On the Impact of Policing and Rate Guarantees in Diff-Serv Networks: A Video Streaming Application Perspective

Wael Ashmawi*
Intel Corp.
3600 Juliette Lane
Santa Clara, CA 95052
wael.a.ashmawi@intel.com

Roch Guerin**
U. of Pennsylvania
200 South 33rd Street
Philadelphia, PA 19104
guerin@ee.upenn.edu

Stephen Wolf, Margaret Pinson
Inst. Telecommun. Sciences
325 Broadway
Boulder, CO 80305-3328
(steve,mpinson)@its.bldrdoc.gov

ABSTRACT

Over the past few years, there have been a number of proposals aimed at introducing different levels of service in the Internet. One of the more recent proposals is the Differentiated Services (Diff-Serv) architecture, and in this paper we explore how the policing actions and associated rate guarantees provided by the Expedited Forwarding (EF) translate into perceived benefits for applications that are the presumed users of such enhancements. Specifically, we focus on video streaming applications that arguably have relatively strong service quality requirements, and which should, therefore, stand to benefit from the availability of some form of enhanced service. Our goal is to gain a better understanding of the relation that exists between application level quality measures and the selection of the network level parameters that govern the delivery of the guarantees that an EF based service would provide. Our investigation, which is experimental in nature, relies on a number of standard streaming video servers and clients that have been modified and instrumented to allow quantification of the perceived quality of the received video stream. Quality assessments are performed using a Video Quality Measurement tool based on the ANSI objective quality standard. Measurements were made over both a local Diff-Serv testbed and across the QBone, a QoS enabled segment of the Internet2 infrastructure. The paper reports and analyzes the results of those measurements.

1. Introduction

As IP networks are being used to carry an ever-increasing range of traffic types, it has been argued that the associated diversity of service requirements calls for the introduction of service differentiation in the network. A recent proposal in that direction is embodied in the Differentiated Services (Diff-Serv) architecture [1] that supports service differentiation by specifying a small number of different "behaviors" [12],[16] for the network when forwarding packets. Our general goal in this paper is to gain a better understanding of the relation that exists between the better network guarantees that such new capabilities are providing and the *actual* benefits that ensue as perceived by the applications that use them. Because this is obviously a broad topic with possibly as many different answers as there are applications, we narrow our investigation to a specific application, namely, streaming video. Video applications are gaining in popularity and have seen significant evolution over the past five years. Early video transmissions over IP networks were mostly limited to *content delivery* that allowed users to download compressed videos in order to play them back locally. The disadvantage of having to wait for the download to complete before being able to start playing back a video as well as the desire to avoid storage, and ultimately duplication, of local copies, quickly led to the development of *progressive download* technologies that allow clients to watch the video as the content files are being downloaded. Several currently available video compression schemes were modified to enable progressive downloads: QuickTime™, VIVO™, VIDO™, and RealNetworks™ are examples of such schemes[♣].

One of the main challenges for these new technologies is to develop adaptive schemes capable of dealing with the uncertainty of network resource availability between clients and servers. This has led to a number of approaches aimed at "dynamically" adjusting the rate of video delivery based on estimating the resources available between the server and the client. The more basic schemes only allow users to select from several possible video qualities at the connection setup time, based on an initial estimate of the available bandwidth between the server and the client. Some of the more sophisticated and recent schemes, e.g., [23], include the ability to increase or decrease quality in real-time as available resources vary. These advances not withstanding, the transmission of video over the Internet has remained somewhat hampered by the relative unpredictability of its quality, in particular when it comes to the delivery of high bit rate video streams. As a result, video streaming has been identified by many as an application that stands to benefit most from the introduction of *Differentiated Services*.

Our goal in this paper is not to argue for or against such a position. Instead, as stated earlier, we are interested in understanding how the *network level* mechanisms of Diff-Serv, and in particular their configuration, translate into better *video* quality for the users viewing them. Specifically, we want to "plot" the evolution of video quality perceived by a user, as a function of

❑

[*] The work of this author was done while at the University of Pennsylvania and was supported by NSF Grant ANI99-06855.

[**] The work of this author was supported in part by NSF Grants ANI99-02943, ANI99-06855, and ANI00-85930.

the various network level parameters, e.g., token rate and burst size, associated with the support of different types of network services. Clearly, video quality will depend on a number of other factors, in particular the design of the server itself, which can heavily influence the outcome based on how well it takes advantage of the different services offered by the network. Our perspective is, however, not that of a video server designer. Instead, we want to take the position of a typical user, one who relies on existing client and server technology and who is trying to determine how best to connect the two across a Diff-Serv enabled network. In other words, the user has no or limited information about the internals of the video server and clients, and is primarily focused on selecting network parameters for its video connection in order to maximize video quality. Implicit here is the fact that there is a cost associated with network resources, so that the user has an incentive for understanding their relation to video quality, and achieving the best possible quality with the minimum amount of network resources.

Despite its relatively simple formulation and the several restrictions we impose, e.g., servers and clients as "black boxes," and Diff-Serv as the only network level mechanism we consider, achieving the stated goal of accurately evaluating the relation between network parameters and user level quality is not a simple task, as there are still many different aspects to consider and possible parameters to vary. A first and critical dimension is to measure video quality, *as users viewing the received video streams perceive it*. Clearly, network-level impairments such as packet losses or delay and the availability of resources such as bandwidth will play a role, and it is safe to say that video quality will improve as available bandwidth increases or as losses and delay decrease. However, this is a far cry from providing a precise and quantitative evaluation of the relation between them. On one hand, characterizing the relation that exists between network parameters such as token bucket parameters and network performance, e.g., losses or delay, is a relatively well investigated area when it comes to video, and a number of works have provided guidelines on how to select parameters, e.g., [20][21][24] and how to measure network performance [11][22]. On the other hand, determining how these translate into different levels of video quality has received little attention. As a result, one of this paper's contributions is to provide a quantitative, even if still partial, answer to this question and to illustrate the use of *objective video quality* measurement tools in doing so.

Another difficulty in accurately characterizing how "better" network services can improve user level perception of video quality is to perform such an evaluation in a reasonably *realistic* setting. Because our investigation is experimental in nature, the approach we took to address this issue was to consider a relatively broad range of configurations and scenarios. Specifically, we experimented with multiple types of video servers and clients, as well as several video clips and encoding schemes. In addition, several different network configurations were employed, ranging from a Diff-Serv capable local testbed to video transmissions across the QBone [25], a wide area Diff-Serv enabled segment of the Internet2 Abilene network.

In the rest of this paper, we describe the experimental setup used for our investigation and report our findings regarding the relation that exists between network level quality and the resulting video quality as perceived by users. The paper is structured as follows. In Section 2, we provide background information on Differentiated Services, video server technology, and video quality measurement tools. Because of constraints imposed by the testbeds over which we conducted our experiments, in particular the wide area testbed we had access to, our focus has mainly been on a service built on the Expedited Forwarding Per Hop Behavior (EF PHB). Section 3 is devoted to a description of the three main components of our experimental setup: the video quality measurement tool, the setup we have developed in order to capture the video information we feed to the quality measurement tool, and the different network testbed configurations. Results and their analyses that illustrate the relation and differences that exist between network level and user level quality measures are the topic of Section 4. Finally, Section 5 summarizes our findings and concludes the paper.

2. Background Material

2.1 The Differentiated Services Architecture

The Diff-Serv working group has put forth a number of specifications outlining not only the general architecture of the Diff-Serv approach [1] and the fields in the IP header available to support it [19], but also specifying two[1] initial *Per Hop Behaviors* (PHBs) that are to be used to provide differentiated forwarding treatments in Diff-Serv enabled networks. The basic premise of the Diff-Serv architecture is that complex flow level functions are performed at the network edge, and that core routers only support a few coarse classes or *Behavior Aggregates* (BAs). The Diff-Serv architecture, therefore, relies on two distinct components: flow classifiers and policers at the edges that are responsible for mapping incoming packets to the appropriate BA and marking them with the appropriate *Diff-Serv Code Point* (DSCP) [19], and various scheduling and buffer management mechanisms in the core that are responsible for implementing the different PHBs and mapping packets to PHBs based on their DSCP.

Of most significance in the context of our investigation are the *policing actions* performed at the network edge and the parameters they involve. Specifically, policers are used to impose limitations on the traffic entering the network marked with a given DSCP. These limitations are typically enforced through a token bucket, e.g., [13][14], that controls both the rate and the burstiness of the traffic. The token bucket parameters, i.e., token rate and token bucket depth, therefore play a major role in determining the level of service provided to a flow assigned to a given BA. In addition to the token bucket parameters, the policing actions applied to non-conformant packets, i.e., packets that arrive to find an empty token bucket[2], are also expected to have a major impact on performance. A complete investigation should, therefore, consider all possible combinations of these different factors to assess their relative effect on user level video quality. Fortunately, the current specification of the two PHBs that have been defined somewhat limits the range of combinations that need to be considered. Specifically, the *Expedited Forwarding* (EF) PHB [16] essentially requires a small burst size, i.e., a token bucket depth of one or at most two Maximum Transmission Units (MTUs), and a policer that either drops or shapes non-conformant packets. Conversely, while the *Assured Forwarding* (AF) PHB [12] group allows for a wider range of burst sizes, it primarily calls for

[1] Actually one PHB [16] and a PHB Group [12].

