Beyond Audio and Video: Multimedia Networking Support for Distributed, Immersive Virtual Environments[*]

Kevin Jeffay   Thomas Hudson   Mark Parris
Department of Computer Science
University of North Carolina at Chapel Hill
Chapel Hill, NC 27599-3175 USA
http://www.cs.unc.edu/Research/dirt   http://www.cs.unc.edu/Research/nano

Abstract: *Like interactive audio/video applications, distributed virtual environments (DVEs) require continuous, low-latency delivery of media. While end-system media adaptations and network-based forwarding services have been developed to support audio/video applications, it remains an open question whether these mechanisms can be either directly applied or adapted to realize the requirements of DVEs. We present the results of a study on the use of audio/video media adaptations and router-based active queue management (AQM) to support the data-flows generated by the UNC nanoManipulator — a DVE interface to a scanned-probe microscope. We present a delay-jitter management scheme used to support a haptic force-feedback tracking/pointing device used in the nanoManipulator and an AQM scheme based on buffer allocation in routers to reduce packet loss. The results of early experiments are promising and provide evidence that a sophisticated virtual environment interface can operate over the Internet to control a remote microscope in real-time.*

1. Introduction

The problem of managing streams of audio and video in a distributed system was one of the most popular and fundamental problems studied in computer systems research during the last ten years. In its essence, systems support for digital audio and video was network and operating system support for *continuous media flows*. These were streams of data generated by long-lived distributed applications that produced data objects periodically and had some degree of flexibility in their requirements for reliable delivery of data.

We are considering a generalization of the multimedia networking problem for audio and video flows, namely, the problem of supporting the continuous media flows generated by distributed virtual environment applications (DVEs). Our specific interest are DVEs arising from the increasing trend of scientists interacting with instruments such as microscopes, spectrometers, and medical imaging equipment through computer-based interfaces. For these systems, one can often use computing power to create novel interfaces such as virtual environment interfaces that enable new paradigms of instrument operation and data visualization. These systems are inherently distributed systems, often with the instrument, an interface generation computer, and special purpose interface devices such as haptic force-feedback devices and head-mounted displays distributed on a LAN in a single laboratory. This simple distribution naturally motivates a larger scale distribution of system components across the Internet to enable collaborative work and promote the sharing of expensive or special purpose equipment. The result is a distributed virtual laboratory consisting of a geographically separated collection of scientific instruments, computers, and scientists all interconnected via a computer network such as the Internet.

From a computer systems standpoint, the primary technical challenge is the realization of network quality-of-service (QoS) requirements for a compelling immersive virtual environment. While proposals for realizing QoS on the Internet are evolving, it remains unclear whether these proposals are necessary and/or sufficient for supporting applications such as DVEs. As has been the case with interactive audio/video applications, the central question is whether a network should provide end-to-end services with guarantees of QoS. On the one hand it has been argued that given the tremendous increases in network bandwidth, one can exploit the flexibility inherent in the way audio/video applications generate media to ameliorate the effects network congestion encountered in a network without guarantees of QoS. Indeed the prevalence of streaming media applications on the Internet today is an existence proof that techniques for best-effort delivery and adaptation of real-time media are capable of realizing the real-time transmission requirements for many applications. On the other hand, the limited use of more interactive applications such as audio/video conferencing suggests that the lack of alternate service models on the Internet may be impeding the realization of the advanced communication, collaboration, and entertainment environments envisioned for the Internet. Moreover, as the bandwidth consumed by real-time streaming applications rises, concern has been raised about the effect that these (UDP-based) applications may be having on the TCP-based applications (such as Web-browsing) that make up the majority of traffic on the Internet.

[*] Work supported by grants from the National Science Foundation (grants CDA-9624662, ITR-0082870, & ITR-0082866), NIH National Center for Research Resources Award RR02170 on Interactive Graphics for Molecular Studies and Microscopy, and grants from the North Carolina Networking Initiative and the Dell, Cisco, Cabletron, IBM, Sun Microsystems, and Intel corporations.

In: Proceedings of the 27th EUROMICRO Conference, Warsaw, Poland, September 2001, pages 300-307.

Our interests lie in determining (1) the extent to which existing best-effort media adaptation techniques developed for audio and video can be used to realize the performance requirements of a DVE and (2) whether network mechanisms proposed for advanced congestion control and differentiated services can be used to isolate non-real-time flows from real-time flows and provide a "better-than-best-effort" delivery service to the flows generated in DVE applications.

In this paper we summarize the results of some early experiments supporting a DVE for advanced microscopy. The system, the UNC nanoManipulator, is a virtual reality interface to a scanned probe microscope (SPM) that allows chemists, biologists, and physicists to "see" the surface of a material sample such as a virus particle or a carbon nanotube at nanometer scale, and "feel" the properties of the surface through the use of force-feedback. This is accomplished by integrating an SPM with high-performance 3D graphics and a haptic force-feedback/tracking device.

