Adaptive Networking for Tele-Immersion

Jason Leigh+, Oliver Yu, Dan Schonfeld, Rashid Ansari,
Eric He, Atul Nayak, Jinghua Ge, Naveen Krishnaprasad,
Kyoung Park, Yong-joo Cho, Liujia Hu, Ray Fang,
Alan Verlo, Linda Winkler, Thomas A. DeFanti

+spiff@evl.uic.edu

Electronic Visualization Laboratory **(EVL)**
Dept EECS, MC 154, 1120 SEO, 851 S. Morgan St.

University of Illinois at Chicago
Chicago, IL 60607, USA

Abstract. Tele-Immersive applications possess an unusually broad range of networking requirements. As high-speed and Quality of Service-enabled networks emerge, it will becoming more difficult for developers of Tele-Immersion applications, and networked applications in general, to take advantage of these enhanced services. This paper proposes an adaptive networking framework to ultimately allow applications to optimize their network utilization in pace with advances in networking services. In working toward this goal, this paper will present a number of networking techniques for improving performance in tele-immersive applications and examines whether the Differentiated Services mechanism for network Quality of Service is suitable for Tele-Immersion.

1   Introduction

Tele-Immersion is the integration of collaborative virtual reality (VR) with audio and video conferencing in the context of data-mining and significant computation. The ultimate goal of Tele-Immersion is not only to reproduce a real face-to-face meeting in every detail, but to provide the "next generation" interface for collaborators, world-wide, to work together in a virtual environment that is seamlessly enhanced by computation and large databases. When participants are Tele-Immersed, they are able to see and interact with each other in a shared virtual environment. They are able to query and visualize data stores and steer complex scientific and engineering simulations[1].

One of the challenges of Tele-Immersion is that it poses diverse requirements of the underlying networks (Figure 1). For example, to convey audio and gestures of virtual participants (avatars,) low network latency is required; to distribute state updates, low latency but reliable data transmission is preferred; and to distribute data sets high-speed bulk data transfer is needed.

	Estimated bandwidth (bps)	Burstiness	Latency Sensitive	Jitter Sensitive	Error Sensitive
Avatar	6K x N (at 15fps)	Constant	y	y	n
Audio Streaming	64K x N (at 8kHz)	Brief	y	y	n
Video Streaming	10M (2-way)	Constant	y	y	y&n
Pre-recorded Stream Playback	Variable	Constant	y	n	y&n
Control Data	7K x N	Brief	y&n	y&n	y&n
Bulk Data	Variable	Sustained burst	n	n	y

Figure 1: Network characteristics of typical Tele-Immersion data streams. N refers to N collaborators. The 6K estimate for avatar representation assumes position and orientation data for a head tracker and a wand tracker. Pre-recorded streams refer to previously recorded VR streams that are replayed for viewing by multiple participants as in collaborative training simulations. Video streaming assumes low-latency NTSC resolution motion Jpeg. Bulk data refers to 3D models or large data sets.

In this paper, we will present our most recent work in exploiting advanced networking techniques to optimize data distribution in Tele-Immersion. We will describe our experiences in using Quality-of-Service-enabled high-speed networks for supporting Tele-Immersion. We will structure this work by proposing an adaptive networking framework to allow application developers to map their data distribution requirements to suitable networking services. We believe that as networking technology becomes more complex, application developers will have to rely increasingly on intelligent adaptive systems to make decisions on how to optimally distribute their data over them. The work discussed in this paper serves as a starting point toward that ultimate goal.

2   Adaptive Networking for Tele-Immersion

We propose an intelligent adaptive networking system (Figure 2) consisting of a Strategy Selector, Adaptive Controller and three supporting services: a Resource Monitor, a Quality of Service (QoS) Provisioner and a collection of network transport mechanisms. The Strategy Selector's role is to take application-specified data delivery requirements (e.g. bandwidth, latency, jitter, reliability, etc) and translate them into networking and computational resource allocations needed to meet the applications' demands. The Strategy Selector must monitor the current state of the network, select an optimal transmission protocol, and make QoS requests (if available.) Some protocols such as Forward Error Corrected UDP or Parallel TCP (described later) may pose additional computational overhead, which the Strategy Selector must take into account. If QoS is available, the Strategy Selector must contact the Admission Control system to determine if the available bandwidth is available, and then make a reservation using the Reservation Controller. Once a strategy has been activated, the Adaptive Controller must monitor the progress of the data transmission and adjust networking and computational parameters to sustain the desired performance. To accommodate multiple simultaneous and heterogeneous network flows, the Adaptive Controller may alter some of the low-level transport protocol parameters (such as buffer or window size,) or may adjust QoS reservations dynamically.