[2] Note that we assume here that tokens correspond to credits that are needed to transmit packets. This is consistent with the terminology of RFC 2212, 2697, and 2698.

policing actions that mark packets with different "colors" (DSCPs) depending on their level of non-conformance. As mentioned earlier, our focus is on the EF PHB for which the main variables are the token rate and token bucket depth assigned to the stream and whether the policer shapes or drops non-conformant packets. Some preliminary experiments were conducted using the AF PHB that are not reported in this paper, as the results were heavily dependent on the level of cross traffic and its impact on the performance given to marked packets. A comprehensive investigation of these issues calls for an altogether separate paper.

2.2 Streaming Video Technologies

New streaming technologies use special servers to deliver video content *intelligently and adaptively* to clients. This is accomplished through the use of a protocol between the server and its clients that allows continuous monitoring of the resources, e.g., bandwidth, available between them. Based on this information, the server can respond to network dynamics and deliver the content at a suitable data rate given the available bandwidth and the compression rate of the audio and video content.

Existing servers rely on either standard, e.g., MPEG [18], or proprietary, e.g., Windows Media Technology™ (WMT) [28], media formats, and this affects the type of video clips they can stream.

Standard servers such as MPEG streaming servers support either MPEG-1 or MPEG-2 encoded video with different rates and frame sizes. Often, large to medium scale servers are designed to deliver MPEG content types in broadband Intranet/Internet environments, i.e., environments where bandwidth is reasonably plentiful. *Microsoft Netshow Theater™, 2netfx-ThunderCastIP™*, and *IBM Video Charger™* are examples of such servers with which we experimented. These three servers are designed to deliver high quality full motion MPEG-1 or MPEG-2 digital video streams (up to 15 Mbps per stream) over IP networks. In our experiments, we noticed that even for constant rate encoding, all three servers generate reasonably bursty traffic, although with sizable differences in the magnitude of the burstiness. This is not entirely unexpected in the context of video, but does have important implications for the network and the potential benefits of service differentiation. The main contributor to the size of the bursts generated by each server appears to be the size of the (application level) datagrams they are configured to use. In particular, the first two servers are configured to generate large datagrams that can be up to 16280 bytes long[3], and which are then fragmented into smaller (1500-byte) packets by the IP stack on the server itself prior to their transmission on the network. This results in the generation of relatively large bursts of back-to-back packets. In contrast, the Video Charger™ server allows smaller message sizes so that while some burstiness remained in the traffic it generated, it was significantly lower than with the other two. Because of their propensity to generate large intrinsic bursts, the first two servers did not perform very well in bandwidth-constrained environments, and we found that their behavior in settings supporting service differentiation was of limited interest, i.e., mostly bi-modal with poor performance until sufficient (peak) bandwidth was allocated and nearly perfect performance thereafter. As a result, our experiments with video servers using standard video formats were mostly limited to the Video Charger™ server.

We also experimented with another server that exhibited similar burstiness characteristics, while relying on a proprietary data format. Specifically, we experimented with a *Windows Media Technologies™* server, which uses Microsoft Advanced Streaming Format (ASF) or Windows Media Video™ (WMV). ASF and WMV integrate and synchronize objects of different media types (audio, video, and data) optimized (encoded) for delivery over lossy networks. Both formats support multiple bandwidths within a single file. Media objects stored in the media files are encoded using a variety of encoders implemented by Microsoft media technologies. In addition, the server also supports reduced message size (UDP/TCP packet size) for its encoded content that can fit in a single packet. This eliminates the previously mentioned problem of generating large bursts of back-to-back packets because of large messages being fragmented in many small packets prior to transmission.

2.3 Video quality assessments

Rating the (user level) quality of video streamed over IP networks is a difficult task, and there are no standard procedures one can simply apply to generate the desired quality measures. As a result, the approach we took was to rely on methods that had been developed for video quality assessment in more traditional environments, i.e., television and video conferencing technologies, and then to adapt those techniques to our own environment. This task was further complicated by the fact that standardized objective video quality assessment methods for digital video systems are a recent development (see [26] for details and pointers). Video quality assessment can be performed in either a subjective or an objective manner. Subjective quality tests typically require a group of viewers to watch short video clips of approximately 10 seconds in duration in a very controlled environment, and then rate this material. The most widely used methods for measuring the subjective quality of speech and video images have been standardized and recommended by the International Telecommunication Union (ITU), and the results are frequently expressed in terms of the ITU-T *mean opinion score* (MOS) (see [7] and [9]).

Although subjective quality measurements reflect real human perception of the quality of a video stream, they involve a relatively complex and time-consuming process that is often not practical when a large number of configurations with varying parameters need to be assessed. These limitations of subjective quality measurement methods prompted the development of alternative approaches, called "objective quality measurement methods," that lend themselves more easily to automation and are capable of operating in less controlled environments. Objective quality measurement methods are based on computational models that combine a number of key video quality measures, and which are calibrated based on the correlation of the model scores to subjective scores for a number of pre-determined experiments. Just as with subjective methods, the various parameters that objective performance assessment tools can incorporate have been standardized by the American National Standards Institute (ANSI) (see [1]). In our experiments, we relied on an objective video quality measurement tool developed by the Institute for Telecommunication Sciences (ITS), which enabled us to evaluate the quality of a large number of combinations of video servers and formats and network configurations. In the next section, we provide additional details on both the video quality measurement tool and on the overall experimental setup that was used.

❑

[3] The datagram size for both the Netshow Theater™ and ThunderCastIP™ servers is actually proportional to the frame size and the MPEG encoding rate.

3. Methodology and Experimental Setup

The development of the various components required to carry out our video quality assessment experiments represented a significant fraction of the work, and several aspects turned out to be of independent interest. In this section, we briefly describe the three main pieces of our experimental setup: the video quality measurement tool, the network testbeds over which video was transmitted, and finally the video clips themselves, with their intrinsic properties.

3.1 Video Quality Measurement Tool

The tool and methodology that we adopted in our experiments are those developed by ITS, and which are based on the ANSI objective quality standards T1.801.03-1996 [1] as well as several more recently developed metrics [29]. The ITS Video Quality Measurement (VQM) software tool is based on a family of objective quality assessment methods called *Feature Extraction* or Reduced Reference [17], [27]. The approach is to rely on mathematical models to capture the major features of either individual frames (spatial features) or sequence of frames (temporal features) from both the received video stream and the reference video stream. The quality of a received series of video frames is then assessed by comparing the time histories of the received feature streams with the reference feature streams, and by combining multiple quality parameters so generated into an overall quality score. As mentioned earlier, the combination is performed so as to generate good agreement with the results of a number of previous subjective assessment experiments. In summary, the ITS tool follows this method and performs the following three steps to generate quality-rating indices.

- Extract quality features that characterize fundamental aspects of video quality (spatial detail, motion, color) from sequences of input and output video frames,
- Compute *perception-based* video quality parameters by comparing the features of the received (output) video frames with the corresponding features of the original (input) video frames, and
- Produce a composite quality score from the computed digital video quality parameters that is highly correlated with the subjective assessments of human viewer panels.

A number of additions were needed in order to be able to use the VQM tool to assess the quality of video transmitted over an IP network. This is because the tool was originally developed for television and video conferencing systems, where both the video formats and the content delivery mechanisms are completely different from those used to transmit video over IP networks. The tool runs on an IRIX™ platform and takes decoded (uncompressed) video as input in the YUV 4:2:2 [8] file format, which is a binary file format used by the ITU Video Quality Experts Group (VQEG) (this file format will henceforth be referred to as BigYUV, since all the YUV frames from a scene are stored in one large file). The main challenge for us was to generate an appropriately formatted input for the VQM tool, based on the received video streams after they were transmitted over the network. This difficulty was compounded by the fact that because the processing involved is computationally intensive, the tool is typically unable to process received video streams in "real-time." As a result, it is necessary to provide some intermediate storage of the received video prior to feeding them to the tool. This introduced a number of additional problems.

The first problem is that video streaming clients are typically built to only render contents on a screen and do not have the option of recording or saving the received video. As a result, it was necessary to add an intermediate step in which received video streams are saved to a file. The quality measurement process could then be conducted offline, i.e., after the streaming process, by presenting the saved video frames to the VQM tool. The second problem associated with storing the received video stream was that this additional step had to accurately preserve the perturbations, e.g., information regarding frame delays and drops, introduced by the various network configurations being tested. Finally, a third problem we encountered was caused by the fact that the tool, like most other objective assessment tools, was designed to handle relatively short duration video segments, e.g., on the order of 5 to 10 seconds. We wanted to experiment with longer video segments, e.g., between 75 and 150 seconds, to be able to consider video clips that would incorporate a broader and more representative range of scene types, and get a more realistic assessment of the overall impact of network configurations. As a result, some care had to be exercised when using the tool to process such extended segments. In the rest of this section, we outline how we addressed these problems in developing our experimental setup.