We are currently developing a version of the nanoManipulator that allows the operation of a remote SPM in real-time. The approach is to use well-understood media adaptations developed for audio/video flows and apply these adaptations to the flows generated by the nanoManipulator to deal with common IP network pathologies such as delay-jitter, packet loss, and out-of-order or duplicate arrivals. In parallel with this effort we are also investigating extensions to router-based active queue management schemes to protect classes of traffic from the effects of one another and to provide a limited form of QoS to designated traffic classes.

The remainder of this paper is organized as follows. Section 2 describes the nanoManipulator system in greater detail and presents its requirements for real-time media transmission. Section 3 reviews the best effort multimedia networking problem for audio/video applications and discusses our use of audio/video media adaptation to ameliorate the effects of delay-jitter on nanoManipulator flows. Section 4 discusses a router-based differentiated services approach we use to protect TCP applications from the potential ill-effects of continuous media applications and to provide a better-than-best-effort delivery service to applications such as the nanoManipulator. We conclude in Section 6 with a discussion of plans for future investigations and evaluations.

2. The UNC nanoManipulator

Scanned-probe microscopes, such as the atomic-force microscope (AFM), allow the investigation and manipulation of surfaces down to the atomic scale. An AFM is capable of positioning a tip very precisely (*e.g.*, within a fraction of an atomic diameter) over a surface in a wide variety of environments, including ambient, ultrahigh vacuum, and under water. It is capable of resolving individual atoms on crystalline surfaces, and molecules, proteins, and viruses under physiological conditions. The AFM tip can provide quantitative data on a wide range of sample features including the surface topography, friction, adhesion, temperature, and compliance. Most important, the surface can be modified through the deposition of material or through mechanical tip/sample interaction by machining the surface or manipulating surface-bound objects.

An AFM generates a three-dimensional dataset measuring the height of a surface over a two-dimensional region. Traditional AFM interfaces only display this data in two dimensions as a rectangular area whose color at a point indicates the height of the surface at the point. Three-dimensional display has long been used as a post-process to help scientists understand their data after they have completed the experiment. By displaying the data from the AFM three-dimensionally as the experiment is in progress, scientists have a better grasp of the experiment as it happens and are able to change their plans to suit their observations.

The nanoManipulator integrates 3D graphics and force feedback to give a virtual environment interface to SPMs. Force feedback is used to convey information about the surface (*e.g.*, topography, compliance, *etc.*) to the user's sense of touch. A force-feedback device measures the position and orientation of a stylus held by the user. Given a geometric model of the surface obtained from the microscope, the system computes the force that should be felt by the user at the measured position, and uses motors to display that force to the user by mechanically moving the stylus. Thus, although an SPM tip is only a few microns across, it can be felt and made to respond as if it were a pen in the user's hand. In this manner the nanoManipulator gives the scientist virtual telepresence on the surface, scaled by a factor of a million to one, blowing a virus up to the size of an apple, or a strand of DNA up to the size of a piece of spaghetti. A head-tracked stereo display presents the 3D surface floating in space within arm's reach. The force-feedback device allows the scientist to actually feel surface contours and to manipulate objects on the surface, such as the tobacco-mosaic virus particles in Figure 1. This direct, natural interface to SPMs has enabled new forms of experimentation with the instrument that could not have been otherwise performed.

The nanoManipulator application has two basic modes of operation: scanning and point mode. Scanning is used to provide a single, consistent representation of the topography of the entire surface that is rendered with three-dimensional graphics and displayed to the user. In scanning mode, the microscope tip moves across the surface in a raster pattern at

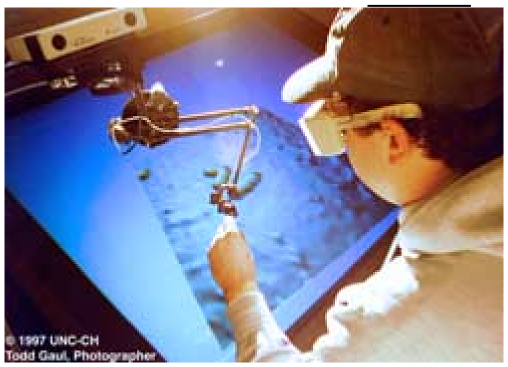

Figure 1: The nanoManipulator interface. Shown are stereo, head-tracked shutter glasses and haptic force-feedback device.

a constant speed, taking quick measurements of surface height and other datasets at regular intervals. Scanning is the standard mode in which SPMs operate when imaging a surface. The nanoManipulator augments the standard SPM by allowing manipulation of the surface. In point mode, the user's hand is coupled to the position of the microscope's tip through the haptic force feedback device. The nanoManipulator tracks the user's hand and tells the microscope to move its tip to a specific point corresponding to the user's current position on the virtual surface. When the tip arrives, it responds with measurements of the sample at that point.