It is clear that the realization of such a system is an ambitious endeavor. In this paper, we will present several steps we have taken towards understanding the issues and creating the necessary building blocks towards such a system. These include:
-    a study of the behavior of the Differentiated Services QoS with respect to the requirements of real-time tele-immersive applications;
-    the development of a collection of advanced data delivery mechanisms as part of our CAVERNsoft Tele-Immersion toolkit[2, 3]. These include Parallel TCP, and Reliable Blast UDP for bulk data transfer, and Forward Error Corrected UDP for semi-reliable, low-latency state transfer.
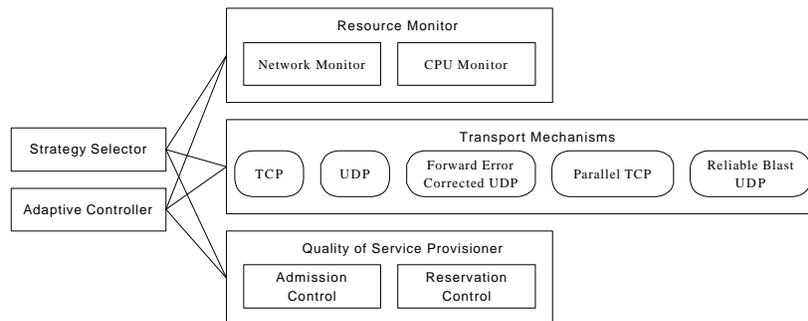
Figure 2: Components of an Adaptive Networking System

3   Quality of Service for Real-time Collaborative Applications

Network Quality of Service (QoS) refers to the ability of an application to request a guaranteed level of networking service in the form of bandwidth, latency or jitter (in reality only bandwidth is guaranteed.) There are two well-known types of QoS: Integrated Services (IntServ) and Differentiated Services

(DiffServ). IntServ achieves QoS by having intervening routers maintain the state of a particular network link between two end points[4]. While this method can provide a hard guarantee on QoS, it is ultimately not scalable to the larger Internet. DiffServ, on the other hand, takes the approach of marking network traffic with a priority level that can be interpreted by the router to effect special treatment of the data packet[5]. In particular, the marked packets are promoted to a higher priority queue in the router and, as a result, spend less time in the router. Packets that are not marked are attached to a lower priority queue, and in some cases may be dropped when congestion arises. A more detailed description of DiffServ may be found in the paper by Foster et al [6].

The common misconception in the field of collaborative virtual reality (CVR) is that QoS, once it is available, will solve all of CVR's networking problems. At the same time networking experts believe that bandwidth provisioning is the only form of QoS needed by applications. The experiments described below were performed as a joint project between collaborative virtual reality and networking experts in order to shed some light on these common misconceptions.

A series of experiments were performed over a wide area DiffServ testbed as part of the EMERGE project. EMERGE [7] is a Department of Energy funded project for designing, deploying and testing Differentiated Services on an IP/ATM Regional GigaPoP Network interoperating with ESnet for applications in Combustion, Climate and High-Energy Physics. The main participants of the experiments were EVL and Argonne National Laboratory (ANL)- approximately 30 minutes away by car.
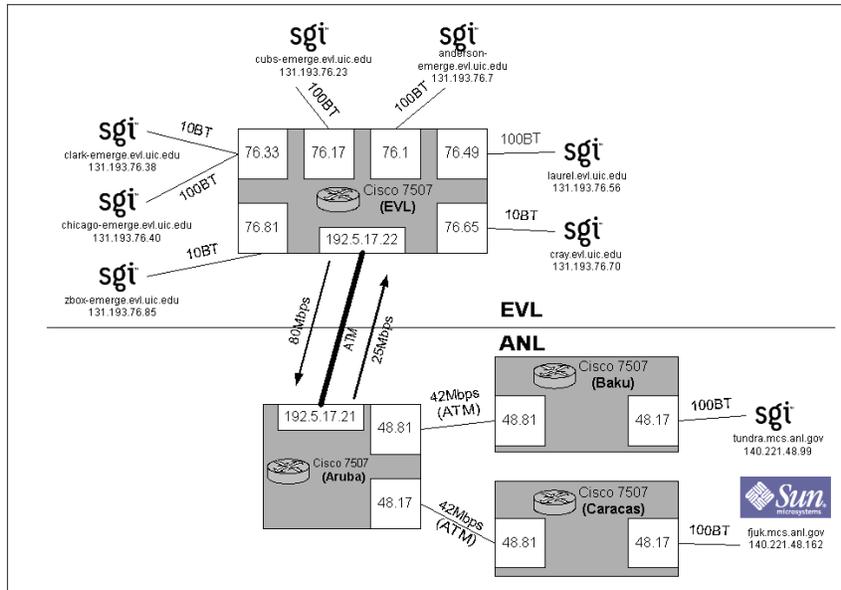
Figure 3: EVL / ANL DiffServ testbed architecture.