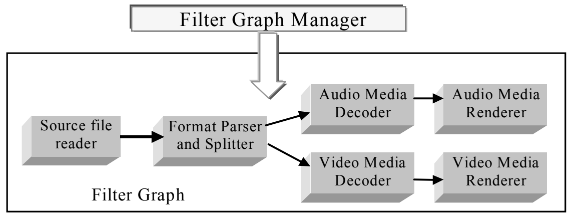


Figure 1. Directshow™ filter graph and graph manager.

3.1.1 Storing Received Video Frames to File

We investigated a number of approaches for implementing the required intermediate video storage step, and ultimately settled on one that had the benefit of being readily portable, at least across the video clients we were using that all ran on the Microsoft Windows™ platform. Most of the multimedia clients built for the Windows™ platform are implemented using the Directshow™ architecture. Directshow™ controls and processes the playback of multimedia streams from local files or network servers. It enables the playback of compressed video and audio contents using a modular approach that divides the processing of multimedia objects into a set of stages known as *filters*. Filters are pluggable components connected to each other through "pins" that represent a logical point of connection for a unidirectional data stream flowing to/from the filter. A *filter graph* is composed of a collection of connected filters of different types used to process a specific media format. Applications use what is called a *filter graph manager* to assemble and connect the filter graph suitable for their media format, and for controlling the movement of data through the assembled graph. When an application starts rendering a piece of media content, the filter graph manager first selects a source filter capable of reading the media content, and then proceeds to select and connect subsequent filters based on the filter graph, the last filter being typically the rendering stage. An illustration of the structure of a typical filter graph is shown in Figure 1.

Given the modular structure of the Directshow™ architecture, a natural approach for introducing the ability to store received video frames is to develop a filter performing the required storage operations, and to insert it at an appropriate location in the filter graph of video clients. The location we chose to insert our "storage" filter was after the video decoder in lieu of the renderer. The one disadvantage of such a choice is the storage requirement it implies, as video frames must be stored in an uncompressed BigYUV format. However, our experience has been that this disadvantage was more than compensated for by a greater robustness and reliability across clients and media formats. In addition to replacing the renderer filter by our new storage filter, another modification that had to be made to the original client filter graph was to ensure that the video decoder would produce an output in the desired BigYUV format. This was relatively easy since most decoders allow the application to select the output format they will generate.

3.1.2 *Capturing Network Dynamics Information*

The storage filter receives each frame, captures the relevant timing information that consists of its arrival time and target presentation time, and then saves the frame in a binary file and the timing information in a parallel ASCII file. The timing information will be used to create a modified set of frames that will be fed to the VQM tool, and that will emulate the impairments caused by network perturbations, as a user viewing frames being played out by the renderer would have perceived them. Specifically, we want to capture in our quality assessment the effect of the techniques used by most renderers[4] to compensate for lost or delayed frames. The most common and simplest technique is to keep repeating the last received frame until a new frame arrives. This is the approach we chose to emulate. This was implemented using a simple PERL script that takes as input the initial file of stored received frames together with the associated statistics file created by the filter. From these inputs, the script produces a new file containing stored frames, but which now incorporates the repeated frames that a renderer would have generated while attempting to compensate for lost or delayed frames.

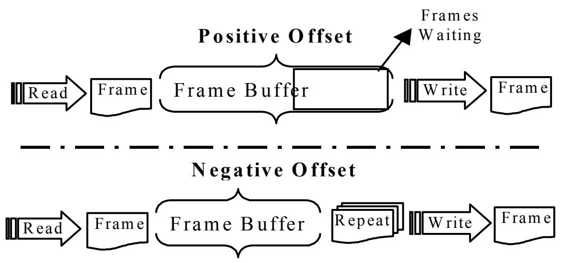


Figure 2. Handling of Lost or Delayed Frames.

The mechanism on which the script relies is based on maintaining two time references associated with the presentation time and the arrival time, and by comparing them to determine when the playback buffer runs out of frames because of lost or delayed frames. This is achieved by maintaining an offset value representing the difference between the arrival and presentation times of frames. Frames that are early or on time have an arrival time less than their presentation time, and result in an increase of the offset value by an amount equal to their difference. In contrast, a late frame has an arrival time that is greater than its presentation time, in which case this negative value is used to decrease the current offset value by a similar amount. The script updates the offset value for each received frame, and the resulting value is used to determine the state of the playback buffer as seen by the renderer. A positive offset value indicates that frames are available for rendering in the buffer, while a negative offset value corresponds to an empty playback buffer that would trigger the repetition of the previous frame by the renderer. As a result, the script inserts copies of the previous frame in the output file it produces. The number of copies that are inserted is a function of the offset value and the presentation and arrival times of the next available frames. An illustration of the process implemented by the script is shown in Figure 2.

3.1.3 *Handling Extended Duration Video Clips*

As mentioned earlier, the VQM tool was originally designed to measure the quality of short (5 to 10 seconds) duration video segments, while we wanted to use longer (between 75 and 150 seconds) video clips in our experiments. In order to use the VQM tool on those longer clips, we therefore had to divide them into smaller 10-second segments that were then fed to the tool one by one, and whose individual quality scores had to be combined. Applying this process raised two questions. The most significant one involved the *calibration* process that is used to remove systematic errors (i.e., gain, spatial shift, temporal shift) from the received video stream. A control file that performs both spatial and temporal calibration between the two sets of frames drives this calibration. The second question concerned how to combine the scores obtained by short duration segments to produce a meaningful overall score for the extended duration video clip.

We configured the tool to segment the stored video files into segments consisting of 300 frames each (10 seconds duration). The segmentation is done so that the first 100 frames of each segment overlap with the last 100 frames of the segment preceding it (see Figure 3). Note that the last 100 frames of the last segment and first 100 frames of the first segment do not overlap with other segments. The overlap of consecutive segments is used to provide sufficient margin to allow the temporal calibration mechanism of the tool to find the proper alignment between the original file and the received file. This is achieved by setting the *Alignment Uncertainty* parameter specified in the control file of the tool to allow searching in this specified range. The quality estimation of a segment is then based on the next 100 frames following the alignment point identified by the temporal calibration mechanism.

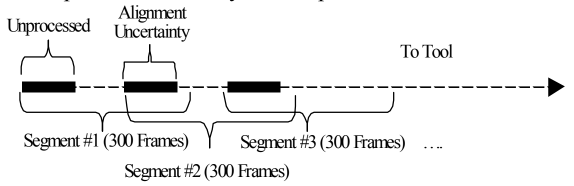


Figure 3. Segmentation Process of the Stored Video File.

In order to calculate the overall quality of an extended video clip, we simply averaged the quality estimates of all the individual segments. However, some care had to be exercised to deal with long (around 10 seconds or more) periods of degraded quality, which the temporal calibration process was not able to deal with even with its extended range. Specifically, segments for which the temporal calibration process did not succeed were assigned a default quality index of 1 that corresponds to the worst quality index assigned by the tool (the best quality index assigned by the

❑ ________________________________

[4] It is the renderer rather than the decoder that is often responsible for concealing the impairments caused by excessive delays or lost frames in the network.

tool is zero). Here, the term “quality index” refers to the quality estimate produced by the tool.

3.2 Network Testbeds

Quality assessments of received video clips were performed using two network testbeds. One was a local testbed consisting of several Diff-Serv capable routers to which a video server and video clients were connected. The ability to easily change the configuration of the local routers as well as the easy access to the video server facilitated the exploration of a relatively wide range of configurations. However, it is clear that a local testbed cannot emulate all of the possible interactions that occur in actual networks. In particular, while the policing actions of the token bucket are expected to be the dominant contributor to video quality, there are other factors that can influence token bucket operations and introduce additional perturbations to a video stream. For example, interactions with cross traffic prior to reaching the router where policing actions are performed can impact the number of frames that are found non-conformant and, therefore, discarded (this jitter effect is well-known and was the motivation behind the introduction of cell delay variation tolerance in ATM [3]). Similarly, the use of shapers[5] as well as the traversal of multiple network hops can affect the end-to-end quality perceived by a video application. As a result, it is desirable to carry out experiments over a broad range of configurations. For that purpose, additional tests were conducted over the QBone; a wide-area testbed implemented across the Internet2 Abilene network that supports a service built on the EF PHB. The rest of this section is devoted to a brief overview of the two testbeds.

3.2.1 Local Testbed

The local testbed consists of three Diff-Serv enabled routers, three workstations, and several Ethernet hubs that were used for local connectivity. A sample configuration of the different testbed elements is shown below in Figure 4. Routers 1 and 2 are connected through a Frame Relay over High Speed Serial Interface (HSSI), which is a standard interface for high-speed (up to 52 Mbps) serial connections over WAN links. The two routers are connected directly using a HSSI Null modem cable. The third router, router 3, is connected to router 2 using Frame Relay over V.35.

