A Framework for Experimenting with QoS for Multimedia Services

Deming Chen, Regis Colwell, Herschel Gelman,
Panos K. Chrysanthis and Daniel Mossé

University of Pittsburgh, Department of Computer Science
Pittsburgh, Pennsylvania 15260

ABSTRACT

It has been recognized that an effective support for multimedia applications must provide Quality of Service (QoS) guarantees. Current methods propose to provide such QoS guarantees through coordinated network resource reservations. In our approach, we extend this idea providing system-wide QoS guarantees that consider the data manipulation and transformations needed in the intermediate and end sites of the network. Given a user's QoS requirements, multisegment virtual channels are established with the necessary communication and computation resources reserved for the timely, synchronized, and reliable delivery of the different datatypes. Such data originate in several distributed data repositories, are transformed at *intermediate service stations* into suitable formats for transportation and presentation, and are delivered to a *viewing unit.*

In this paper, we first review NETWORLD, an architecture that provides such QoS guarantees and an interface for the specification and negotiation of user-level QoS requirements. Our user interface supports both expert and non-expert modes. We then describe how to map user-level QoS requirements into low-level system parameters, leading into a contract between the application and the network. The mapping considers various characteristics of the architecture (such as the hardware and software available at each source, destination, or intermediate site) as well as cost constraints.

Keywords: Multimedia Systems, Quality of Service, QoS, Coordinated resource reservation

1 INTRODUCTION

In the last few years, rapid improvement in computer hardware and software technologies has enabled the sudden growth in development of distributed multimedia systems and services. Examples include teleconferencing, videophone, video-on-demand, and news-on-demand. The ultimate "customers" of multimedia services and users of multimedia systems cannot be expected to be computer-literate. They will be mostly familiar with *VCR-type functions*, and will require a similar, simple user interface to use this new technology. At the system level, however, requirements are expressed in such terms as *bandwidth requirements, traffic patterns, maximum packet delay, packet delay-jitter* and *synchronization* of multiple data streams. In order to support such a VCR-type user interface, a mechanism that translates the users' requirements into system-level parameters is needed. In this paper, we describe such an interface and the mapping of user-level QoS requirements into low-level system parameters, leading into a contract between the application and the network. One unique feature of our interface is the treatment of *value* as a user-specifiable QoS requirement expressed in the form of monetary cost threshold.

Also, our user interface supports both expert and non-expert modes, catering to the different types of users.

It has been recognized that an effective support for multimedia applications must provide Quality of Service (QoS) guarantees. Current methods proposing to provide such QoS guarantees, are based on coordinated network resource reservations. In our approach, we extend this idea to include data manipulation and transformations needed in the intermediate and end sites of future networks, hence providing system-wide QoS guarantees. Given a user's QoS requirements, multisegment virtual channels are established with the necessary communication and computation resources reserved for the timely, synchronized, and reliable delivery of the different datatypes.

In the rest of the paper, we first review NETWORLD, an architecture that provides multimedia QoS guarantees (Section 2), and strongly related work (Section 3). Section 4 forms the crux of the paper, introducing our scheme for experimenting with user and system QoS parameters for multimedia services and the translations mechanisms from the former to the latter. Section 5 concludes the paper by describing the status of our prototype that includes a QoS GUI (graphical user interface), and which is being built by undergraduate students at the University of Pittsburgh.

2 NETWORLD: SYSTEM ARCHITECTURE

In order to set the stage of QoS guarantees, it is important to first introduce NETWORLD (see Figure 1): our assumed system architecture.[3] We have developed NETWORLD to support the structuring of multimedia systems and applications over wide area networks* interconnecting heterogeneous computer systems. Thus, its goal is to interoperate a variety of operating and data management systems, packet scheduling algorithms, and traffic policing mechanisms which are required in the management and delivery of multimedia data from sources to destinations in an application.