Cisco 7507 DiffServ routers at ANL were connected to DiffServ routers at EVL as shown in Figure 3. The router at EVL had Weighted Fair Queueing enabled; and the routers at ANL had Priority Queueing enabled to produce DiffServ's Expedited Forwarding behavior. Several experiments were performed over this testbed, but in the interest of brevity, we will report on the results of only one experiment. The reader is encouraged to peruse the detailed technical report at [8]. The goal of the experiment was to observe whether DiffServ is able to maintain bandwidth allocations while keeping latency low, especially during periods of high network congestion. The experiment began with the transmission of a steady foreground stream of UDP data at the networks saturation point (25Mbps) from ANL (Tundra) to EVL (Laurel). After some time, 25Mbps of competitive networking traffic was transmitted from Fjuk to Laurel to increase congestion over the inter-domain link between EVL and ANL. Then DiffServ was enabled on the foreground stream to determine if bandwidth and latency recovery would occur. GARA (the General-purpose Architecture for Reservation and Allocation,) developed by ANL was used to make the DiffServ reservation[9].

In Figure 4 the dip in the top graph shows that when competitive traffic is injected the throughput of the foreground stream suffers. However, when DiffServ is enabled the throughput is brought back to near its original levels.
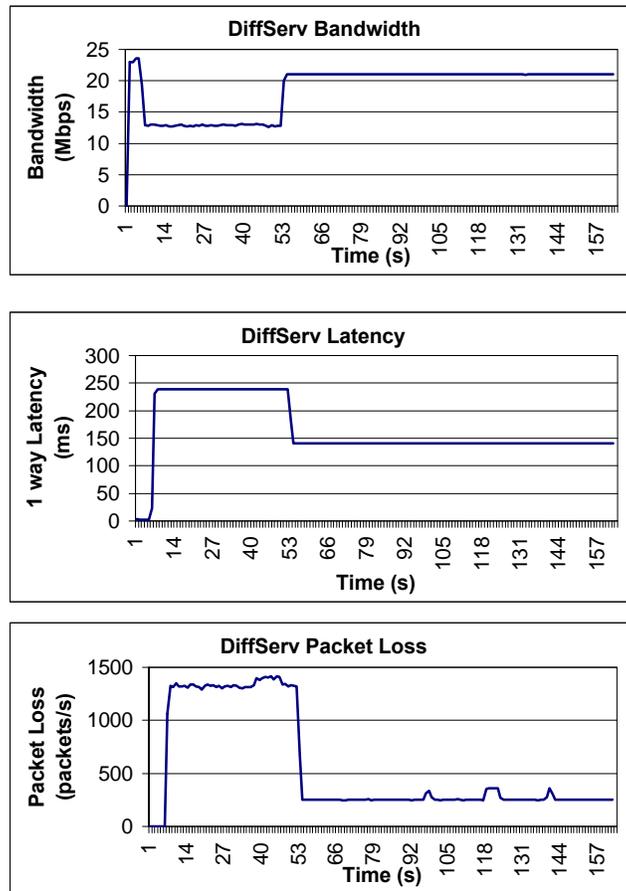
Figure 4 : Results showing that DiffServ, while being able to make bandwidth guarantees, may not always provide latency guarantees.

This shows that DiffServ is effective in providing bandwidth guarantees. However, the second and third graphs show that latency was only partially restored and packet loss has doubled. Note also that the restored (*one-way*) latency is at approximately 150ms, which has been shown in the past, to be intolerably high for real-time tightly coupled interactions in Tele-Immersion. Park et al[1] [10, 11] have found that the *roundtrip* latency threshold where human performance begins to noticeably degrade is approximately 200ms**.** The results suggest that while DiffServ is suitable for making bandwidth guarantees it is unable to reliably make latency guarantees *under heavy network congestion*. We have performed the same experiment over uncongested links (from EVL to ANL over an 80Mbps private virtual circuit) and have not observed the same performance degradation[8].