3.2.1.1 Workstation Configurations

The workstations used for the video server and for the video client are running the Microsoft Windows™ operating systems. A higher end system was allocated to the video client because the combination of real-time decoding of the received video streams and the storing of the resulting large volume of data on the hard disk proved to require significant resources. Specifically, with a frame size of 320x240 pixels, the size of a decoded frame in BigYUV format comes to 153.6 kbytes or 1.2 Mbits. Given a frame playout rate of 30 frames per second, this imposes some reasonably stringent speed constraints on the hard disk of the receiving client system (about 35.16 Mbps).

❑

[5] A shaper is a token bucket, which instead of simply dropping (policing) non-conformant packets, is configured to delay them until the earliest time at which they are deemed conformant.

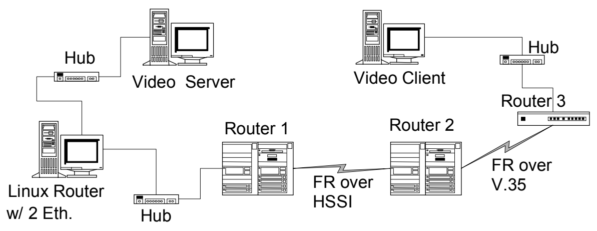


Figure 4. Sample Configuration of the Local Testbed.

In the local testbed, the video server that was used in our quality assessment experiments was the Microsoft Windows Media Technologies™ (WMT) server, which uses the Microsoft proprietary media streaming protocol called Microsoft Media Streaming™ (MMS). The protocol uses a TCP connection for sending and receiving media control commands, and a UDP or TCP connection for streaming the data.

The Linux workstation shown on the left-hand side of Figure 4 is running RedHat™ 6.0 and was configured to act as a router that was used to perform traffic *shaping* in some of our experiments. The use of a shaper prior to the router responsible for implementing the token bucket in charge of policing the video traffic was motivated by the relatively bursty characteristics of the traffic generated by the video server (see Section 4.2).

3.2.1.2 Router Configurations

The two major aspects of the router configurations are the parameters of the frame relay interfaces and the configuration of the policy component responsible for traffic policing.

Frame Relay Interfaces Configuration

Router #	I/f #	CIR	Bc	Be	I/F Type
2	FR 1	2*106	2*106	0	V.35
	FR 0	2*106	2*106	0	HSSI
1	FR 1	2*106	2*106	0	HSSI
3	FR 1	2*106	2*106	0	V.35

Table 1. Configurations of the Frame Relay Interfaces.

The configuration of the frame relay interfaces on the three routers is given in Table 1, which specifies for each interface the values used for the three parameters required by frame relay: Committed Information Rate (CIR), Committed Burst Size (Bc), and Excess Burst Size (Be). The main purpose of the configurations used was to emulate a set of constant rate links connecting the different routers. Note that all CIR values were less than the maximum transmission speed allowed over the V.35 Interface (E1), which was the main bandwidth bottleneck of the system.

Support for different levels of service was provided through a simple priority queue structure, with the high priority queue being assigned to traffic marked with the EF DSCP.

Policy Configuration

Diff-Serv policies were configured on routers 1 through 3. A policy specifies a “profile” that identifies the packet to which the policy applies, and an action that determines the treatment that these packets are to receive. At router 1, the profile specifies the source address of the video server and the destination address of the video client, which will then trigger the creation of a classifier entry at the router to extract the corresponding set of packets. The output of the classifier is connected to a policer that is responsible for marking conformant packets with the EF DSCP (101100) and for forwarding them to the router's high priority queue. The policer's configuration information includes parameters such as token rate and token bucket depth, as well as the treatment of non-conformant packets. Given our focus on a service based on the EF PHB, the policer was configured to drop packets that it did not find conformant. As mentioned earlier, this was meant to allow us to assess the impact that such "hard" policing actions had on the quality of the received video. In addition, the token bucket depth was typically set to two link MTUs in order to limit the size of the burst that was allowed to enter the network. The impact of dropping non-conformant packets was assessed with and without the Linux router performing a shaping function. Policy specification at routers 2 and 3 was simpler, as it only amounted to classifying packets marked with the EF DSCP and forwarding them to the high priority queue at the router.

3.2.2 *QBone Testbed*

The QBone [25] is an inter-domain Diff-Serv testbed aimed at exploring the benefits of service differentiation on a wide area scale. QBone participants are considered to form Diff-Serv enabled domains that are in turn connected to the QBone, with which they agree on Service Level Specifications (SLS's) that define how traffic is classified, policed, and forwarded at boundary nodes. The edge routers located at the ingress boundary nodes are responsible for enforcing SLS's. The QBone currently offers one service, the QBone Premium service that is based on the simplex Abilene Premium Service (APS), which is built on the Expedited Forwarding (EF) PHB. The SLS for the service was in the form of a token bucket that specified peak rate and maximum burst size. At the time our experiments were carried out, the APS test program was in its early phase and the Abilene network was lightly loaded, so that except at boundary nodes, the APS service was implemented simply by means of over-provisioning. As of this writing, service support has been upgraded to also provide preferential forwarding treatment to EF packets within the Abilene backbone.

Figure 5 provides an overview of the connectivity across the QBone between our local testbed and a remote site. A video server (Video Charger) was located at the remote site and used to stream across the QBone different video clips to a video client located in the local testbed. The packets generated by the server were pre-marked as EF packets by the server and were policed at the border Cisco router of the remote site. Policing was performed using Cisco's Committed Access Rate[6] (CAR), which was configured to drop packets that exceeded the APS profile. Experiments were conducted using various APS profiles with different token rates and token depth values, as shown in Section 4.1.

[6] For a brief introduction to CAR, please refer to [10].

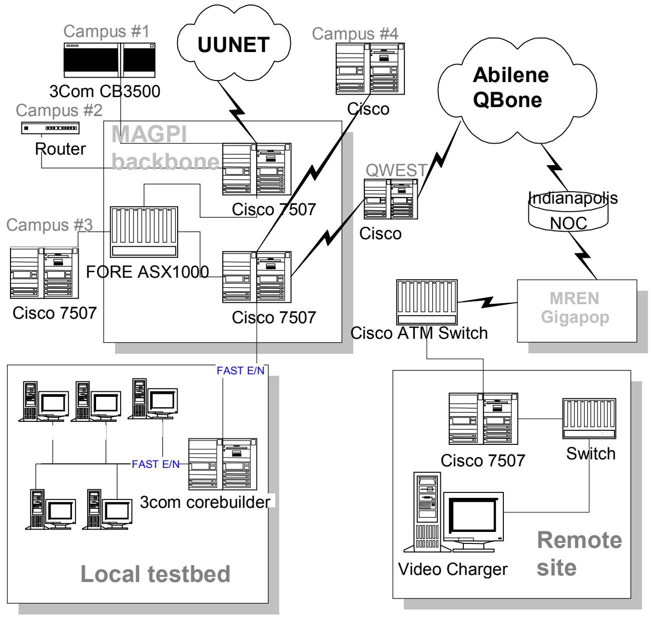


Figure 5. Connectivity Configuration Across the QBone.

3.3 Video Clip Properties

Two video clips, called *Lost* and *Dark* in the rest of the paper, with different scene characteristics were used in our experiments. They were obtained from the trailers of two motion pictures, and were re-encoded using two different encoding formats (MPEG-1 and WVM) and a variety of encoding parameters corresponding to different levels of video "quality." The clips were uploaded to different servers and streamed to clients using a number of network service configurations, i.e., specifying different token bucket parameters. The MPEG-1 versions of the clips were used for experiments carried out over the QBone, and they were uploaded to the video charger server located at the remote site. The WVM format clips were used with the Windows media technologies server in our local testbed experiments.

3.3.1 *MPEG-1 Clips*

The two clips were encoded using a *constant bit rate setup* for three different rate values 1M, 1.5M, and 1.7M, and a fixed frame size of 320x240. The three rate values produced videos of different qualities but potentially better suited to different network configurations, i.e., token rates. Note that the MPEG servers we used do not support multi-rate encoding, i.e., the ability to dynamically select a given video quality when multiple copies encoded at different rates are available. As a result, and although we expect such a capability to be available in future MPEG servers, this means that once a given encoding has been selected, it is the only one used for the remainder of the experiment. The characteristics of the two clips are summarized in Table 2 and Figure 6. The table gives the characteristics of the encoding process and its output (the rate information is computed after every frame using the MPEG_stat tool), while Figure 6 displays rate information for what is actually transmitted to the network.