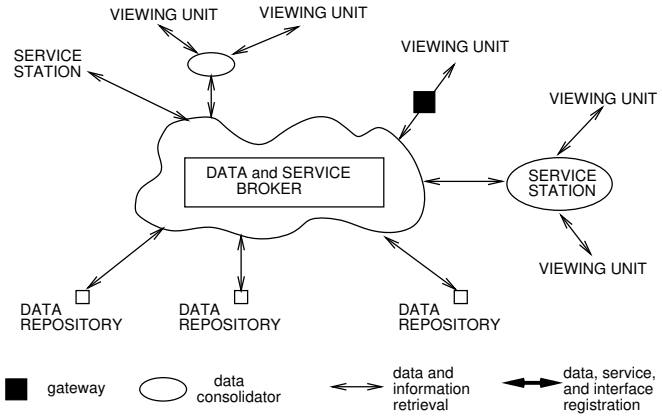


Figure 1: NETWORLD Architecture

Specifically, the NETWORLD architecture comprises:

Viewing Units (VUs), which are multimedia capable client workstations. These workstations allow for different types of multimedia data to be displayed and perused, and contain the appropriate software to translate the user request into requests for data with timing constraints. Furthermore, each VU contains a *Remote Access Manager* that initiates the connection with the remote sites from which it will receive the requested data; it is also part of the allocation scheme, negotiating the QoS parameters and resource usage with the remote components of the NETWORLD.

Data and Service Broker (DSB) that serves both as global resource manager and as an information directory

*The networks we consider are those in which resource reservations are possible.

(i.e., performing meta-scheduling and meta-data management). It provides a common interface between the various components of the system and exports services needed by a VU but not supported by the underlying system. In that respect, the functionality of the DSB resembles that of CORBA,[9] but the DSB is customized for multimedia applications, with timing constraints and special multimedia data services.

Data Repositories (DRs) that incorporate real-time scheduling and data management functions to be able to satisfy requests in a timely fashion. Due to this characteristic, the DRs can be seen as managers of temporal data with timing constraints. That is, real-time tasks execute when a request for multimedia data arrives at the DR. Requests can be known in advance or not, as in pre-planned teleconferences or in dynamic request for current stock prices.

Service Stations (SSTs) that provide specialized data transformations (e.g., data compression and decompression). The resources needed to accomplish the data transformations must be reserved prior to accepting the task of performing the transformations, to provide guaranteed behavior. Furthermore, this must be done without violating the timing constraints of other computations, and is achieved by using a real-time scheduling scheme.

Data Consolidators or *Distribution Centers* (DCs) that temporarily accumulate data to be transmitted to the VUs. These DCs function as front ends to the NETWORLD and at the same time as extended storage for the VUs. The DCs may consolidate multiple requests for the same multimedia data from different VUs into a single request. This simplifies the remote access to achieve better resource utilization, consequently minimizing the cost to the users. The requests from the VUs do not need to be satisfied simultaneously, but data can be sent to the VUs at different times and in accordance to the specified QoS. In that sense, the DCs function as "short-term" DRs.

Multi-Segments that are virtual channels between two sites in the network that will perform some data manipulation (e.g., SSTs, DRs, etc). These segments allow the system to apply a traffic splitting technique for increasing the throughput of channels, or for enhancing the reliability of the channels (by associating more than a single segment with an end-to-end virtual channel).

In summary, NETWORLD supports applications structured as *clients* and *servers* where clients and servers exchange data by means of multi-segmented virtual channels established with specific performance guarantees. For clients with real-time requirements, the NETWORLD provides such guarantees; when time is not important, the NETWORLD adjusts the costs and resource usage for each request.

3 RELATED WORK

Some network reservation schemes grant resources on the basis of fairness,[4] with a channel admission algorithm. To solve the problem of high delays under higher loads and uncontrolled jitter, resource reservation schemes are used.[13] A versatile architecture, called $\mathcal{V}-$NET,[5,6] was introduced to address these concerns, is flexible enough to accept different scheduling disciplines in different sites, different admission control policies, and different types of traffic sources. The architecture is based on an end-to-end communication channel ($\mathcal{V}$-channel) which represents an association between the application's QoS specifications and the network resources. The $\mathcal{V}-$NET provides a framework for flexible support of both real-time and non-real-time communication requirements. In our framework, we use $\mathcal{V}-$NET as our network subsystem, establishing each individual segment as a $\mathcal{V}$-channel.

