of the performance of the video server over a long period of time. The GCRA, as a measure of how PDUs are distributed in bursts over time, provides the most complete characterization of the performance of a video server.

Once we have identified a comprehensive QoS parameter for the video server, we then mapped the GCRA at the video server into an equivalent QoS parameter at the video client. We have established a relationship between the subjective QoS and the PDU loss rate at the video client. Then, we linked the GCRA performance at the video server with the PDU loss probability (i.e., overflow in the network interface) at the video client. From this relationship we have defined the concept of a schedulable region for a video server which models the capacity of the video server under video client QoS constraints. Our model can be connected to network QoS admission control models, so that we have a unified approach for admission control.

Other issues such as more sophisticated QoS parameters that take into account subjective and objective criteria are currently being explored in the VoD Testbed Group at Columbia University.

Acknowledgments

The authors would like to thank the members of the Columbia VoD Testbed Group for their contribution and suggestions. We would also like to thank the COMET group for the use of their Broadband Analyzer, HP-IDACOM for providing us with the MPEG-2

support software and the Network Impairment Emulator board for the Analyzer, C-Cube Microsystems for providing us with the VideoRISC MPEG-2 Encoder, Philips for providing us the MPEG-2 and GTE Labs for their collaboration in the interoperability testing.

This work was supported in part by the Army Research Office under grant DAAH04-95-1-0188, and sponsors of the Columbia ADVENT project.

References

[1] The ATM Forum, Technical Commitee, *Audiovisual Multimedia Services: Video on Demand. Specification 1.0*, Jan. 1996, af-saa-0049.

[2] Digital Audio Visual Council, *DAVIC 1.0 Specification*, Dec. 1995.

[3] Y.-H. Chang, D. Coggins, D. Pitt, D. Skellern, M. Thapar, and C. Venkatraman, "An open-systems approach to video on demand," *IEEE Communications Magazine*, vol. 32, no. 5, pp. 68–80, May 1994.

[4] D. Deloddere, W. Verbiest, and H. Verhille, "Interactive video on demand," *IEEE Communications Magazine*, vol. 32, no. 5, pp. 82–88, May 1994.

[5] J. R. Jones, "Baseband and passband transport systems for interactive video services," *IEEE Communications Magazine*, vol. 32, no. 5, pp. 90–101, May 1994.

[6] W. Y. Chen and D. L. Waring, "Applicability of ADSL to support video dial tone in the copper loop," *IEEE Communications Magazine*, vol. 32, no. 5, pp. 102–109, May 1994.

[7] M. de Prycker, *Aynchronous Transfer Mode. Solution for Broadband ISDN*, Ellis Horwood Series in Computer Commmunications and Networking. Ellis Horwood Limited, New York, NY, second edition, 1993.

[8] S.-F. Chang, A. Eletheriadis, and D. Anastassiou, "Development of Columbia's video on demand testbed," *Image Communication Journal*, vol. 8, no. 3, pp. 191–207, Apr. 1996, Special Issue on Video on Demand.

[9] J. M. Hyman, A. A. Lazar, and G. Pacifici, "Real-time scheduling with quality of service constraints," *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 7, pp. 1052–1063, Sept. 1991.

[10] A. A. Lazar, A. Temple, and R. Gidron, "An architecture for networks that guarantees quality of service," *International Journal of Digital and Analog Communications Systems*, vol. 3, no. 2, pp. 229–238, Apr.–June 1990.

[11] A. A. Lazar and G. Pacifici, "Control of resources in broadband networks with quality of services guarantees," *IEEE Communications Magazine*, vol. 29, no. 10, pp. 66–73, Oct. 1991.

[12] The ATM Forum, *User-Network Interface (UNI) Specification. Version 3.1*, Sept. 1994.

[13] J. Zamora, "Issues of videoservices over ATM," Tech. Rep. 405-95-11, Center for Telecommunications Research, Columbia University, New York, NY, May 1995.

[14] ISO/IEC IS 13818-1 and ITU-T Recommendation H.222.0, *Information Technology - Generic Coding of Moving Pictures and Associated Audio - Part 1: Systems*, June 1994.

[15] A. A. Lazar, K.-S. Lim, and F. Marconcini, "**xbind** version 1.0: A CORBA based binding architecture," Tech. Rep. 412-95-18, Center for Telecommunications Research, Columbia University, New York, NY, June 1995.

[16] Object Management Group, *The Common Object Request Broker: Architecture and Specification. Revision 1.2*, Dec. 1993.

[17] ISO/IEC DIS 13818-6, *Information Technology - Generic Coding of Meoving Pictures and Associated Audio - Part 6: MPEG-2 Digital Storage Media - Command and Control (DSM–CC)*, June 1994.

[18] S. Jacobs and A. Eleftheriadis, "Video pump design for interoperability with set top units: The case against small pdus," in *Proceedings, 6th International Workshop on Network and Operating System Support for Digital Audio and Video*, Tokyo, Japan, Apr. 1996.

[19] J. W. Roberts and F. Guillemin, "Jitter in ATM networks and its impact on peak-rate enforcement," *Performance Evaluation*, vol. 16, pp. 35–48, 1992.

[20] J. Zamora, D. Anastassiou, and K. Ly, "Cell delay variation performance of cbr and vbr mpeg-2 sources in an atm multiplexer," in *Proceedings VIII European Signal Processing Conference*, Trieste, Italy, Sept. 1996.

[21] M.-T. Sun, J. Zdepski, and D. Raychaudhuri, "Error concealment for mpeg video over atm," Tech. Rep. MPEG92 AVC-308, CCITT ISO-IEC/JTC1/SC2/WG1, July 1992.