International Conference on Simulation and Multimedia in Engineering Education

Edited by
Marco Roccetti
Mahbubur Rahman Syed

Third International Symposium on Collaborative Technologies and Systems

Edited by
Waleed W. Smari
William McQuay

Simulation Series
Volume 34
Number 1

SCS

The Society for Modeling and Simulation International

ISBN: 1-56555-243-1

A QoS Middleware for Interactive Multimedia Applications in Education[1].

Giovanni Pau*§
*** Computer Science Department University of California Los Angeles**
Boelter Hall, 420 Westwood Plaza - Los Angeles, CA, 90024
§ DEIS Dipartimento di Elettronica Informatica e Sistemistica
Universitá di Bologna – Viale Risorgimento, 2 – 40126 Bologna IT
{gpau}@cs.ucla.edu

Keywords: Quality of Service, Interactive Multimedia in Education, Mobile Computing, Wireless Communications

ABSTRACT

Interactive multimedia applications, such as Virtual World Navigation Systems or interactive video are becoming largely useful different areas of human activity. In remote education, for example the use of a VWNS allows a virtual trip across the space and the centuries to visit a temple at the time it was built [Jeps95]. The described approach is extremely user driven; each student can select his own path for the visit in a very interactive way. Unfortunately this class of applications still requires a tremendously large amount of computational and network resources and usually a special platform is required. In this paper we describe the architecture of a nouvelle UDP based middleware with a sophisticate mechanism for bandwidth estimation especially designed for drive interactive multimedia applications performance across the Internet to a standard Lap Top or PDA.

INTRODUCTION

Interactive client-server multimedia applications require high computing and communications bandwidth. The allocation of processing functions between server and client, and the efficient use of communication bandwidth are key design issues, particularly when clients are "thin", e.g. running on laptops or PDAs with limited computing speed and limited power. An example of extremely challenging interactive multimedia applications is high fidelity virtual world navigation. At UCLA, the Urban Simulation project has succeeded in providing a simulated, yet sufficiently rich with detail, navigation experience through various Los Angeles neighborhoods, and through parts of the UCLA campus as well as the Jerusalem's Temple Mount as site appeared prior to its destruction by Roman troops in the year 70 CE.

The current implementation stores images of these areas on a parallel database: the Virtual World Data Server, developed at the UCLA Computer Science Department. The server receives requests specifying the user coordinates, his/her direction and speed. It retrieves data objects and textures stored in the database and forwards them to the client. Currently, client systems are in charge not only of specifying users' coordinates, speed and direction; but also of *rendering* and *displaying* the tile of the virtual space in which the user currently reside. When visual databases become too large for local memory they must be fetched page by page from remote storage. This leads to some challenging problems related to delivering the needed tiles of the visual database to the rendering client in a timely manner. In order to hide the paging and transmission latency, the remote storage server does anticipatory tile paging and feeding (to the client). To this end, it must maintain a model of what the client will need to render next. It will also need to take into consideration disk, network and client processing delays. The previous work for the Virtual World Data Server project provides solutions to these challenges by incorporating new techniques for storage layout and disk scheduling [Munt97] as well as new approaches for admission control and traffic shaping for interactive applications [Munt99].

Fig. 1 The Forum of Trajan: a Virtual Reconstruction

In this paper we describe an architecture based on a QoS aware middleware as a support for remote access to high end applications (such as Urban Simulation/VWDS) from portable clients, ie, **thin clients** connected by "last mile" network facilities ranging from e-net and Wireless LAN

[1] This work has been partially funded by the project NEBULA of the Italian Ministry for University and Research "MURST".

speeds (10Mbps), down to digital cellular network speeds (say, 125 Kbps to 250 Kbps), as offered by GPRS, UMTS and Metricom. This capability allows broader use of interactive multimedia for a variety of applications such as the guided tour of a building yet to be constructed, the collaborative reconstruction of an archeological site by different experts around the world, the advanced games played across the Internet, etc. The following description, for the sake of the simplicity, is applied to the Urban Simulation/Virtual World Data Server project at UCLA but can be used to grant the, through thin clients, the access to many different multimedia applications.

SISTEM ARCHITECTURE

The current system architecture of the Virtual World Data Server (VWDS) project and the Urban Simulation still requires a **powerful client** with large main memory (at least several hundred megabytes), and often a multiple processor configuration. The current implementation relies on local communication links between Sever and Client with speed on the order of Gbps. In order to give an off site demonstration, the UCLA team must at this time either arrange that a network and hardware environment similar to the UCLA Lab is at the demo site, or else, the demo is by videotape only [Jep93, Jeps95].