The current deployment of the nanoManipulator couples a Windows NT workstation with 3D graphics card and a force-feedback controller to a microscope controller via a dedicated LAN switch. The dedicated network is used by the graphics computer to send and receive SPM data and commands and position/force descriptions in real-time. The nanoManipulator application is implemented as three processes running on two NT workstations. One machine is directly connected to the SPM which it controls (see Figure 2). The other machine is a two-processor machine with one CPU devoted to a hard-real-time process controlling the force-feedback device. The other CPU runs the main nanoManipulator application, including graphics, user interface, and network communications. Depending on the use of the system, other interface devices and scientific instruments may be attached.

The real-time communications requirements of the nanoManipulator system depend on the mode of system operation. When the system is in scan mode, the microscope controller sends data asynchronously to the graphics display as the tip rasters across the sample surface. This data updates each part of the displayed surface image as the scan data arrives. Because the raster involves a physical interaction, exact sampling rates vary widely with the sample and sampling conditions. Typical rates are 300 sixteen-byte measurements per second. Each second a 4.8 Kbyte message is sent containing 300 measurements resulting in a 38 kbps flow.

When the user is in point mode the microscope tip follows the trajectory of the user's hand as tracked by the force-feedback device. The surface is scanned at these locations and force-feedback tells the user where the surface is and how it is changing. If the nanoManipulator is running at a frame rate of 30 Hz, it will send a 28-byte command to the microscope every frame, requiring 16 kbps of bandwidth. The microscope's response is 56 bytes. If the microscope can keep up with commands (again, a function of the sample and environmental conditions), it will generate a stream of responses requiring 24 kbps of bandwidth. In point mode, system latency becomes critical. Whereas virtual reality applications have historically labored under several hundred milliseconds of latency (often with reduced effectiveness), when the user is interacting with and controlling physical objects latency is more critical. Teleoperation control loops generally become unstable with total system latencies of 100 *ms* [1]. This is problematic because microscope and application latencies leave little room for network latency. Thus a major focus of our work is reducing or ameliorating the effects of network latency so the nanoManipulator can function in point mode over wide-area networks.

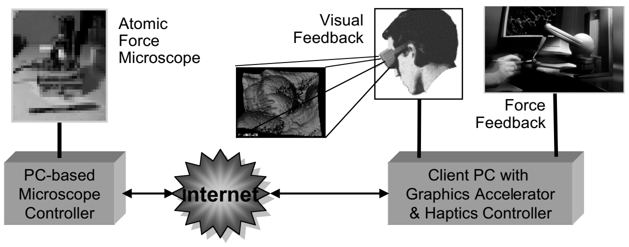
Atomic Force Microscope
Visual Feedback
Force Feedback
PC-based Microscope Controller
Internet
Client PC with Graphics Accelerator & Haptics Controller

Figure 2: NanoManipulator high-level system structure.

3. Best-Effort Multimedia Networking for DVEs

Broadly speaking, research in real-time communications has proceeded along two parallel fronts. Most visible is the work on new network service models for the Internet. The existing best-effort packet forwarding model has been proposed to be extended to include guarantees of QoS including guarantees of throughput and delay. This so-called Integrated Services architecture for the Internet ("*intserv*") has proposed a "call-setup" procedure wherein resources are reserved along the path from receiver(s) to sender and network switches maintain per-connection state and allocate link bandwidth in accordance with application requirements [3]. Because of concerns about the complexity of the interserv architecture, a second, simpler scheme, the Differentiated Services architecture ("*diffserv*"), has become the dominant architecture of study. Under diffserv the emphasis is less on end-to-end services and more on per hop forwarding behaviors such as priority forwarding or low-delay forwarding [12].

The second major focus of research activity has been in the development of media adaptation mechanisms for best-effort multimedia networking. Here, the guiding principle is the observation that in the absence of guaranteed-service models for the Internet, applications requiring real-time transmission must sense the state of network congestion and adapt the media streams they inject into the network to maximize the probability of in-time delivery. The fundamental premise is that (1) network bandwidth, when averaged over sufficiently long intervals, is sufficient for some minimally acceptable operation of the application, and (2) applications have sufficient flexibility inherent in the manner in which they generate media streams that they can adapt to the natural changes in network bandwidth availability.

In this work we are interesting in testing the best-effort multimedia networking assumptions in the context of a DVE application such as the nanoManipulator. As explained in more detail below, we are primarily concerned with ameliorating the effects of high transmission delay and delay-jitter. Packet loss is also an issue and Section 4 describes a diffserv approach to reducing loss.

3.1 The effect of delay and delay-jitter on haptic force feedback display

Haptic force feedback display over a distance is difficult. The traditional approach to force feedback and teleoperation sends force and position data between the controller and the force feedback device at 500 to 1,000 Hz [4]. This is only possible over a synchronous, dedicated, short-haul network or a bus. When run over a shared or long-distance network, transmitting raw data is highly sensitive to delay. Instead of presenting a solid, sharp-edged, stable surface, delayed force feedback results in soft, mushy surfaces, making the use of haptics ineffective or unstable [1].