The Lancaster Quality of Service Architecture (QoS-A),[2] similar to NETWORLD, provides a framework to specify and implement QoS guarantees as service contracts. QoS-A incorporates QoS interfaces, management, and mechanisms across all network layers. In terms of flow, and flow management, QoS-A suggests three classes of service commitment, namely, *deterministic*, *statistical*, and *best effort*. In NETWORLD, we have four classes of service, *basic*, *enhancement*, *excess* and *datagram*, and we allow an application more flexibility as far as distinguishing between basic traffic required to operate properly, enhancement traffic to achieve acceptable QoS, and excess traffic to provide a better QoS.

The NETWORLD shares the idea of a service broker that negotiates a service contract for the delivery of data at a specified quality, with the QoS Broker.[8] The QoS Broker is an intermediate between the application and both the operating system and the network protocol subsystem. In NETWORLD, the functions of the QoS Broker to arrange for the delivery of data from source to destinations and to request reservations from the network, are performed by the remote access manager in each VU with the help of DSB. QoS Broker assumes *Tenet's* Real-time Channel Administration protocol[7] to be the network subsystem whereas NETWORLD uses the more flexible $\mathcal{V}-$NET.

Another difference between our framework and both QoS-A and QoS Broker is that NETWORLD takes into consideration potential incompatibilities between the data source format at a DR or VU and the data presentation format at a VU, and arranges for transformation of data format at intermediate sites in an application transparent fashion. Further, it performs data transformations into suitable formats to enforce the real-time requirements and to minimize the cost of data transfer. Also, our QoS GUI includes several means of specifying user level parameters, making mapping to the application level easier.

4 QUALITY OF SERVICE

In this section, after precisely stating our definition of QoS, we detail how the user specifies QoS through our graphical user interface. We define a *session* as the interval from the start to the end of a multimedia application or service. We then explain how to translate the user's specification into a *QoS contract* for the session, which is sent to the system. Lastly, we describe how the system realizes this contract, and briefly detail negotiations.

4.1 QoS Parameters

We divide the QoS parameters into the following four groups:

- *Media Parameters* – These are qualitative and quantitative characteristics of the involved data. Qualitative characteristics include the type of the media (e.g., live audio, taped audio, graphics, still and motion video) and the desired format (e.g., for video, raw, MPEG-I or MJPEG) that the player at the VU can understand.

  Quantitative characteristics are media specific values in terms of which the quality of a media is expressed. For example, for live audio, quantitative characteristics include *sampling rate, number of bits per sample* and *number of channels.* For taped video, quantitative characteristics include *resolution* (pixels/frame), *speed* (frames/sec), and *color depth* (color bits).

- *Delivery Parameters* – Delivery parameters, also known as real-time performance requirements, include *start time*, *end time*, the source to destination *packet delay*, and *delay-jitter.*

- *Value Parameter* – In addition to actual quality of the data delivered at a site, we also consider the *monetary cost threshold* that an application is willing to pay for a specific combination of media, delivery and fault-tolerance parameters. The price charged for each session depends on: (1) The cost of establishing the session (i.e., cost of reserving the required resources); (2) actual cost of retrieval, transfer, transformations, delivery and presentation of data (i.e., cost of utilizing the resources); and (3) the base cost of the data (i.e., cost of producing the media).

  These costs are not fixed but they are determined on a per-session basis. Time of the request and type of session are of crucial importance in determining the cost, as well as the datatype being requested.