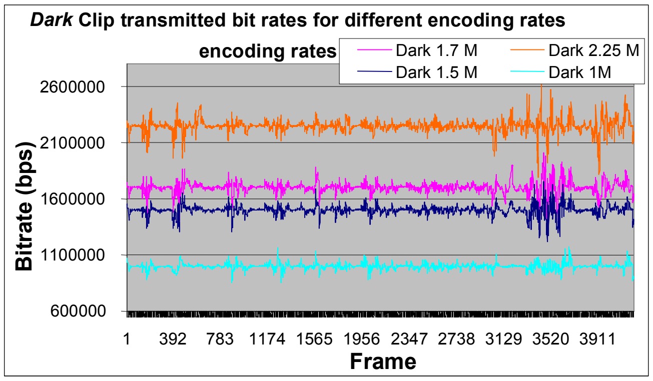
As can be seen from both the table and the figure, although a constant rate encoding was used, the resulting output still exhibits significant variations, as illustrated through the differences between minimum, average, and maximum rate values in the table and the range of transmission rates found on the figure.

Clip *Lost*							
Encoding rate	Bytes read	frames	Length	Avg. Frame size	Rate Information in bps		
					Max	Avg.	Min
1.7M	15276442	2150	71.74 s	7101 bytes + 4 bits	2047496	1702659.43	128640
1.5M	13453779	2150	71.74 s	6253 bytes + 6 bits	1835320	1499402.84	117976
1M	8970075	2150	71.74 s	4168 bytes + 2 bits	1263464	999396.85	87744

Clip *Dark*							
Encoding rate	Bytes read	frames	Length	Avg. Frame size	Rate Information in bps		
					Max	Avg.	Min
1.7M	29975812	4219	140.77 s	7101 bytes + 2 bits	2038840	1702624.58	153152
1.5M	26399218	4219	140.77 s	6253 bytes + 5 bits	1789408	1499371.64	139088
1M	17600951	4219	140.77 s	4168 bytes + 1 bits	1155672	999378.16	97592

Table 2. MPEG Encoding Properties of Clips *Lost* and *Dark*.

Figure 6 also illustrates the impact of the differences in scenery between the two video clips, with the high motion content of the clip *Dark* translating into greater rate fluctuations (especially towards the end of the clip). We expected those fluctuations to have some impact on the interactions between the policer used for different configurations and the resulting video quality.



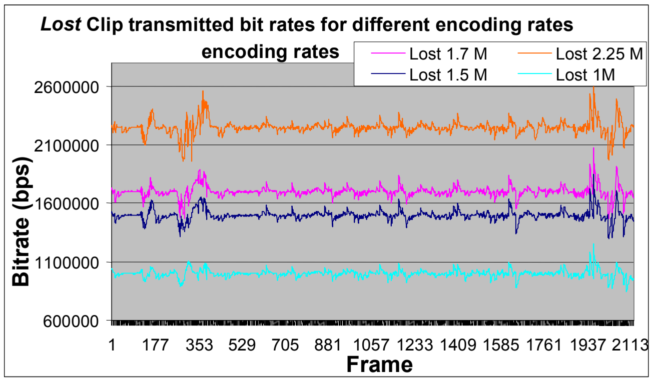


Figure 6. Instantaneous Transmission Rates of MPEG-1 Clips for Different Encoding Rates.

3.3.2 *Windows Media Encoded Clips*

The properties of the two clips encoded using the Windows Media™ encoder are summarized in Table 3. The encoder allows the specification of several encoding parameters, including the desired bandwidth. However, the resulting encoding produced by selecting a given bandwidth value is not a constant rate encoding, and instead corresponds to a maximum bandwidth value. This can be seen by comparing the *expected* and *average rates* values in Table 3. Note that the WMV format supports the multi-rate encoding feature mentioned earlier, and we used it in some experiments. Note also that the encoder does not encode video-only content, so that both audio and video had to be accounted for. In order to minimize the effect of the additional audio packets, we set the encoding rate for audio near zero.

Lost Clip	*Dark* Clip
Session:	Session:
Bytes encoded (total): 6936504	Bytes encoded (total):11976984
Bit rate (expected): 1015.5 Kbps	Bit rate (expected):1015.6 Kbps
Bit rate (average): 771.7 Kbps	Bit rate (average):680.5 Kbps
Video [1015.4 Kbps]:	Video [1015.5 Kbps]:
Bytes encoded (total):6935380	Bytes encoded (total):11974782
Bit rate (expected):1015.4 Kbps	Bit rate (expected):1015.5 Kbps
Bit rate (average):771.6 Kbps	Bit rate (average):680.4 Kbps
Frames per second (expected):30.0	Frames per second (expected):30.0
Frames per second (average):29.9	Frames per second (average):30.0
Frames (total):2150	Frames (total):4219
Audio [0.1 Kbps]:	Audio [0.1 Kbps]:
Bytes encoded (total):1124	Bytes encoded (total):2202
Bit rate (expected):0.1 Kbps	Bit rate (expected):0.1 Kbps
Bit rate (average):0.1 Kbps	Bit rate (average):0.1 Kbps
Samples (total):562	Samples (total):1101

Table 3. Properties of Windows Media Encoded Clips.

4. Results

As mentioned in Section 2.2, we originally considered a number of different video servers but ultimately limited our experiments to only two types of servers: The Video Charger™ server for experiments over the QBone and a Windows Media™ server for experiments over our local testbed. Table 4 summarizes the different configurations used.

As mentioned previously in Section 2.2, the main motivation for not including the other servers we initially considered was their relatively poor performance in the presence of the dropped packets induced by traffic policers. Recall that those servers rely on large datagrams for the transmission of video frames, and those datagrams were then fragmented into many smaller packets, so that the loss of even one packet at the policer would typically result in the loss of an entire datagram. This problem was further compounded by the fact that a single datagram triggered the generation of many back-to-back packets, which resulted in several dropped packets at the policer because of the small token bucket depth used for EF traffic. In addition, policing losses together with the service guarantees provided to EF traffic appeared to somewhat confuse the adaptation mechanism of the servers. Specifically, the fact that delivered packets experienced small delays seems to have been interpreted by the server as an indication that sufficient bandwidth was available. As a result, the adaptation mechanism reacted to the loss of packets (because of policing) by forcing the server to increase its data rate to make up for the losses. This in turn resulted in further packet losses followed by yet other rate increases until performance got so poor that the server would back down to very low transmission rates. This cycle would repeat a number of times, until the client decided to break the connection, as it was deemed too unreliable. In short, traffic conditioning essentially misled the dynamic rate control approach of the servers, to the point of making them unusable unless the token rate was set to the maximum rate of the server.

	Experiments on	
	QBone	Local Testbed
Video Server used	Video Charger	Windows Media Server
Network protocol	UDP	TCP, UDP
Contents Type	MPEG1	WMV Format
Contents properties	Constant Bit rate	Max bit rate is constant
PHB tested	EF	
Service parameters	Token rate, Bucket Depth	
Out of profile action	Drop	Drop (router 1) (Shape – Linux router)

Table 4. Summary of Experimental Configurations.

The main results of our experiments consist of the quality estimates generated by the VQM tool for the different configurations we tried. Before we proceed with the description and discussion of the results, it is important to note that there is some variability in the results themselves. Specifically, for the same combination of video server, video client, and network parameters, it is possible to obtain slightly different quality estimates in consecutive runs of an experiment. This is because many factors can affect the set of packets ultimately delivered to the client together with their timing. For example, different load conditions at the server or variations in the level of interfering traffic through the local network connecting the server to the router performing policing, can all influence the set of packets that the policer will ultimately drop. Those differences in lost packets will in turn affect the resulting quality of the video played out at the client. In particular, as seen in, say, in Figure 7 and Figure 8, it is quite possible for a small increase in token rate to yield a degraded video quality. This is in part because depending on the types of scenes, the intrinsic rate of the video clip, and how the server reacts to a slight increase in token rate, the end-result need not always be fewer dropped frames. Such inherent issues not withstanding, we have tried to minimize such variations by eliminating most external interference sources, e.g., dedicated video server, absence of local interfering traffic, etc., so that the focus was on the impact of policing actions for different configurations. Note that a few experiments on the local testbed did involve interfering cross-traffic, and the QBone experiments did not allow us to control the presence and absence of interfering traffic. In all cases where we were able to compare the outcome of experiments with and without interfering traffic, only minor variations were observed that were primarily a reflection of how the different routers implemented the prioritization of EF traffic. In general, it is impossible to completely eliminate all sources of variation, and although results did not vary significantly when experiments were repeated, it is important to keep this in mind when interpreting the results. In other words, general trends are clearly meaningful, but minor fluctuations in quality need not be.