The most common approach used by virtual reality applications to control force feedback without requiring high update rates is the plane approximation to a surface [9]. In the nanoManipulator's implementation of plane approximation, each new measurement from the microscope is combined with the previous measurement to determine a plane that approximates the shape of the surface near the two measurements. The user feels this plane until the next measurement arrives and a new plane is determined (see Figure 3). If the two planes are far apart, or the user's hand is "inside" the new plane, the force feedback device interpolates from the old plane to the new to avoid sharp discontinuities. Although the force feedback system is reading the user's position and changing the force it exerts at 1,000 Hz, the plane approximation only needs to be updated at 20 to 30 Hz to simulate stiff, sharp-edged surfaces. This update rate is comparable to full-motion video rates and is within the realm of sustainability over the Internet. However, as illustrated in Figure 4, the plane approximation is sensitive to delay. With moderately high delay, a plane approximation displays incorrect and inconsistent surface topography to the user. Since the plane is a first-order approximation to the shape of the surface near the location of a measurement, if the user moves far from the point where the measurement was taken, the approximation is invalid.

We are investigating alternate representations that have better delay tolerance. The first of these is the "warped" plane approximation, which uses a user-interface compensation to mask delay. The warped plane approximation measures network latency and determines the displacement of the user's hand that occurred while the current tip position data was traveling over the network. This same displacement is then applied to the current plane approximation. As illustrated in Figure 5, although the warped plane approximation tolerates delay much better than the simple plane approximation, it is sensitive to delay-jitter, which can distort the shape of the surface being felt. To ameliorate the effects of delay-jitter, we turn to adaptive buffering schemes developed for interactive audio/video streaming applications.

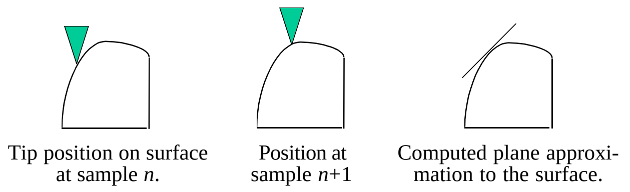
Tip position on surface at sample *n*.
Position at sample *n*+1
Computed plane approximation to the surface.

Figure 3: Local plane approximation example.

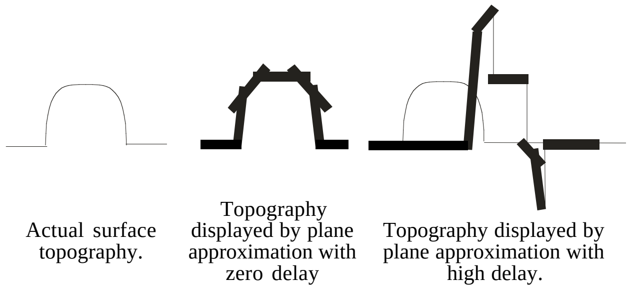
Actual surface topography.
Topography displayed by plane approximation with zero delay
Topography displayed by plane approximation with high delay.

Figure 4: Local plane approximation in the face of high delay. (Motion of microscope tip is from left to right.)

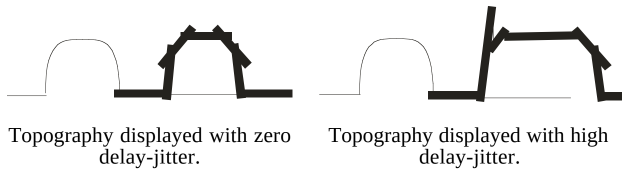
Topography displayed with zero delay-jitter.
Topography displayed with high delay-jitter.

Figure 5: Topography displayed by warped plane approximation in the face of delay-jitter. (Motion is left to right.)

3.2 Dealing with delay and delay-jitter

High transmission delay and delay-jitter are caused by the build-up of queues in the network. Display processes for periodically generated media require that samples arrive at the receiver at regular (precise) intervals in order to avoid "gaps" in the playout. When queues grow and shrink in the network the delay experienced by packets changes accordingly. The obvious solution to ameliorating the effects of delay-jitter is to introduce a buffer at the receiver to smooth the arrival process. If the depth of the buffer is equal to the largest end-to-end delay likely to be encountered divided by the sample period, then playout will be gap-free if the buffer is filled prior to the initiation of the playout process. However, there is a clear trade-off. While a deep buffer guarantees gap-free playout, it also has the effect of delaying each sample as if it actually encountered the worst case delay in the network. If the worst case delay is rare then the acquisition-to-display latency of the media may be unnecessarily high. A shallow (or no) buffer may result in a few gaps in the playout process but ensures that receiver-side latency is minimal.

Given that in general it is not possible to determine the worst case end-to-end delay a packet will encounter, multimedia applications commonly use a buffer management algorithm to dynamically set the depth of the playout buffer. These so-called "elastic" buffering schemes sense the level of congestion (queuing) in the network and set the depth of the buffer accordingly. For example, for the nanoManipulator, we adopt a receiver playout buffer management scheme called *queue monitoring* [11]. Queue monitoring is based on the observation that over a given interval, if the occupancy of a receiver's buffer remains constant, then the instantaneous arrival rate is the same as the playout rate (independent of the actual number of media samples enqueued). This implies that with respect to the duration of a media sample, the level of delay-jitter is "0." If the occupancy of a receiver's buffer changes by ±1 elements over time, then the level of delay-jitter is equal to the duration of 1 media sample. More generally, if the occupancy of a receiver's buffer changes by ±*k* elements over time, then the level of delay-jitter is equal to the duration of *k* media samples.