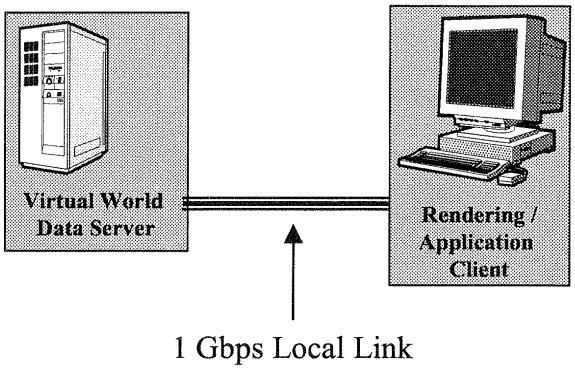

Fig. 2 Current Virtual Data Server/Application Architecture

With thin clients such as laptops and PDAs, rendering must be moved to the server, which then will send rendered images over the network to the thin clients. The communication bandwidth to such clients is typically in the range of 100 Kbps to 10 Mpbs. We propose to use an adaptive MPEG-4 video stream with variable bandwidth and an end-to-end latency below 150ms with careful delay jitter control. The delay and jitter constraints are required to maintain reasonable interactivity [Broo99]. Latency hiding is key also in this solution approach. However, the server not only must carry out anticipatory retrieval and rendering, it also must deliver "multiple video streams" to the thin client since in this case the client cannot store multiple tiles to prepare for moves in all possible directions.

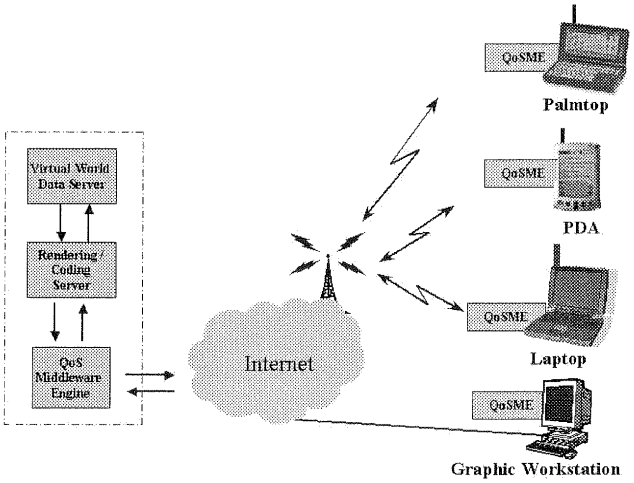

Fig. 3: The QoS Middleware/Application architecture

Note that in a dynamic, possibly mobile environment, the Server must continuously monitor the performance parameters of the connection (ie, bandwidth, delay, delay jitter) in order to optimize the delivery of the multiple video streams to the thin client. In particular, it must select compression rate, frame rate, size of the picture, number of video streams to be simultaneously transmitted, etc. To this end, we plan to develop a **thin client middleware** component that can be easily integrated with our existing client software and with the thin-client running at a remote location. The middle-ware component manages the communication between a thin-client and the rendering server. It keeps track of network and connection performance in order to adapt the video streams and to hide latency, thus satisfying the basic performance constraints.

The Quality of Service Middleware

The middleware must perform two important classes of functions:

(a) Gathering and delivering network QoS parameters, e.g. bandwidth, delay, delay jitter, to the rendering server. We envision a specialized transport layer able to gather relevant end-to-end measurements and to forward them to the application via proper API(s).

(b) Support the application in adapting to dynamic network performance characteristics and (predicted) user behavior. As part of this function, QoS parameters are negotiated/re-negotiated to match network status and user demands. In the Urban Simulation/VWDS context, the application adaptivity can be manifested in the use of a new code, a different image resolution, a change in frame rate, or even the use of a different user model.

In particular the QoS Middleware will be in charge of:

(a) the development of end to end QoS monitoring techniques for bandwidth, delay and jitter;
(b) the implementation of a novel rate control technique that prevents congestion and yet allows the video stream(s) to utilize in full, and to fairly share, the bandwidth available on the path;
(c) the development of prediction strategies and adaptive video formatting, encoding and compression at the server.

The QoS middleware must be portable to different client systems (e.g. Laptops, PDA, 3G cell phones, etc.) and across different network infrastructures (e.g. wireless LAN, blue-tooth, UMTS, etc.).