- *Fault-Tolerance Parameters* – These define different classes of service commitment or traffic as well as different types of redundancy that can be applied to provide extra tolerance to faults. The classes of service include *basic* that is required traffic to operate properly (e.g., MPEG I-Frames), *enhancement* traffic to achieve acceptable QoS (e.g., MPEG B & P frames), *excess* traffic to provide for a better QoS and *datagram*

traffic that offers no real-time guarantees. Some types of redundancy increase latency (e.g., primary/backup site or segment), but cost less than others (e.g., modular redundancy). Due to space limitations we will not elaborate on fault-tolerance parameters in our description below but we will only consider *on-time reliability*, i.e., percentage of the data guaranteed to arrive on time.

Clearly, all the above QoS parameters cannot be specified by the user. The nature of each QoS parameter may be one of the following: *fully specifiable by the user*, *fully system specifiable* and *specified by the user, but system dependent.* Most of the media parameters, except ones dealing with the actual format of the media, are fully specifiable by the user. The value or cost threshold is also fully user specifiable. The fault-tolerance parameters are specifiable by the user but they are often restricted by the characteristics of the system.

4.2 Network System Parameters

In our approach, the system parameters are specified in a way similar to DASH, with a Linear Bounded Array Process (LBAP),[1] supporting a variety of traffic rate descriptors. A LBAP is a mechanism which specifies the maximum number of packets an application can generate over any interval of time. A LBAP is defined by the maximum burst size m, an interval r, and the number of packets n, generated per interval r. Based on these values, the application's long-term packet rate is $\frac{n}{r}$, but over any interval of time, the application can be in excess of this rate by no more than m packets. This specification is useful to packet based applications because it allows characterization of a long-term rate with burstiness.

Allowing an application to specify its traffic rate parameters, however, is not sufficient to infer the worst case arrival pattern of the generated traffic. The exact characterization of such a pattern depends on the policing mechanism used to monitor the traffic and is used by the admission control procedure to verify the feasibility of supporting the real-time requirements of the application. Thus, in addition to specifying values for r, n, and m, the application must specify the packet policing mechanism to be used in enforcing its rate. One interesting feature of LBAP is that the specification of the three traffic rate parameters m, r, and n provides sufficient information to specify the worst-case packet arrival patterns generated by several policing mechanisms, including Moving Window (inferred $m = 0$), Jumping Window (inferred $m = n$), a Peak Rate Controller (inferred $m = 0$), Leaky Bucket, and Token Bucket (user-specified m values).