A second key observation is that if the occupancy of a receiver's buffer changes by only ±*k* elements over time, then a buffer of depth *k* is sufficient to smooth the delay-jitter currently being experienced. In particular, in this situation there is no utility in maintaining a buffer of depth greater than *k* elements as doing so only increases the ultimate acquisition-to-display latency of samples. Combined, these two observations form the basis of a buffer management algorithm. One can record the variation in buffer occupancy over time and use this as a measure of the current level of delay-jitter. In addition, one can use this measure to dynamically set the depth of the buffer.

The nanoManipulator uses queue monitoring at the client PC to manage the microscope tip position sample queue. A count and a time threshold are associated with each position in the queue. The count for position *k* in the queue represents the most recent duration (in units of packet interarrival times) that the queue continuously contained at least *k* elements. When a packet arrives at the receiver and the playout queue currently contains *n* items, the counts for positions 1 through *n* are incremented. If the count for any queue position *k* between 1 and *n* exceeds its threshold, then the arriving packet is dropped. This indicates that since the queue has contained at least *k* elements for a sufficiently long time the current level of jitter is believed to be less than *k* sample inter-arrival times. By dropping a packet the acquisition-to-display latency of media samples arriving in the future is reduced by reducing the time they will spend in the receiver's queue. However, dropping a packet introduces a gap into the playout of the stream. Thus we are explicitly trading off gap-free playout for low latency playout — a fundamental trade-off for continuously generated media. For media that is continuously generated and displayed at a constant rate, the only way latency can be reduced is to "skip" the display of a media sample.

Queue Monitoring assumes that a media stream has a source and a sink that continuously produce and consume packets at equal rates. This allows it to quantize time, dealing only with buffer occupancies at regular polling intervals. While this works for audio/video display processes, in the nanoManipulator, the data generation and data display processes do not have equal rates: the user sends requests to the microscope at anywhere from 10 to 60 Hz, while the microscope's response rate varies widely because of device delays. We use an *updateable queue* [8] at the microscope to scale back the rate of requests received to a rate that the microscope can process. An updateable queue is a FIFO buffer that associates a key with every entry and drops any entry in the buffer with a key *k* before inserting a new entry with key *k*.

The user process needs to attempt to reconstruct the timing at which the microscope took samples. This requires extending Queue Monitoring to deal explicitly with sub-frame timing, making the implementation more complex and less deterministic. Instead of playing out one sample from the queue every frame, the receiving process must explicitly track the mean interarrival time from the microscope and use that as the effective inter-frame time for queue playout. In tracking time like this, the efficacy of the queue monitor depends on the accuracy of our interarrival time computation, but so long as the computation is reasonably accurate we should be able to compensate for device jitter (variation in the time it takes the microscope to process a request) in addition to network jitter.

3.3 Experimental Results

A number of experiments were conducted to evaluate delay-jitter adaptation using the network infrastructure described in [5]. Briefly, a controlled laboratory network was used to model an enterprise or campus network having a single 10 Mbps link to an upstream Internet service provider (ISP). One endpoint of the nanoManipulator system was placed on the "campus" side of the network and the other endpoint on the "ISP" side of the network. This system shares the ISP link with a large collection (thousands) of simulated Web browsers. Web browsers on the campus generate requests for Web servers out in the ISP's network which return responses that are sampled from an empirical model of Web traffic [5]. By appropriately configuring the number of Web browsers and servers, the level of congestion on the campus-ISP link can be closely controlled. NanoManipulator traffic competes with the web traffic on this link and hence is subject to varying (but predictable and reproducible) levels of congestion. The experiments reported below are the results of operating the nanoManipulator in an environment where the Web browsers and servers kept utilization of the 10 Mbps campus-ISP link at 98% of capacity.

With no adaptation, end-to-end (scope to haptics) system latency is in the range 90-100 *ms*. This is significantly more than the 40-60 *ms* delay our users are used to when working over the UNC campus network. The teleoperation literature leads us to expect reduced effectiveness and possible instability in the 90-100 *ms* region [1] (*e.g.*, the behavior illustrated in Figure 4). We thus use the more delay-tolerant warped plane surface representation. However, to get the best results, delay-jitter must also be addressed.

Table 1 shows the results of the use of queue monitoring to smooth delay-jitter in the nanoManipulator system. Three queue monitoring settings were tested corresponding to "low," "medium," and "high" levels of jitter tolerance. For each setting the loss rate (a measure of network congestion and a value that should remain constant between experiments as loss is not addressed here), the rate at which packets are dropped by the queue monitoring algorithm (in an effort to reduce playout latency), the resulting gap-rate in the playout of this data in the application, and the end-to-end latency were measured.