We believe the increased latency under heavy congestion is due to queueing that is occurring at the interior router (Aruba.) Priority queueing would ensure that the DiffServ marked packets would receive priority transmission, however to prevent starvation the router will allow a small amount of background traffic through. Since the private virtual circuit from ANL to EVL is bottlenecked at 25Mbps, the allowance of

---

[1] Park's results also suggest that 200ms of roundtrip latency with 0 jitter has the same effect on users as 10ms of roundtrip latency with 7ms jitter. In essence, this implies that the minimization of jitter is also critical in realtime tightly coupled tele-immersive applications.

background traffic caused the priority queue's length to grow hence resulting in longer delays in priority packet delivery.

Finally to confirm our results in the context of a real tele-immersive application, we repeated the experiment using the Tele-Immersive Data Explorer (a collaborative VR application for querying data mining servers)[12]. In this experiment, low-bandwidth avatar data represented the foreground traffic. Again over a congested network, we observed the same increase in latency, as in our previous experiments [8].

4   Advanced Data Transport Techniques for Tele-Immersion

CAVERNsoft is an open source, cross-platform, C++ toolkit for building tele-immersive applications[2, 3]. Although its initial intended audience was developers of VR applications, CAVERNsoft's decoupled networking tools has made it useful in building networked applications in general. CAVERNsoft supports a broad collection of low-level to high-level data distribution tools. Low-level tools include classes for basic UDP, TCP and multicasting. Mid-level tools consist of classes for remote procedure calls, key/value database, parallel sockets, UDP and TCP reflectors, and remote file transfer. High-level tools include C++ classes for rendering articulated avatars, audio conferencing, and shared scene graphs. As one of the intended audiences for these tools are computational scientists, a 64-bit version was implemented- for example the 64-bit remote file transfer API allows the transfer of files greater than 2 Gigabytes in size. Finally, since optimal network utilization requires careful monitoring of network performance, every CAVERNsoft network class has been instrumented to measure bandwidth, latency, jitter and burstiness.

In our continuing efforts to expand CAVERNsoft's networking services, we present below, schemes for reliable low-latency state transmission, and high throughput bulk data transfer.

4.1 Reliable Low-Latency Data Transfer for Tele-Immersion

For long distance networks such as international networks, latencies are high (on the order of hundreds of milliseconds). In Tele-Immersion we would ideally like state updates in the shared environment to occur with a minimum amount of latency and with a high degree of reliability. A scheme is therefore needed to transmit data reliably over long distances without requiring the acknowledgement typically used in protocols such as TCP. Forward Error Correction (FEC) provides a promising solution to this problem. FEC works by collecting between 1 and N (typically 2 or 3) data packets and performing a bit-wise operation on the packets (such as XOR) to generate a "redundant" packet. This packet is delivered along with the regular UDP traffic as a separate UDP stream. If any data packets are lost, FEC packets can be used to reconstruct the missing packet. A detailed description of the encoding algorithm can be found in [13].