4.1 QBone Testbed Results

Copies of the clips *Dark* and *Lost* encoded at the different rates identified in Section 3.3 were streamed through the QBone from a Video Charger server located at the remote site to a video client at the local site (see Figure 5). Streaming was done over UDP, as this was the only configurable option at the remote server when EF marking was also to be applied. Each clip was streamed through the network several times and for different choices of network service parameters (token rate and bucket depth). Two (small) token bucket depth values were used, namely 3000 bytes and 4500 bytes, and for each the token rate value was varied from just below the average stream rate to a value for which the maximum video quality rating of 0 was achieved. This typically corresponded to the maximum rate of the video stream. Initial results are shown in Figure 7 through Figure 9 for the *Lost* clip and in Figure 10 through Figure 12 for the *Dark* clip. Each figure has two sets of curves, one for each token bucket depth. The two curves in each set correspond to the fraction of lost frames and the corresponding video quality rating produced by the VQM tool. For comparison purposes, those values are plotted against the same y-axis scale, while the x-axis corresponds to increasing token rate values. Tables available from [2] provide more precise numerical values. Recall that a quality score of 1.0 is the worst possible[7], while a score of 0.0 corresponds to the best possible video quality, i.e., identical to the quality of the original clip used as reference.

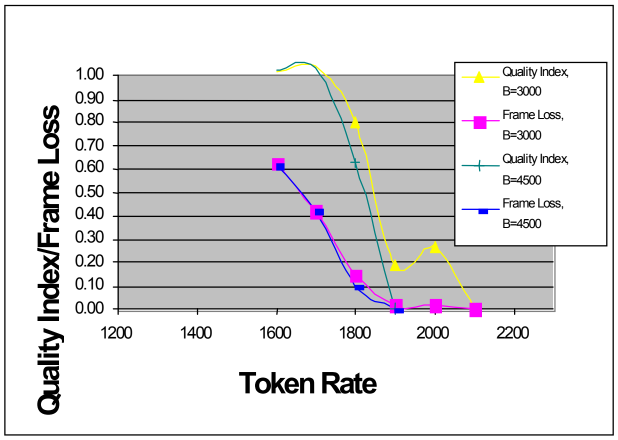


Figure 7. QBone Streaming (*Lost* clip/1.7Mbps encoding): Video Quality & Frame Loss vs Token Rate.

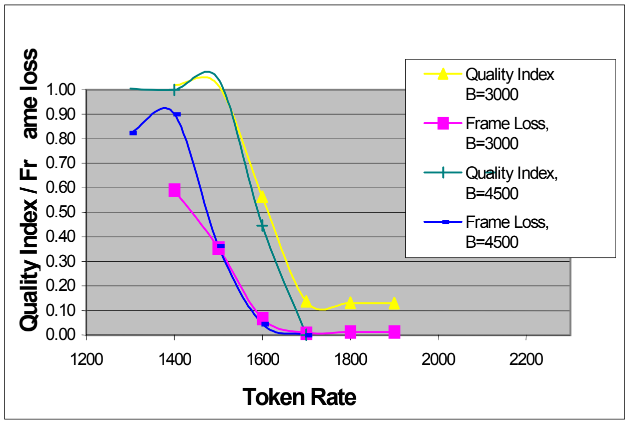


Figure 8. QBone Streaming (*Lost* clip/1.5 Mbps encoding): Video Quality & Frame Loss vs Token Rate.

[7] Quality index scores may exceed 1.0 for extremely distorted video that falls outside the range of subjective assessments used to develop the VQM tool.

In this first set of experiments, the quality of the received video was compared to that of the transmitted clip, i.e., the reference points were different for each encoding rate. The first set of experiments was performed to assess the quality degradations resulting from network impairments. In a latter set of experiments, the comparison was done with respect to the highest quality original clip, namely the clip corresponding to a 1.7 Mbps encoding rate, so as to assess the trade-off that exists between quality degradations imposed by the network and those due to the encoding itself.

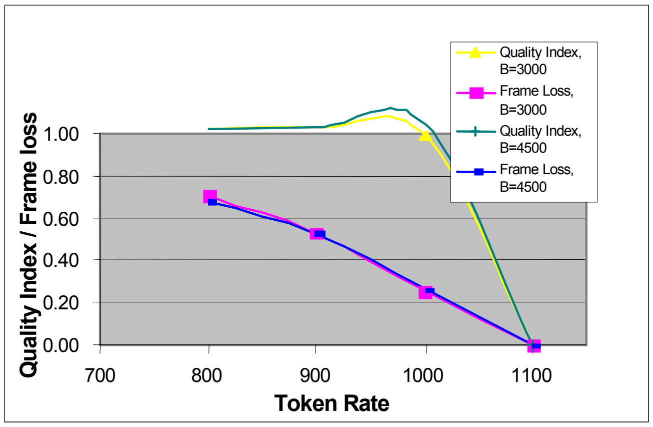


Figure 9. QBone Streaming (*Lost* clip/1.0Mbps encoding): Video Quality & Frame Loss vs Token Rate.

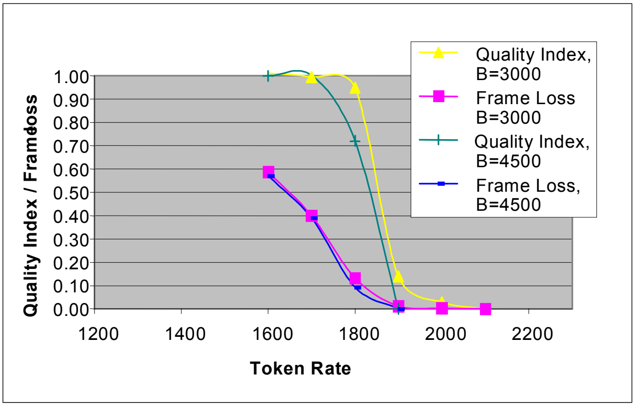


Figure 10. QBone Streaming (*Dark* clip/1.7Mbps enc.): Video Quality & Frame Loss vs Token Rate.

These first results show very similar behavior for the two clips, *Lost* and *Dark*, which appears to indicate that the different motion characteristics of their content do not significantly affect the basic relation that exists between video quality and network resources. In other words, while video characteristics will clearly play a role in determining the absolute level of quality achievable given certain network resources, general trends should remain similar across different types of video clips. For example, when comparing the results of, say, Figure 7 and Figure 10, that correspond to the 1.7Mbps encoding rate versions of the *Lost* and *Dark* clips, respectively, we see that for a token bucket depth of 3000 bytes and a token rate of 1.9 Mbps, both clips experience a similar frame loss of about 1%, but their respective quality measures differ, i.e., 0.19 versus 0.14. However, despite those differences that are actually more pronounced for higher encoding rates, the general "shape" of the quality index curves are similar for the two clips.

There are a number of initial conclusions one can draw based on the results obtained from this first set of experiments. The first and most interesting one is that the relation between video quality and network level performance improvements is highly non-linear. For example, we see that in some regions, improvements in frame losses hardly affect the quality of the received video that remains relatively poor until a cutoff point is reached. Once this cutoff point is passed, video quality improves at a much faster pace than the corresponding improvements in frame loss. The location of the cutoff point as well as the subsequent difference in slope between video quality and frame losses vary based on the encoding rate as well as the type of clip, but the behavior is consistent across all the experiments.

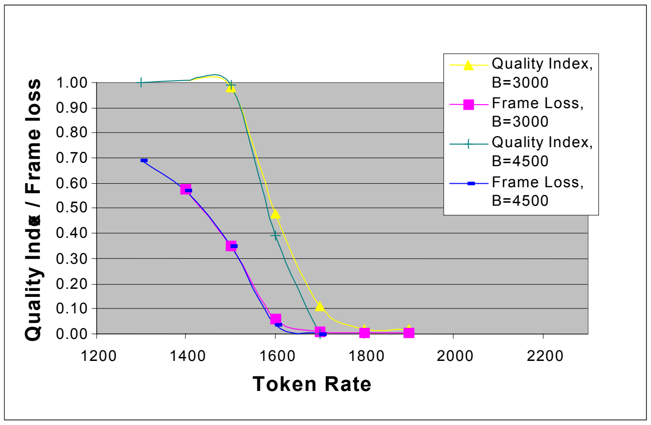


Figure 11. QBone Streaming (*Dark* clip/1.5Mbps enc.): Video Quality & Frame Loss vs Token Rate.

In addition to this general conclusion, the experiments also provide more specific information in terms of the network "service" parameters required to ensure adequate video quality. The first observation is that setting the token rate value below the encoding rate is of no use at all. This is not surprising given the dropping actions taken on non-conformant packets by the ingress policer. The token rate value needed to achieve good video quality, i.e., a score close to 0, depends on the token bucket depth. We see that with a token bucket depth of 3000 bytes (2 Ethernet MTUs), the token rate has to be set to a value around or even above the maximum encoding rate of the clip (see Table 2 for rate values) in order to approach the desired quality score. However, when the token bucket depth is increased to 4500 bytes, a token rate set to the average (constant) encoding rate is typically sufficient. Note that this is despite the fact that the network level transmission rates of the video streams still exhibit significant variations (see **Figure 6**). Nevertheless, this points to the fact that a service built on the EF PHB may not be really suitable to the efficient transmission of streaming video, because of the constraint it imposes on the token bucket depth. This is not a major surprise in itself, but the results of the experiments help quantify this behavior. In particular, we see that although the token bucket depth needed is greater than the "typical" 2*MTUs often quoted for the EF PHB, it is not significantly larger, at least for video clips using constant bit rate encoding. This means that a minor relaxation on token bucket depth limits may make such a service more useful to video streaming applications. Clearly, such relaxation needs to be weighed against its impact on delay and losses in the network, but depending on the intended use of the service, it may be worth considering.