With no delay-jitter adaptation, jitter results in a gap rate that gives unacceptable application performance. Even with small parameter value settings for queue monitoring, the queue hardly ever empties. With a threshold of 30 packet arrival times for a queue of length 2 or greater (*i.e.*, the playout queue must have at least 2 elements for 30 consecutive sample arrivals before a sample is discarded in an effort to reduce playout latency), we see a decrease in the drop-rate to 0.6%, and a gap rate just marginally above the loss-rate. (Without error control lost samples always result in a gap in the playout, hence the loss-rate is a lower bound for the gap-rate.) At higher levels of jitter tolerance (*i.e.*, queue monitoring settings that require longer durations with jitter that does not exceed 2 sample interarrival times), the drop-rate continues to decrease, and the gap-rate falls to match the loss-rate. Although we measure a slight increase in latency with queue monitoring over the non-queue monitoring system, the increase is tolerable and worth the decrease in the drop-rate.

Table 1: NanoManipulator performance with queue monitoring. Average packet rate is 28 packets/second.

QM Parameters	Packet Loss (per second)	Drop-Rate (per second)	Gap-Rate (per second)	Latency (*ms*)
No buffering	2.8 (9.7%)	3.4 (11.7%)	6.2 (21.5%)	89
(30, 2)	2.8 (10%)	0.18 (0.6%)	3.0 (10.6%)	94
(150, 2)	2.8 (9.7%)	0.06 (0.02%)	2.8 (9.7%)	96
(3600, 2)	2.8 (9.5%)	.003 (0.001%)	2.8 (9.5%)	91

4. Differentiated Services for DVEs

Queue monitoring is an end-system adaptation that ameliorates the effects of delay-jitter. However, in addition to delay-jitter, DVEs must also deal with packet loss. While end-system adaptations such as forward error correction are possible here, we are investigating a network-based approach of providing differentiated levels of services to applications. To motivate this approach one must consider the potential impact that DVEs and continuous media (CM) applications in general have on the network as a whole.

4.1 The tension between TCP and non-responsive UDP applications

From a network management perspective the requirement of DVEs for continuous transmission puts these applications at odds with current and proposed Internet network management practices. Whereas other network applications such as Web browsers rely on the underlying network transport protocol, most notably TCP, for congestion control and avoidance, real-time applications eschew TCP in favor of UDP. The reliance on UDP by CM applications typically results from a conscious trade-off of reliable, in-order data delivery for the ability to exercise control over the reaction to packet loss and variable delay. This gives rise to a potential problem as UDP applications have the potential to starve TCP connections of bandwidth. During congestion, queues of packets build up in routers and switches. When the queues overflow packets are dropped. TCP connections detect packet loss in the network, treat it as a signal of congestion, and react by reducing their transmission rate. A high-bandwidth UDP CM application that does not respond to packet loss as an indicator of congestion will continue transmitting data into queues that are already overflowing. TCP connections that share these queues (and thus are having their own packets dropped) will react to packet loss and reduce their transmissions. If the buffer space in routers previously consumed by these TCP connections are now consumed by packets from "non-responsive" UDP applications, then the situation spirals. Router queues remain full and TCP connections, although transmitting at lower rates, still lose packets and therefore continue to reduce their transmission rates until they effectively stop transmitting.

Thus tension exists between the current practice of relying on end-system transport protocols to limit the rate at which they inject traffic into the network and the desire of many CM applications to transmit data continuously at a uniform rate. While TCP's reaction to packet loss is unsuitable for applications requiring a uniform transmission rate, it is essential for the health of the Internet. It ensures a crude form of fair sharing of link capacity between TCP connections and ensures that a pathological condition called congestion collapse does not occur. The presence of CM applications on the Internet therefore potentially threatens the stability of the Internet and has even led to proposals that routers attempt to identify non-responsive flows and drop their packets during times of congestion [7].

Our research here attempts to develop network resource-management mechanisms that strike a balance between the conflicting requirements of continuous transmission and congestion control. We are investigating a router-based approach to meeting these requirements that allocates router queue capacity to traffic classes. The practice of manipulating router queues to bias the performance of network connections is known as *active queue management* (AQM). We are developing lightweight AQM schemes for routers that protect TCP connections from starvation by non-responsive UDP applications, and reserve router capacity for UDP applications that require continuous transmission. The goal is not to provide guarantees of performance, but rather to provide a so-called "better-than-best-effort" forwarding service for CM flows that results in better throughput and latency for these CM flows in times of congestion. The challenge is to provide a better-than-best-effort service without unnecessarily degrading TCP performance.

4.2 Active queue management

The default best-effort packet-forwarding service of the Internet Protocol (IP) is typically implemented in routers with a single, fixed-size, FIFO queue shared by all flows. The queue exists simply to provide capacity for tolerating variability ("bursts") in the arrival rate of packets. Router implementations using a fixed size FIFO queue typically drop any packet that arrives at an already full queue. This "drop-tail" packet discarding, behavior creates two problems [2]. First, in some situations flows can be "locked out," a condition in which a small subset of the flows sharing the outbound link monopolize the queue during periods of congestion. Flows generating packets at a high rate fill up the queue such that packets from flows generating packets at substantially lower rates have a higher probability of arriving at the queue when it is full and being discarded. This loss has the effect of providing feedback to the "wrong" sources. The locked out flows receive feedback that they should reduce their transmission rate while the "aggressive," high-packet-rate flows receive less feedback and do not reduce their rate appropriately.