The middleware component manages the communication between the new remote thin-client and the rendering host. This integration includes an API that provides for the middleware to replace a local user on the rendering host for navigation and allows retrieval of rendered frames for processing and transmission to the thin client for display, all within our basic performance constraints.

A user's thin client experience should be as close as possible to using the rendering host directly. At a minimum the thin-client should be able to transmit application navigation and command data to the rendering host. Visual feedback will come through the return video stream provided by the middle-ware component.

The middle-ware provides four basic parts. The first is to transmit user input from the thin-client to the rendering host. There are two kinds of input that need to be considered, statefull and stateless. Statefull data are things like application mode changes that must be delivered to the rendering host. Stateless data are things like user telemetry which is transmitted continuously as the user navigates the visual database. This part is relatively simple as the bandwidth required is expected to be on the order of kilobits per second. Separate TCP and UDP connections can be dedicated to statefull and stateless data.

The second responsibility of the middle-ware is to transmit rendered frames to the thin-client. For the purposes of a prototype system we are going to make an assumption that the rendering host can deliver frames at a specific minimum rate and resolution, 720x576 with 16 bits per pixel (bpp) at 15 frames per second (fps); clearly some form of compression is needed as the raw data rate would be more than 90MBits/s! We propose to encode these rendered frames as an MPEG-4 elementary stream. MPEG-4 should give us the level of compression we are looking for (in the order of a few Mbps) and there are already implementations available for both encoding and decoding [Bier99, Rose95].

However, simply encoding each rendered frame and sending it to the thin-client to decode and display will surely exceed the user imposed latency constraint of 150ms. Latency in this system is composed of several elementary contributions. There is the small, but variable, latency of transmitting user navigation inputs from the thin client to the rendering host. That user input is used to render a frame, followed by reading the frame back from the graphics hardware and encoding it. At this point we are going to be very close to our 150ms constraint limit and the encoded frame has yet to be transmitted to the thin-client. Thus, the network's variable latency and the thin-client's decoding of the video stream must yet to be added. Simplifying the encoding will increase file size and put us in danger of exceeding our bandwidth constraint; on the other hand, higher compression levels will increase processing time and push us over our latency constraint. Techniques are needed to hide the inherent end-to-end latency of the system while satisfying our basic constraints. The third and forth components of the middle-ware are intended to provide this capability.

Latency hiding is accomplished by working ahead of the user. Since our system is meant to be used by interactive applications, prediction of the user's future decisions is needed. Clearly, there is no context by which we can predict user state changes to the rendering host so we will consider those exempt from our basic system constraints. User navigation has possibilities for some form of rough prediction. In essence we are going to speculate on what the next frames are going to be. There are two challenges, one of which is that the rendering system has to be able to render frames faster than they are ultimately displayed. The second is that there are really an infinite number of next frames to choose from. A possible solution to the first challenge is to use a very powerful rendering host (like our Onyx) or possibly a cluster of rendering hosts rendering different "guesses" in parallel. Limiting the number of possible future frames to work on is more difficult. However, if we make the assumption that a user will continue to navigate in the direction he is already set on, that narrows things down. Furthermore, if we modify the user model somewhat to additionally limit how fast the user can change the current direction, that narrows the possibilities even further. We can also take advantage of the fact that the rendered frame has a finite resolution. This means that many viewpoints are going to result in the same final image which, in effect, quantizes the number of potential viewpoints that will result in a substantially changed image. These approaches limit the real number of speculated frames to something more manageable [jeps96, jeps97].

Lastly, we see a network performance component that communicates with the encoding and prediction components in order to optimize network usage given the fluctuations present in bandwidth and congestion. When bandwidth is available and congestion is low prediction frames can be freely sent to the thin-client buffer. When the opposite is true, compression could be increased or resolution lowered.

In particular the thin client QoS middleware will provide network QoS information to the applications such that:

(a) the application can negotiate the QoS parameters with the network. Direct negotiation of QoS parameters between application and network requires that the network itself supports QoS monitoring and advertising. Today, this is possible in networks supporting RSVP, or (for Diff-Serv networks) Q-OSPF. Since in the current Internet RSPV, resource allocation and Q-OSPF have not yet been broadly implemented, we will address this scenario only via simulation [Kaza01a].