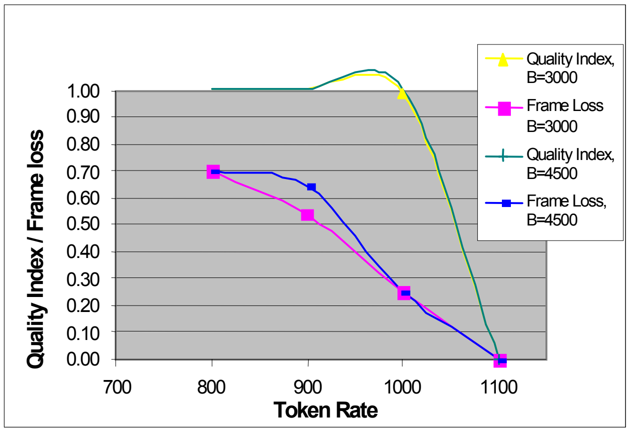


Figure 12. QBone Streaming (*Dark* clip/1.0Mbps enc.): Video Quality & Frame Loss vs Token Rate.

The next set of experiments carried out across the QBone attempted to answer the more general question of the relation between initial video quality and available network resources. In other words, is it better to lose a relatively large number of packets from a high quality video stream, or is it better to loose fewer packets from a lower quality video. The latter seems the intuitively natural answer, but we wanted a more quantitative assessment of this trade-off. For that purpose, we conducted another set of experiments, using again the *Lost* and *Dark* video clips encoded at the three rate values of 1.0Mbps, 1.5Mbps, and 1.7Mbps, and compared the quality of the received video to that of the highest quality 1.7Mbps version of the clip. As before, the comparison was carried out for different values of the token rate. A token bucket depth of 3000 bytes was used for all the experiments. The results are reported in

Figure 13 and Figure 14 for the *Dark* and *Lost* clips, respectively.

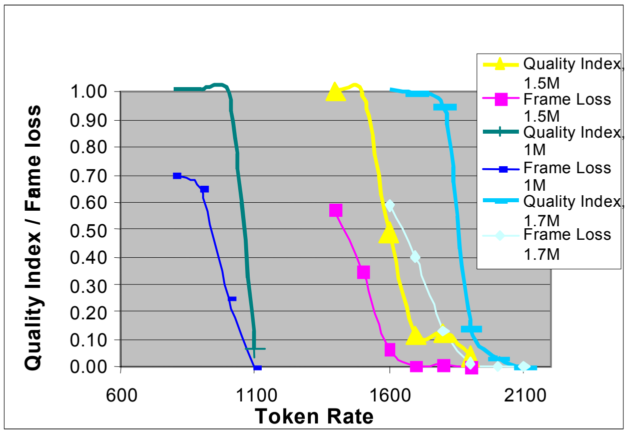


Figure 13. Frame Loss and Relative (compared to 1.7Mbps version) Quality for *Dark* Clip.

The results from the two figures essentially confirm our earlier intuition that one should select an encoding rate that is near but below the specified token rate. This is clear from the data in the figures: as for all token rate values the best performing stream in terms of quality score is always the one with the closest encoding rate. In other words, at least for the range of encoding rates available in our experiments, the basic rule of thumb to apply is to select the largest encoding rate that is less than the token rate. This seems to be primarily due to the fact that the impairments caused by packet losses have a much larger impact on video quality than the differences in raw video quality caused by different encoding rates.

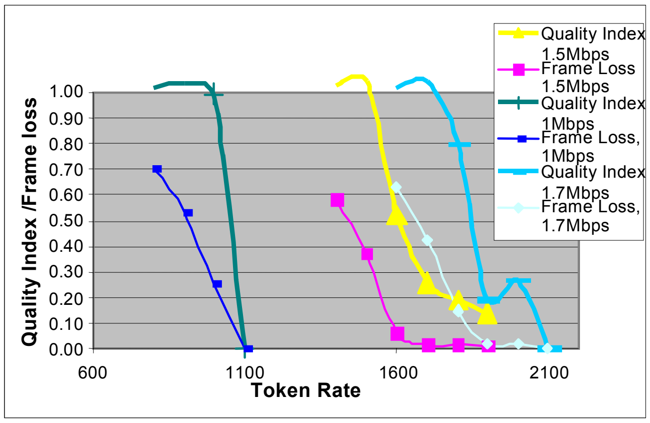


Figure 14. Frame Loss and Relative (compared to 1.7Mbps version) Quality for *Lost* Clip.

4.2 Local Testbed Results

Experiments conducted in the local testbed were aimed at exploring the same set of issues as those that motivated the QBone experiments, but this time using a different video server and possibly more configurations because of the accessibility of the local equipment. The motivation for using a different server was to determine the level of dependency of our initial conclusions on the specific characteristics of the video server we used. The use of a different server at the remote site, which would have enabled us to carry this new set of experiments over the QBone, was not feasible due to logistics constraints. As a result, this next set of experiments was conducted over the local testbed described in Section 3.2.1. The basic conclusion from these additional experiments is that most of the earlier findings remain essentially true. In particular, the fact that frame loss is not an accurate predictor of video quality still holds. As before, in many cases small differences in frame loss resulted in large differences in video quality and vice-versa. These behaviors are illustrated in Figure 15 and Figure 16, where we can again see areas of rate changes that yielded substantial reductions in frame losses without comparable improvements in video quality, as well as cases where a relatively small decrease in frame loss resulted in a significant improvement in video quality.

Another behavior that was consistent with our observations from experiments over the QBone was the impact of increasing the token bucket depth. As before, a small increase in the token bucket depth, i.e., from 3000 bytes to 4500 bytes, resulted in significant improvements in our ability to achieve nearly perfect video quality using a token rate value close to the encoding rate of the video stream itself. However, despite those similarities, some differences were also observed. They reflected the impact of a different server technology as well as differences in the network configurations that were used, and it is worthwhile to briefly review them as well as their causes.

The main difference is that much higher token rates were required in order to achieve nearly perfect video quality. As a matter of fact, because our maximum token rate was limited by the speed of the V.35 link between routers 2 and 3 (about 2 Mbps), we could not exceed this value, which prevented us from achieving the ideal quality score of 0 when the token bucket depth was 3000 bytes. This was in spite of the fact that the "maximum" rate specified when encoding both clips was limited to about 1 Mbps (see Table 3). In other words, despite a token rate of about twice the maximum encoding rate, we were still not able to achieve the best quality level. Note that increasing the token bucket depth to 4500 bytes largely eliminates this difference. The main reason behind this behavior is the relatively bursty nature of the server's output. As a matter of fact, the main reason that the experimental results we report are limited to the 1 Mbps clip is that we were not able to achieve decent levels of quality at higher rates, at least not with the network configurations we were using. This was observed even after switching over to TCP streaming and relying on shaping in the Linux router to which the server was connected. UDP streaming remained too bursty to allow meaningful experimentation in the network configurations we were considering. TCP streaming, because of the intrinsic rate adaptation capability of TCP, resulted in a smoother traffic flow that produced better quality results.

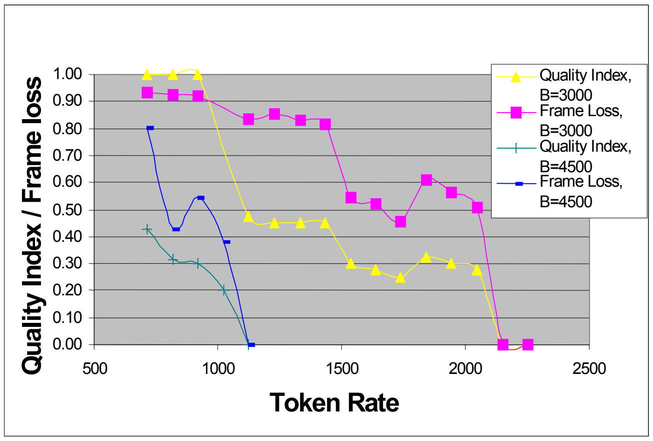


Figure 15. Local Testbed Experiments (*Lost* clip at 1Mbps) – Quality and Frame Loss vs Token Rate.

A related point worth emphasizing in this context is that the difference in video quality achieved by increasing the token bucket depth from 3000 bytes to 4500 bytes is much more substantial in this setting. In other words, the benefits derived from allowing a slight increase in bucket size are much larger with this type of server and encoding, than when using constant rate encoding together with servers that attempt to pace their transmissions, as was the case for the QBone experiments. This seems to add further support to considering slightly larger bucket sizes than the two MTUs limit that was originally mentioned for services built on the EF PHB.