The second problem is that latency is increased for all flows when the queue remains full or nearly full. Simply making the queue shorter will decrease the latency but negates the possibility of accommodating brief bursts of traffic without dropping packets. For TCP flows, this problem can be addressed by selectively dropping packets before the queue fills. Since TCP flows decrease the load they generate in response to drops, dropping packets "early" should have the effect of eventually slowing or ceasing the growth of queues.

The Internet community is promoting a specific form of AQM for routers known as *random early detection*, or RED [2, 6]. Under RED, packets will randomly be dropped from a non-full queue with the probability of a packet being dropped at any given time being a function of the average length of the queue in the recent past. RED avoids lockout by dropping packets uniformly from all flows. By providing early notification of congestion, RED has been shown to result in shorter average queue lengths in routers.

Although RED improves the network performance of TCP flows, they are still susceptible to starvation in the presence of non-responsive UDP flows. The starvation phenomenon has led to the call for routers to explicitly recognize non-responsive flows and to place them in a "penalty box" [7]. Non-responsive flows would be penalized by some draconian action such as having all their packets dropped during times of congestion. While such proposals are arguably necessary to protect TCP from non-responsive flows and to provide an incentive for applications to respond to packet loss as an indicator of congestion, the penalty box concept does not bode well for CM applications that, while potentially adaptive, do not or can not adapt in a manner that mimics TCP's response to congestion.

We are experimenting with extensions to RED that will isolate TCP flows from non-TCP flows while at the same time ensuring that CM applications receive a tunable share of network resources. TCP flows are isolated from other flows by constraining the average number of non-TCP packets that may reside simultaneously in a router's queue. Classes of non-TCP traffic are also isolated from one another. This is done by marking packets of designated CM streams before they reach the router so that they can be classified appropriately. This marking is accomplished either by end systems or network administrators as part of a larger architecture for differentiated services on the Internet [12]. Once flows are tagged, a router will maintain simple queue statistics for a limited number of classes of traffic. For simplicity, in the following we assume three traffic classes: TCP, marked non-TCP, and all others. The throughput of traffic classes will be constrained during times of congestion by limiting the average number of packets each traffic class can have enqueued in the router. When a packet arrives at the router, it is classified into one of the three traffic classes. TCP packets are subject to the RED algorithm. For the non-TCP traffic classes, a weighted average number of enqueued packets from each class is maintained. When a non-TCP packet arrives, the weighted average for the appropriate class is updated and compared against a maximum threshold for the class. If the class average exceeds the threshold, the incoming packet is dropped, otherwise it is enqueued. Although packets are classified, there is still only one queue of packets, shared by all traffic classes, per outbound link in the router, and all packets are enqueued and dequeued in a FIFO manner.

We call our scheme *class-based thresholds* (CBT) [10]. CBT provides isolation between traffic classes by maintaining separate thresholds for each class. These thresholds effectively allocate each class a portion of the queue's capacity and ensure this capacity is available to the class independent of the transmission rates of other classes. This share of queue capacity corresponds to a share of the outbound link's capacity. Thus by tuning the queue allocation one can control the link bandwidth allocated to a traffic class.

To demonstrate the effectiveness of CBT, we empirically compare the performance of CBT to simple FIFO (drop-tail) queuing and RED using the same experimental facility described in Section 3. The router between the campus and the ISP implements CBT and supports the three classes of traffic described above. The TCP class consists of a collection of simulated Web connections capable of consuming approximately 98% of the 10 Mbps bottleneck link connecting the campus to the ISP. The marked non-TCP class is a collection of simulated videoconferencing applications and a nanoManipulator that combined generate approximately 200 kbps of marked UDP traffic. The third "other" traffic class contains a single high-bandwidth bulk-transfer UDP application.

Experiments were run using FIFO, RED, and CBT queuing at the router. Table 2 summarizes the results of the experiments. Not surprisingly, FIFO queuing results in the worst performance for all flows. The high-bandwidth, unresponsive UDP application causes the network to become saturated and drop packets at a high rate. The TCP connections react to the loss by reducing their transmission rates and consuming fewer buffers in the router. The UDP bulk-transfer also sees a high rate of packet loss but nonetheless does not adapt its transmissions. Because of this, under FIFO and RED, the buffers in the router remain full and all network connections continue to experience packet loss. TCP throughput plummets from a high of 1,000 kBps achieved in the absence of the bulk-transfer application to 200 kBps as the TCP connections are starved of bandwidth. CBT also shows a decrease in TCP throughput, but the decrease is on the order of 10-20% because CBT constrains the amount of bandwidth that the unresponsive flow can consume.