(b) the application can adapt its parameters (eg, frame rate, compression scheme, etc) so as to run most efficiently with the available, possibly continuously changing network resources (this is mandatory if the network does not support QoS, or if the application has selected the best effort service)[Kaza01a, Kaza01b];

(d) the application can control its rate so as to fairly share network resources and be "friendly" to other sessions. This is a must for real time connections that use UDP and have not reserved network resources in advance (ie, best effort mode). Indeed, this is the majority of the real time applications in the current Internet. To this end, middleware protocols must be developed to avoid congestion and be TCP friendly[Kazao1a,Kaza01b, Kaza01c].

The considered network QoS parameters can be either (1) network layer measurements, or (2) end-to-end measurements. For example, in a Q-OSPF network, the bandwidth is advertised by the network layer. In the current Internet, in contrast, virtually all measurements passed to the applications are end-to-end measurements. In our approach, we focus on end-to-end measurements, since they are the only ones that can be easily implemented in an Internet test-bed today.

The performance parameters those are important to adaptive, high performance applications are: round trip delay, delay variations, bandwidth available to the connection, achieved bandwidth, packet loss, and channel errors (due to random noise).

1. Round Trip Delay: important in our thin client application. It can be computed at the source by TCP, or for UDP connections.

2. Bandwidth: there are two possible measures: (a) maximum capacity achievable on this connection (typically, the path is probed with a packet pair or a packet train, BEFORE the actual data is transmitted)[Lai01]; or, (b) rate currently achieved by this session.

3. Packet overflow loss and channel error loss: end to end measurements. Unfortunately, it is difficult to distinguish between them.

A key issue for the QoS middleware will be the bandwidth estimation. We propose to monitor the bandwidth for a connection, and to use this information to achieve fair and friendly sharing of network resources. This proposal is based on our experience with TCP Westwood (TCPW), and in fact extends it to rate control for real-time applications [Case01,Gerl01].

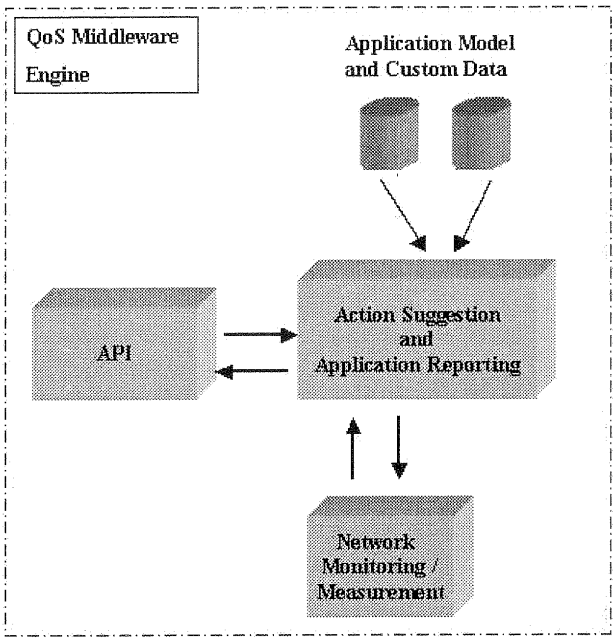

Fig. 4: QoS middleware Architecture

The middleware can implement the rate estimation as (1) an extended version of RTP, or (2) at the application level. In either case, methods to gather and average bandwidth/throughput samples are needed. In contrast to TCPW, we might prefer here to have estimation of the bandwidth done at both sender and receiver.

We have already started to investigate the use of bandwidth estimates in controlling layered encoding in video streaming. To provide a preliminary assessment of this idea, we ran the streaming application over TCPW. Five layers of video encoding were used. Based on the estimate of available bandwidth provided by TCPW, the streaming application adjusted its offered traffic rate by choosing an appropriate video encoding layer. Preliminary simulation experiments confirmed that using bandwidth estimates was beneficial in that the application appropriately adapted to network conditions, and was friendly in sharing available bandwidth with other types of traffic.

Experimental Test-bed

Once developed the proposed architecture will be evaluated with a distance learning application of the urban simulation tool. In particular the "Forum of Trajan" archeological reconstruction will be used as a case of study. The temple

model encompasses a significant piece of ancient Rome including not only the well studied Monuments, but also the markets, the homes of the working class (the Suburra) and one or more of the early Christain churches (Santa Maria Maggiore). The objective of this project is to create a cross section of Roman life which encompasses all facets of the ancient City. Virtual actors and street life have been incorporated in the model to allow the visitor to experience a typical day in Rome at selected times in the history of the City. Because we have incorporated the concept of time into the system, we are creating a history of Rome, by adding attributes to the graphic objects (static or animated), which comprise the virtual city of ancient Rome, specifying creation and dissolution dates. The user can then interactively select (in addition to a starting location) a time or date (which can be changed at any point) to begin the simulation.