To evaluate our FEC implementation we performed a series of experiments between EVL and SARA (SARA- Stichting Academisch Rekencentrum Amsterdam) over STAR TAP (the Science and Technology Advanced Research Transit Access Point- a National Science Foundation funded project to interconnect international high speed networks.)[14]. The experiment involved comparing the latency, jitter and packet loss of basic UDP, TCP and UDP augmented with FEC. Details on the experimental setup can be found in [13]. Figure 5 shows the results of the experiment using an FEC scheme that generated 1 redundant packet for every 3 consecutive UDP packets. Notice that FEC does appear to incur lower latency than TCP but slightly higher latency than UDP. Also, note that the benefits of FEC are greatest at small packet sizes since it takes less time to accumulate 3 small packets and encode. The goal in future work would be to bring FEC performance closer to UDP performance while maintaining maximal reliability.
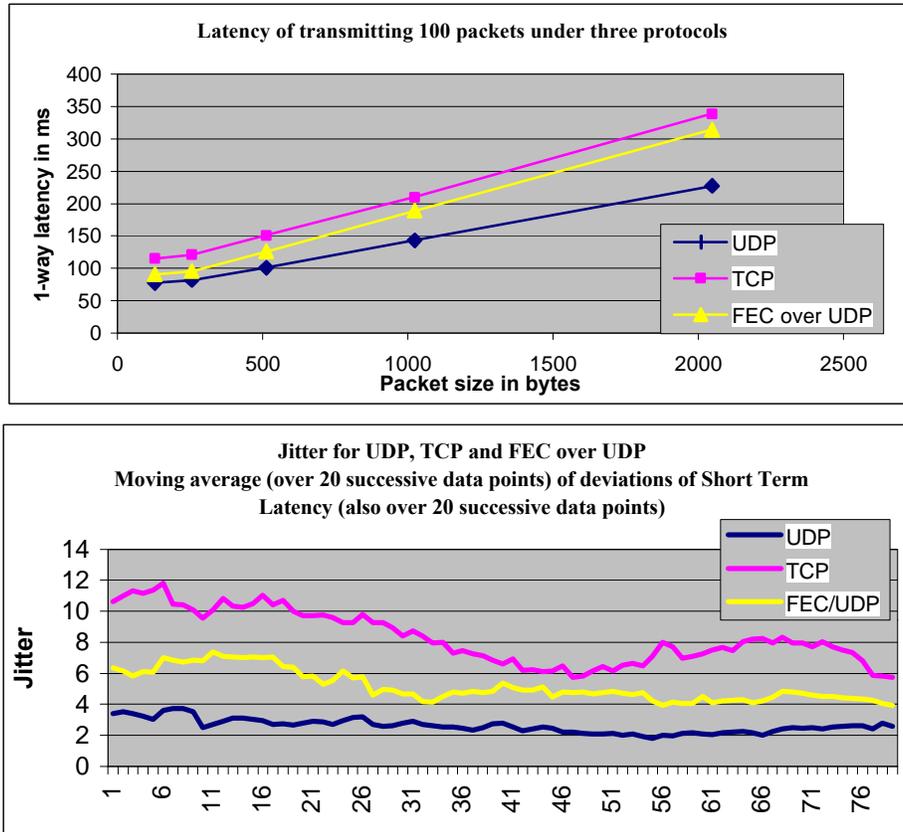
Figure 5: The top graph shows that our FEC scheme derives its greatest benefit when packet sizes are small as larger packet sizes incur additional buffer processing time. The bottom graph shows that FEC also introduces jitter in the data stream. Jitter in is computed by first calculating the short-term latency over every 20 data points and then computing the average deviation of the instantaneous latency as compared to the short-term latency.

Figure 5 also shows the variation of latency (jitter) between TCP, UDP and FEC. Notice that FEC produces lower levels of jitter than TCP but higher levels than standard UDP.

Finally, to observe packet loss of UDP versus FEC, packets were sent at a rate of 50Mbps over a 100Mbps link between EVL and SARA. Then an additional 50Mbps of congestion traffic was injected. The results in Figure 6 shows that FEC was quite effective at recovering lost packets. However, note that while our FEC scheme is good at recovering intermittent lost packets it cannot recover from the loss of large consecutive blocks of packets. To recover from this the selection of encoding packets must be more widely distributed across the data stream. This unfortunately has the effect of increasing latency. Another drawback of FEC is that the FEC packets themselves can become a source of congestion. Hence, careful selections of FEC parameters need to be made to ensure optimal performance. For the purposes of transmitting real-time state information in tele-immersive applications, the recommendation is to encode few (like 3) and small (like 1Kbyte) consecutive packets over networks that have the capacity to sustain the additional bandwidth needed by FEC.

	Packet Loss
UDP	1.90%
FEC	0.05%
UDP with congestion	17.40%
FEC with congestion	4.15%

Figure 6: Comparison of Packet Loss in UDP vs. FEC over an uncongested and congested network.

4.2 High Throughput Techniques for Tele-Immersion

Quality of Service is designed to guarantee a level of service (mainly bandwidth) for the networked application. It does not however guarantee that the networked application will be able to take full advantage of the bandwidth, even when it is made available. Since tele-Immersive applications are so highly interactive, the expectation is for the retrieval of distantly located data sets, such as 3D models, to be equally expedient. This can only be achieved by being able to maximally use the available bandwidth to deliver the data.