In addition to protecting TCP, CBT also isolates the marked non-TCP CM flows from the non-responsive flows in the "other" class. This can be seen from the drop-rates for marked CM packets shown in Table 2. Both FIFO and RED produced high drop-rates for CM flows while CBT produced drop-rates of just over 1% — closely approximating the performance of a more complex packet scheduling scheme that provides perfect isolation between traffic classes. In particular, CBT results in a drop rate for marked CM flows that is likely to be tolerated by CM applications.

Table 2 also compares the average end-to-end network latency of marked CM packets under each scheme. Latency is highest under FIFO queuing because the queue is consistently full. RED and CBT maintain shorter average queues, and hence yield significantly lower average latencies.

Table 2: Drop rates and network latency for tagged continuous media traffic and the impact on TCP traffic.

Queue management scheme	Packet drop-rate for marked flows	Average latency of marked flows	Ave TCP throughput
FIFO	32.4%	63.2 ms	200 kBps
RED	30.0%	26.2 ms	300 kBps
CBT	1.3%	28.4 ms	790 kBps

5. Summary

The UNC nanoManipulator is a state-of-the-art virtual environment interface for advanced microscopy that is being used by scientists as a tool for basic research in the material and biological sciences. Like interactive audio/video applications, to be effective the nanoManipulator requires quasi-real-time transmission of data. In this study we sought to investigate the use of end-system media adaptations originally developed for audio and video applications to ameliorate the effects network of congestion on the data-flows generated by the nanoManipulator. In addition we used a router-based differentiated services mechanism based on explicit buffer allocation to protect TCP applications from DVE and other (desirable) continuous media applications that do not react to packet loss as an indicator of network congestion.

The current work has focused on enabling direct control of a remote microscope tip via a haptic force-feedback device. The method of displaying surface information to the user is highly sensitive to delay-jitter and packet loss. Early experiments show that the queue monitoring delay-jitter adaptation can satisfactorily ameliorate the effects of delay-jitter. Moreover, when deployed on a network that supports packet markings for differentiated services, a simple active router-queue management scheme that controls the buffer occupancies of traffic classes can provide both a better-than-best-effort forwarding service to DVE and other continuous media flows and protect TCP flows from the potential ill-effects of the non-responsive nature of continuous media flows.

While these experiments show the promise of differentiated services and end-system adaptation for realizing remote microscopy, the problems of high-delay due to the store-and-forward implementation of today's Internet are causes for concern. Effective solutions here must consider trade-offs among (*a*) computer-human interfaces, (*b*) network infrastructure for integrated/differentiated services, and (*c*) costs of replicating special purpose hardware. The challenge is to understand which network service models, application system structures, and user interaction models are required to achieve useful, cost-effective results in a distributed laboratory given the constraints introduced by the distribution of system elements, network bandwidth and latency, and the desired interactions with the instrument. This problem is the subject of our future investigations.

6. References

[1] Anderson & Spong, "Bilateral Control of Teleoperators with Time Delay," IEEE Transactions on Automatic Control, v34, n5 (May 1989) pp. 494-501.

[2] Braden, Clark, Crowcroft, Davie, Deering, Estrin, Floyd, Jacobson, Minshall, Partridge, Peterson, Ramakrishnan, Shenker, Wroclawski, & Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet," *Internet draft, work in progress*, 1998.

[3] Braden, Clark, & Shenker, "Integrated Services in the Internet Architecture: an Overview," IETF RFC-1633, July 1994.

[4] Burdea, *Force and Touch Feedback for Virtual Reality*, John Wiley & Sons, New York, 1996.

[5] Christiansen, Jeffay, Ott, & Smith, "Tuning RED for Web Traffic," Proceedings of ACM SIGCOMM 2000, Stockholm, Sweden, August-September 2000, pp. 139-150.

[6] Floyd & Jacobson, "Random Early Detection gateways for Congestion Avoidance," *IEEE/ACM Trans. on Networking*, v1, n4, August 1993, p. 397-413.

[7] Floyd & Fall, "Promoting the Use of End-to-End Congestion Control in the Internet," *IEEE/ACM Transactions on Networking*, August 1999.

[8] Kessler & Hodges, "A Network Communication Protocol for Distributed Virtual Environment Systems," *Proceedings of Virtual Reality Annual International Symposium '96*, March 1996, pp. 214-221.

[9] Mark, Randolph, Finch, Van Verth & Taylor, "Adding Force Feedback to Graphics Systems: Issues and Solutions," *Proceedings of Siggraph 96*, New Orleans, LA, August, 1996, pp. 447-452.

[10] Parris, Jeffay, & Smith, "Lightweight Active Router-Queue Management for Multimedia Networking," *Multimedia Computing & Networking 1999*, SPIE Proceedings Series, Vol. 3020, San Jose, CA, January 1999, pp. 162-174.

[11] Stone & Jeffay, "An Empirical Study of Delay Jitter Management Policies," *Multimedia Systems*, v2, n6 (January 1995) pp. 267-279.

[12] *http://www.ietf.org/html.charters/diffserv-charter.html*