5. Conclusion

In this paper, we have carried out an experimental investigation of the relation that exists between (user level) video quality and various network configurations embodying a service based on the Diff-Serv EF PHB. The experiments considered different types of video clips, different encoding rates, different video servers, and evaluated the quality of received video streams transmitted over both local and wide area (QBone) testbeds for different settings of the token bucket associated with the "service" assigned to the stream. The assessment of the quality of the received video was performed using an objective quality measurement tool that provides accurate estimates of video quality, as users perceive it. The main focus was on the impact of the dropping actions performed by the policer on non-conformant packets.

One of the findings of the investigation was to confirm that frame loss itself is not necessarily an accurate measure of video quality. The evolution of the two is often decoupled, so that in some instances large increases in allocated rate do not translate into significant quality improvements, while in other cases, a small amount of additional bandwidth can yield a drastically better video quality. This conclusion does not appear to significantly depend on the type of video used, but the exact relation between video quality and frame loss (or token rate) does depend on the type of server and encoding used. In general, a token rate larger than the encoding used is needed in order to achieve high quality for the received video. The required margin above the encoding rate is where the dependency on the server and encoding used comes in.

Another interesting finding that emerged from our investigation is that a small increase of the token bucket depth used by the policer can translate into substantial improvements in the quality of the received video. Allowing such an increase should clearly be weighed against the larger burstiness it will allow and its associated impact on frame losses and delays in the network. Specifically, a larger token bucket means that larger EF bursts can now enter the network. This can in turn contribute to the accumulation of larger bursts as the EF traffic traverses multiple hops, e.g., [4]. However, given the magnitude of the observed improvements and the relatively small increase in token bucket depth that is required, this may be an option worth considering when building services that will use the EF PHB. Especially since increasing the potential burst size by one MTU (from two to three) is unlikely to contribute a significant increase in burstiness in the network, at least not for moderate EF loads.

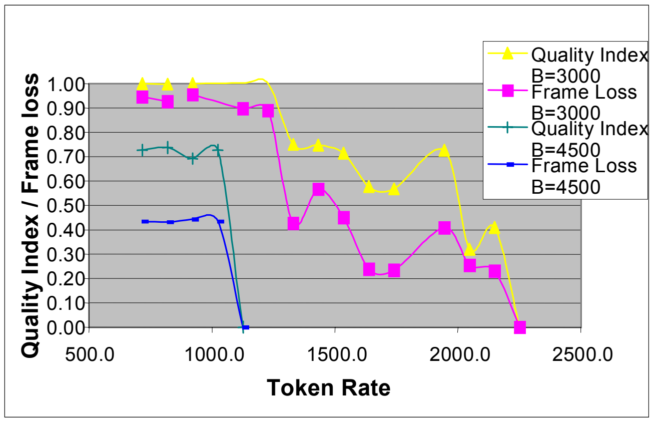


Figure 16. Local Testbed Experiments (*Lost* clip at 1Mbps) – Quality and Frame Loss vs Token Rate.

6. Acknowledgments

The authors would like to acknowledge the assistance of the many individuals who made the experiments over the Internet2 QBone possible. They include Matthew Davy, Indiana University, Shumon Huque, Dikram Kassabian, and Jon Looney, ISC, University of Pennsylvania, Ben Teitelbaum, Internet2, Tim Ward, Daniel Weaver, and Jeremy Weinberger, NorthWestern University, Ira Winston, University of Pennsylvania, Rick Wood, NorthWestern University, and Matthew Zekauskas, ANS. In addition, we would like to thank iCAIR and in particular Joel Mambretti, as well as IBM for providing us with access to the Video Charger™ video server that was used for streaming video

over the QBone, and IBM for donating most of the equipment used in the local testbed experiments.

7. References

[1] ANSI T1.801.03 – 1996, "American National Standard for Telecommunications - Digital Transport of One-Way Video Signals – Parameters for Objective Performance Assessment," American National Standards Institute.

[2] W. Ashmawi, R. Guerin, S. Wolf, and M. Pinson, "On the impact of policing and rate guarantees in Diff-Serv networks: A video streaming application perspective." Technical report, 2001, University of Pennsylvania, http://pender.ee.upenn.edu/~guerin.

[3] ATM Forum – "Traffic Management Specification Version 4.1," (AF-TM-0121.000), John Kenney, Ed., March 1999.

[4] J. C. R. Bennett, K. Benson, A. Charny, W. F. Courtney, and J.-Y. Le Boudec, "Delay Jitter Bounds and Packet Scale Rate Guarantee for Expedited Forwarding." Proceedings of INFOCOM'2001, Anchorage, Alaska, April 2001.

[5] D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, and S. Blake, "An Architecture for Differentiated Services." Internet Engineering Task Force, Request For Comments, RFC 2475 (Informational), December 1998.

[6] B. Carpenter, D. Kandlur, and J. Mambretti, "Experiments with Differentiated Services at iCAIR." Proceedings of the First Joint Internet2/DOE QoS Workshop, Houston, TX, February 2000.

[7] ITU-R Recommendation BT.500, "Methodology for subjective assessment of the quality of television pictures," Recommendations of the ITU, Radiocommunication Sector.

[8] ITU-R Recommendation BT.601-2, "Encoding parameters of digital television for studios," Recommendations of the ITU, Radiocommunication Sector.

[9] ITU-T Recommendation P.910, "Subjective video quality assessment methods for multimedia applications," Recommendations of the ITU, Telecommunications Standardization Sector.

[10] Cisco Committed Access Rate, http://www.cisco.com/warp/public/732/Tech/car/.

[11] T. Ferrari and P. Chimento "A Measurement-based Analysis of Expedited Forwarding PHB Mechanisms." Proceedings of IWQoS 2000, Pittsburgh, PA, June 2000.

[12] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured Forwarding PHB Group." Internet Engineering Task Force, Request For Comments, RFC 2597 (Standards Track), June 1999.

[13] J. Heinanen and R. Guerin, "A Single Rate Three Color Marker." Internet Engineering Task Force, Request For Comments, RFC 2697 (Informational), September 1999.

[14] J. Heinanen and R. Guerin, "A Two Rate Three Color Marker." Internet Engineering Task Force, Request For Comments, RFC 2698 (Informational), September 1999.

[15] M. Hemy, U. Hengartner, P. Steenkiste, T. Gross, "MPEG System Streams in Best-Effort Networks." Packet Video 99, Columbia University, New York, NY, April 1999.

[16] V. Jacobson, K. Nichols, and K. Poduri, "An Expedited Forwarding PHB." Internet Engineering Task Force, Request For Comments, RFC 2598 (Standards Track), June 1999.

[17] M. Knee, "The Picture Appraisal Rating (PAR) – a single-ended picture quality measure for MPEG-2." Proceedings of the International Broadcasting Convention, Amsterdam, The Netherlands, September 2000.

[18] The MPEG Home Page, http://www.cselt.it/mpeg/.

[19] K. Nichols, S. Blake, F. Baker, D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers." Request For Comments, RFC 2474 (Standards Track), December 1998.

[20] T. V. Lakshman, A. Ortega, and A. R. Reibman, "VBR Video: Trade-Offs and Potentials." Proc. IEEE, pp. 952-973, May 1998.

[21] J.-Y. Le Boudec and O. Verscheure, "Optimal Smoothing for Guaranteed Service." IEEE/ACM Trans. Networking, Vol. 8. No. 6, pp. 689-696, December 2000.

[22] V. Paxson, G. Almes, J. Mahdavi, and M. Mathis, "Framework for IP Performance Metrics." Internet Engineering Task Force, Request For Comments, RFC 2330 (Informational), May 1998.

[23] R. Rejaie, M. Handley, and D. Estrin, "Quality Adaptation for Congestion Controlled Video Playback Over the Internet." Proceedings of SIGCOMM'99, Cambridge, MA, August 1999.

[24] J. Rexford and D. Towsley, "Smoothing Variable Rate Video in and Internetwork." IEEE/ACM Trans. Networking, Vol. 7, No. 2, pp. 202-215, April 1999.

[25] B. Teitelbaum, Editor, "QBone Architecture (v1.0)." Document available at http://qbone.internet2.edu/.

[26] VQEG Home page, http://www-ext.crc.ca/vqeg/frames.html.

[27] ITU-T Recommendation J.143, "User requirements for objective perceptual video quality measurements in digital cable television," Recommendations of the ITU, Telecommunication Standardization Sector.

[28] The Windows Media Home page, http://www.microsoft.com/windows/windowsmedia/.

[29] S. Wolf and M. H. Pinson, "Spatial-Temporal Distortion Metrics for In-Service Quality Monitoring of Any Digital Video System." Proceedings of SPIE International Symposium on Voice, Video, and Data Communications, Boston, MA, September 11-22, 1999.

♣ Certain commercial equipment and software products are identified in this paper to ensure completeness and accuracy in describing the information presented. In no case does such identification imply recommendation or endorsement by the University of Pennsylvania or the National Telecommunications and Information Administration, nor does it imply that the equipment or software is necessarily the best available for this application.