To support high-bandwidth bulk data transfer, we will describe two techniques- Parallel TCP Socket Striping and Reliable Blast UDP. These two schemes allow applications to overcome what is classically known as the Long Fat Network (LFN) problem[15]. The LFN problem occurs over long, high-bandwidth networks, such as those between the United States and Europe or Asia. The high round-trip latencies in these networks (at best 120ms) will result in gross bandwidth *under*-utilization when default TCP is used for data delivery. This is because TCP's windowing mechanism imposes a limit on the amount of data it will send before it must wait for an acknowledgement. The long delays that occur over international networks means that TCP will spend an inordinate amount of time waiting for acknowledges, which in turn means that the client's data transmission will never reach the peak available transmission rate of the network. Traditionally this is "remedied" by modifying TCP's window and buffer sizes to match the bandwidth * delay product (capacity) of the network. For example, for a 45Mbps link between Chicago and Amsterdam, with an average round trip time of 150ms, the capacity is 45*0.15/8 = 0.84Mbytes.
Unfortunately adjusting TCP window size is problematic for two reasons: firstly, on some operating systems (such as IRIX for the SGI,) the window size can only be modified by building a new version of the kernel- hence this is not an operation a user-level application can invoke. Secondly, one needs to know the current capacity of the network in order to set the window size correctly. The current capacity varies with the amount of background traffic already on the network.

Parallel TCP Socket striping overcomes the LFN problem by partitioning a payload and delivering it over multiple TCP connections. Reliable Blast UDP overcomes the LFN problem by literally "blasting" UDP packets over the networks and then transmitting an acknowledgement only after the full payload has been transmitted. Our experiments between Chicago and Amsterdam (at SARA) will illustrate how each of these schemes can significantly improve throughput over LFNs.

Parallel TCP Socket Striping

Figure 7 shows the results of a socket stripe test between SGI Onyx2s at EVL and SARA. At the time of the experiment the STAR TAP link provided a bandwidth of 45Mbps with a round trip time of approximately 150ms. The experiment was conducted using between 1 and 30 sockets. Notice that using TCP's default window size (64Kbytes) and one socket (as in tools like ftp,) our data transfer is able to achieve a throughput of only 3.5Mbps. But when 12 sockets are used we are able to achieve a throughput of approximate 32Mbps. Note also that if more than the needed sockets are used the performance remains high however somewhat bursty. This burstiness is due to TCP throttling-back its data delivery in response to congestion. While this burstiness does not pose a significant problem for a low bandwidth link, it can be a significant waste of available bandwidth on higher bandwidth links.

Finally, note however that when the TCP window size is set high, the benefits of parallel sockets are no longer as pronounced (Figure 8 shows the result of using a TCP window size of 1.875M.)

Parallel TCP socket striping is currently in use in our tele-immersive image-based volume visualizer (called CIBRView- Collaborative Image Based Rendering Viewer) to retrieve large numbers of composited texture maps from remotely located servers. In a test campaign between Chicago and Japan, at the INET2000 conference in Yokohama, we found parallel TCP to dramatically increase CIBRView's interactivity[2].
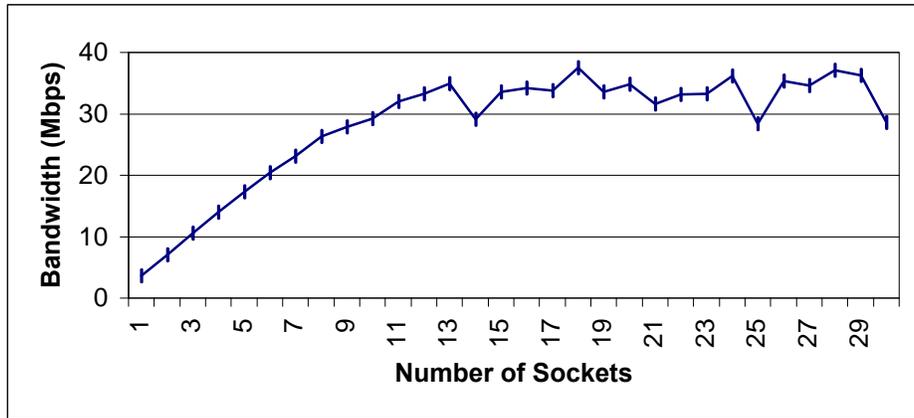
Figure 7: Achieved bandwidth while transmitting 50M from Amsterdam to Chicago over a 45Mbps link using parallel sockets with default TCP window size of 64K. Notice that throughput reaches a maximum at 12 sockets. Beyond 12 sockets, throughput remains high but bursty. This burstiness is due to each TCP socket activating its congestion control mechanism.
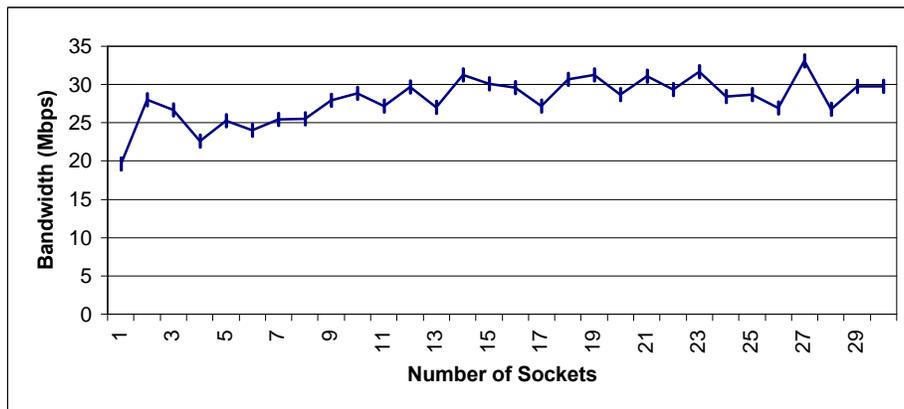
Figure 8: Plot of average achievable bandwidth vs. number of parallel TCP sockets used to deliver a 50M File from Chicago to Amsterdam over 45Mbps Link. In this case, the TCP window size was set to 1.875M rather than the default 64K. Notice that when the window size is set large enough the benefits of parallel sockets are no longer as pronounced.