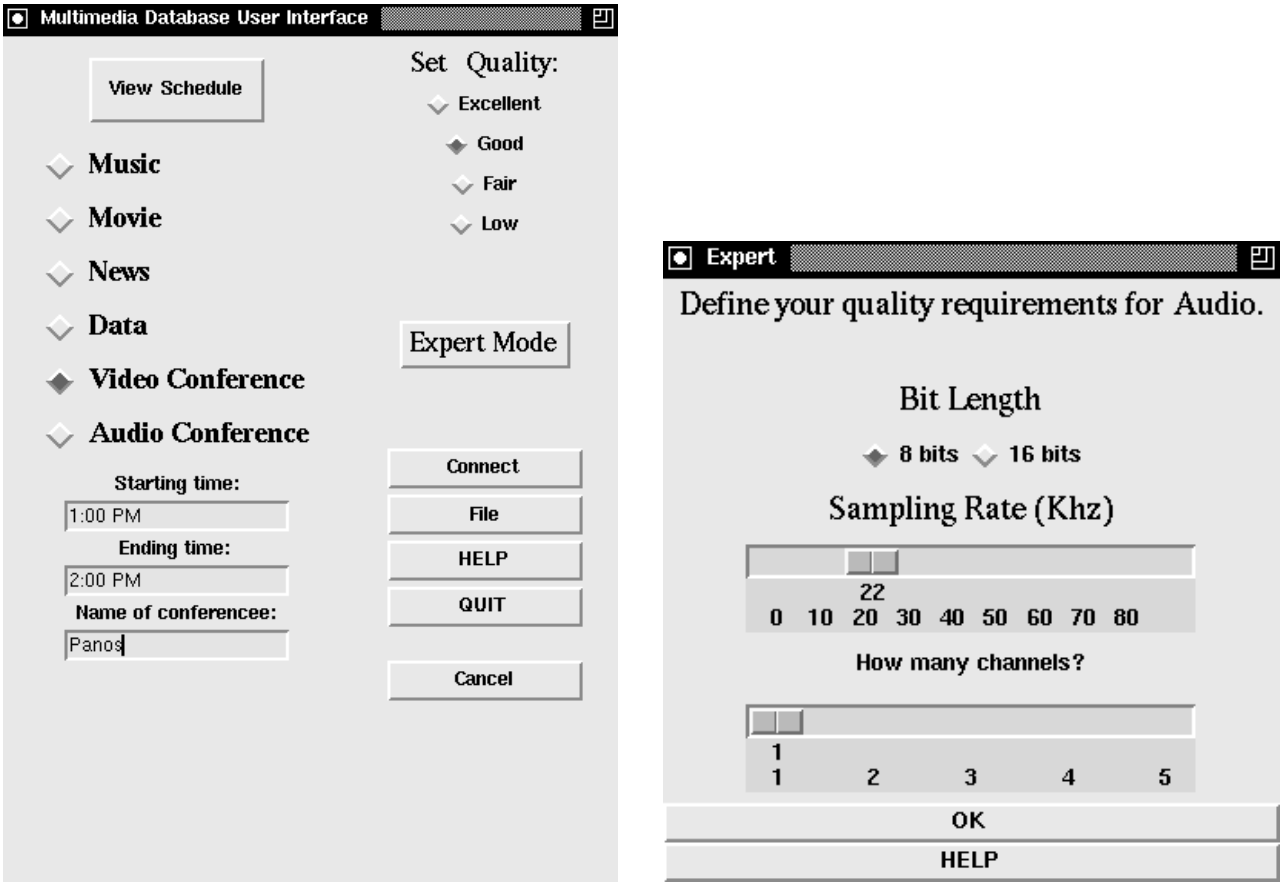
4.3 User Specification of QoS

In a distributed multimedia application, users will be expected to provide values for the user specified QoS parameters. However, most users will not have the technical knowledge[†] and/or patiences to specify each and every media, delivery, cost and fault-tolerance parameter individually. Thus, as mentioned in the introduction, users must be provided with a VCR-type interface that supports human-friendly specification of the desired QoS. We have designed such a user-friendly interface and implemented it in our prototype in the form of a graphical user interface (GUI) to the VUs, as described in this section.

Our user interface is menu-driven that requires minimum typing for the specification of information to be retrieved. The specification of the QoS parameters including the value parameters of the multimedia presentation are primarily achieved via pointing and clicking on appropriate selections, dragging sliding rulers and turning knobs (see Figure 2).

Our user interface supports both expert (Figure 2b) and non-expert (Figure 2a) modes. Thus, while the simplicity

[†]In the movie "City Slickers" one of the people attempts to explain to another how to fix the VCR clock, so that it does not blink. The reply: "Isn't it supposed to blink?"



(a) Quality Based Mode

(b) Audio Expert Mode

Figure 2: User Interface

of the graphical interface facilitates the user's task of specifying the QoS parameters, it also provides freedom to the more expert users to specify directly the QoS parameters. For example, the user interface allows users to specify the delivery parameters. In the absence of user specified delivery parameters the system will fill in appropriate values considering the type of media (e.g., the system will place a reasonable delay on live media).

The user will perceive quality mainly through selection of the media and value parameters. The type of media must always be specified by the user and is done indirectly in our interface (e.g., specifying music selects audio with a high sampling rate as the media type). The other media and cost parameters are determined in one of these three ways, two of which are targeted for all users and one for expert users.

1. *Cost based* – This option presents the user with a knob for controlling price only. The user expects the system to produce a tentative QoS contract of the highest quality for the specified cost threshold. The GUI then displays a sample of this media at this QoS which the user either accepts or rejects.