Ultimately faculty will be able to conduct their office hours, not in their offices, but in ancient Rome at a time of their own choosing. The students will (from their own homes) join the faculty in ancient Rome on a pre-selected date in time and as avatars they will experience Rome as it was in the days of Augustus (or Trajan, or Julius Caesar). During these sessions it will be possible for those proximally located to see and speak to each other, as well as see and hear the virtual actors (Augustus, Julius Caesar?) as they make their way through the virtual environment. In this fashion the proposed middleware will allow students to access the application and follow the class even if they are connected to the infrastructure by a mobile device such as a PDA, a Palmtop or a Laptop as well as from the Lab Workstation.

CONCLUSIONS

In this paper the design of a new architecture for Interactive multimedia applications has been described. The proposed architecture introduces a middleware to allow the access to high bandwidth and high computing Interactive Multimedia Applications from thin clients. The Urban Simulation Applications has been used to show how the proposed architecture direct benefits the education process. The proposed middleware will be prototyped at UCLA under the NSF funding in the near future.

References

[Bier99] Biermann, H., Hertzmann, A., Meyer, J., Perlin, K., "Stateless Remote Environment Navigation with View Compression", NYU CS Technical Report 1999-784. April 22, 1999.

[Broo99] Brooks, Frederick P., "What's Real About Virtual Reality?", IEEE Computer Graphics and Applications, Vol. 19, No. 6, pp. 16-27, 1999.

[Case01] C. Casetti, M. Gerla, S. Mascolo. M. Sanadidi, and R. Wang; "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links; To appear in Mobicom 2001, Rome, Italy.

[Gerl01] M. Gerla et al: "TCP West: Window Control based on bandwidth estimation" submitted to Globecom 2001; also. UCLA CSD Tech Report

[Jeps93] Jepson, W. "New Technology Includes Community". California Planner, Vol. IV, Issue 2, March/April 1993.

[Jeps95] Jepson, W., Liggett, R., and Friedman, S., "An environment for real-time urban visualization". Proceedings of the Symposium on Interactive 3D Graphics, Monterey, California, (April, 1995)

[Jeps96] Jepson, W., Liggett, R. and Friedman, S., "Virtual Modeling of Urban Environments". PRESENCE 5.1 March, 1996.

[Jeps97] Jepson, W., Friedman, S., "An Efficient Environment for Real-Time Community Visualization", Proceedings of the 1997 InterService/Industrial Training, Simulation and Education Conference, Orlando Florida (December 1997).

[Kaza01a] Manthos Kazantzidis "Wireless Multimedia using Network Measurements" - UCLA CS Technical Report No. 200102

[Kaza01b] M. Kazantzidis, SJ Lee, M. Gerla *"Permissible Throughput Network Feedback for Adaptive Multimedia in AODV MANETs"* In Proceedings of ICC 2001

[Kaza99a] M. Kazantzidis, I. Slain, T.-W. Chen, M. Gerla, Y. Romanenko Experiments on QoS Adaptation for Improving Enduser Speech Perception over Multihop Wireless Networks *IEEE ICC'99*

[Kaza99b] M. Kazantzidis, L. Wang, and M. Gerla, On Fairness and Efficiency of Adaptive Audio Application Layers for Multihop Wireless Networks *Proceedings of IEEE MOMUC'99,* Nov. 1999.

[Lai00] Kevin Lai; Mary Baker *"Measuring Link Bandwidths Using a Deterministic Model of Packet Delay"*. Proceedings of SIGCOMM 2000

[Munt97] Santos, J.R. and Muntz, R., "Design of the RIO (Randomized I/O) Storage Server". Technical Report #970032, UCLA Computer Science Department, May1997.

[Munt99] Wong, W.R. and Muntz, R., "Providing Guaranteed Quality of Service for Interactive Visualization Applications". Technical Report 990052, UCLA Computer Science Department, November 1999.

[Rose95] O. Rose. "Statistical Properties of MPEG video traffic and their Impact on Traffic Modeling in ATM Systems", Univ. of Wuerzburg – Inst. Of Computer Science Research Report N. 101, February 1995