Reliable Blast UDP (RUDP)

When operating over QoS-enabled networks the probability of packet loss is low. To take advantage of this we have implemented an alternative reliable data transmission scheme using UDP augmented with acknowledgements. The scheme works by "blasting" the contents of a data file at just below the available bandwidth without asking the remote site to acknowledge any of the packets. Hence, all the bandwidth is used for pure data transmission. At the remote site, a tally is kept for all the packets that have arrived and, after some timeout period, a list of missing packets is sent back to the sending client. The sender reacts by resending all the missing packets and again waiting for another negative acknowledgement.

Figure 9 shows the results of using RUDP to deliver a 50M file over a 100Mbps link between EVL and SARA. Since QoS is currently not available between the two sites, we chose to send data at rates well below the maximum available bandwidth of the network link- in essence emulating a smaller reserved QoS link. The table shows the bandwidth at which RUDP data was transmitted and the effective throughput of the total file transfer. Notice that on an over-provisioned network, effective throughput is almost as high as the sending bandwidth. This is in contrast to TCP, which typically incurs a 20% bandwidth loss by requiring frequent acknowledgements. Note also that as we approach the bandwidth limit of the link, performance begins to decrease- however performance is no poorer than what is expected of optimally

tuned TCP. Remember that these experiments were conducted without the benefit of QoS. Hence, interference from competing traffic streams was possible and likely. We predict that our results would yield even better performance with QoS.

Sending Bandwidth (Mbps)	Effective Bandwidth	Number of NAKs
20	19.7	0
40	38.5	0
60	54-57	1
80	56-70	2
90	61-77	3

Figure 9: Comparison of transmission bandwidth vs. effective throughput using Reliable Blast UDP over a 100Mbps non-QoS provisioned link between Chicago and Amsterdam. When the network is over provisioned performance is good. We anticipate the best results of RUDP to occur in conjunction with the use of a QoS scheme such as DiffServ or IntServ.

5   Conclusion

The work presented in this paper serves as a beginning for our research in adaptive networking. While our interest has been driven mainly by our need to provide better networking services for Tele-Immersion, the results we have presented are generalizable to other networked applications.

Future research on our adaptive networking framework will consist of the following goals:

1. To define QoS from the point of view of application developers. For example QoS may mean: "reliable low-latency transmission," "faster download," "faster realtime response," "guaranteed delivery by a deadline."
2. To characterize transmission protocols, QoS mechanisms and computational resources as a set of parameters or functions.
3. To find suitable mappings that will translate application-level descriptions of QoS to characterizations of required network services. Ie. To map from item 1. to 2. to guarantee optimal application and network performance. Heuristics may be used initially. For example:
    o If data requires low latency but can tolerate much loss use UDP
    o If data requires low latency and can tolerate low loss use Forward Error Corrected UDP
    o Else use TCP
    o If data set is large and requires reliable delivery:
        ▪ Use TCP and adjust window size and buffer size to match network capacity if possible;
        ▪ Else use parallel TCP socket striping;
        ▪ If QoS is available- use Reliable Blast UDP
4. To find prediction models that can be used to predict the consequences of making mapping decisions during dynamic network situations.
5. To test our predictions on real applications and on real networking testbeds.

Our QoS experiments have shown that Differentiated Services is able to provide bandwidth and latency guarantees when the network is not over-subscribed. However, if the network *is* over-subscribed then only bandwidth guarantees are possible. We believe that Integrated Services QoS should be able to overcome the limitations of DiffServ. However the main drawback of IntServ is that it is not scalable to the larger Internet. We, as well as other researchers have proposed that a scalable solution might require the use of IntServ at the edge routers (where the clients connect,) and the use of DiffServ in the core routers (where the excess bandwidth resides.)[16] We are in the process of constructing a network testbed to firstly, examine IntServ, and then a combination of IntServ and DiffServ.

Finally, we are in the continual process of translating our research results into useable tools for the computational science community. CAVERNsoft [2, 3], our open source toolkit for building networked applications, already has built-in performance monitoring, and parallel socket striping capabilities. Modules for Forward Error Correction and Reliable Blast UDP will be incorporated in the near future.

6   Acknowledgements

7   References

[1]   J. Leigh, Johnson, A., DeFanti, T., et al., "A Review of Tele-Immersive Applications in the CAVE Research Network," presented at IEEE VR99, Houston, Texas, 1999.

[2]   K. Park, Cho, Y., Krishnaprasad, N., Scharver, C., Lewis, M., Leigh, J., "CAVERNsoft G2: A Toolkit for High Performance Tele-Immersive Collaboration," presented at ACM Symposium on Virtual Reality Software and Technology, Seoul, Korea, 2000.

[3]   J. Leigh, Cho, Y. J., Krishnaprasad, N., Nayak, A., Fang, R., "the CAVERNsoft Tele-Immersion Toolkit : http://www.evl.uic.edu/cavern/cavernG2."

[4]   IntServ, "IETF Integrated Services Charter : http://www.ietf.org/html.charters/intserv-charter.html."

[5]   DiffServ, "IETF Differentiated Services Charter : http://www.ietf.org/html.charters/diffserv-charter.html."

[6]   I. Foster, Kesselman, C., "Globus. A Metacomputing Infrastructure Toolkit," *International Journal of Supercomputing Application*, vol. 11, pp. 115-128, 1997.

[7]   EMERGE, "EMERGE : http://www.evl.uic.edu/cavern/EMERGE."

[8]   J. Leigh, Yu, O., Verlo, A., Roy, A., Winkler, L., DeFanti, T. A., "Differentiated Services Experiments Between the Electronic Visualization Laboratory and Argonne National Laboratory : http://www.evl.uic.edu/cavern/papers/DiffServ12_12_2K.pdf," 2000.

[9]   I. Foster, Sander, V., Roy, A., "A Quality of Service Architecture that Combines Resource Reservation and Application Adaptation," presented at 8th International Workshop on Quality of Service (IWQOS, 2000.

[10]   K. Park, Kenyon, R, "Effects of Network Characteristics on Human Performance in a Collaborative Virtual Environment," presented at IEEE VR, Houston, Texas, 1999.

[11]   J. Leigh, Park, K., Kenyon, R., Johnson, A., DeFanti T., Wong, H, "Preliminary STAR TAP Tele-Immersion Experiments between Chicago and Singapore," presented at 3rd High Performance Computing Asia Conference (HPCAsia '98), Singapore, 1998.

[12]   N. Sawant, Scharver, C., Leigh, J., Johnson, A., Reinhart, G., Creel, E., Batchu, S., Bailey, S., Grossman, R, "The Tele-Immersive Data Explorer: A Distributed Architecture for Collaborative Interactive Visualization of Large Data-sets," presented at 4th International Immersive Projection Technology Workshop, Ames, Iowa, 2000.

[13]   R. Fang, Schonfeld, D., Ansari, R., Leigh, J., "Forward Error Correction for Multimedia and Tele-immersion Streams : http://www.startap.net/images/PDF/RayFangFEC1999.pdf," 2000.

[14]   T. A. DeFanti, Goldstein, S., "STAR TAP, http://www.startap.net."

[15]   W. R. Stevens, "TCP/IP Illustrated," vol. 1: Addison Wesley, 1994, pp. 344-350.

[16]   O. Yu, "Dynamic QoS and Routing Support for Real-time Multimedia Applications over Next Generation Internet," presented at Proceedings of IEEE International Conference on Multimedia and Expo 2000, 2000.