2. *Quality based* – This option allows for four choices for the quality of the presentation: *poor, fair, good,* and *excellent* quality. After the user chooses the quality, the interface shows to the users the price of the connection, according to the availability of network resources, quality of presentation, and available hardware in the user's Viewing Unit. For example, if the user's VU contains a hardware MPEG decoder, the quality of presentation can be enhanced with decreased cost, since less NETWORLD resources to move

the compressed data will be needed.

3. *Expert* – This third alternative is similar to the second, allowing an advanced user more freedom in the specification of quality at a level between the user-level and the LBAP parameters. On this level the user specifies each media parameter individually. For an example of audio specification, see Figure 2(b).

4.4 QoS Contracts

Once the user has specified the complete information for a session using our GUI, this information is collected and translated to form a service contract, not unlike the service contract used by QoS-A.

Our service contract consists of:

Flow Specification – information of what is expected of the virtual channel

flow id: a unique identifier for the channel to be established

start time: time after which channel must exist

end time: time before which channel must exist

source: where data is coming from

destination: where data is going to

burst: m in the LBAP parameters

packets: n in the LBAP parameters

interval: r in the LBAP parameters

commitment: traffic guarantees on the channel

basic - data is guaranteed to make it 100%

enhancement - α percent of this data may be lost, or late, $(1 - \alpha)$ is guaranteed

datagram - data is guaranteed to make it, but within no real-time constraints

excess - data is in excess of what was reserved, no guarantees are made

Contingency – What action is to be taken in the event that a particular QoS guarantee cannot be met

event (packet loss, jitter, throughput, delay, disconnect)

action (renegotiate contract, notify, disconnect, none)

Connection – these are parameters used in the selection of an acceptable route for the establishment of a virtual channel from source to destination

cost limit: the maximum cost the user is willing to spend

max extension: the maximum number of sites to be considered when extending source and destination

max bandwidth: the maximum bandwidth on any link on the path from the source to destination.

source media format: set of formats of the data at the source

destination media format: set of data formats that can be interpreted at the destination

restrictions: list of sites a user wants to exclude

Flow id is a unique identifier for both the virtual channel and the contract. The value for the flow specification fields of start time, end time, destination and commitment are directly provided by the user interface. The selection of the source depends on the service. For teleconferencing, source is directly provided by the user interface. For stored data, source is selected based on the availability with the help of DSB. The remaining fields

of the flow specifications correspond to the LBAP parameters and must be derived from the information the user provides.

The derivation of LBAP parameters is performed in two steps: First, user's specifications (low, fair, good, and excellent) are translated into media parameters, such as frame size, frames per sec, etc. From these media parameters the LBAP parameters are computed. Translating to the media parameters is only a matter of selecting the appropriate parameters. In our current prototype, the translation is implemented using Tables 1 and 2. The tables were derived from the characteristics of the different available qualities of audio and video available and the capabilities of the hardware.

Quality	Comparable to	Sampling Rate (KHz)	Number Bits	Number Channels	BandWidth Raw (KB/sec)
low	metallic	Enhanced LPC		1	0.35
fair	phone	8.0	8	1	8.00
good	CD stereo	44.1	16	2	175.00
excellent	CD surround	44.1	16	5	437.50

Table 1: Audio Mapping Table

Quality	Comparable to	Resolution x Color depth	Frames/sec	BandWidth Raw
low	Low-res (B&W)	300x200x1	10	75 KB/sec
fair	B&W TV	640x486x1	30	225 KB/sec
good	Color TV	640x486x24	30	16.5 MB/sec
excellent	HDTV	2048x1024x24	30	190 MB/sec

Table 2: Video Mapping Table

The LBAP parameter of r can be derived depending on the user's specification of *type of media* (e.g., audio only, video, sound and video, textual data, etc) and the media parameters. If the datatype is video, we use an r equal to 1/(the frame rate). If the datatype is audio, we use a much smaller r which is 1/10 of the user's specified end-to-end delay. The other parameters (n and m) are computed using information about the data rate and burstiness of the data format provided by the DRs. (It is also possible to use estimates of data rates and burstiness from internal tables.) In our current prototype, default values of burst (m) used are be equal to the 15% of the average rate and package sizes of 48 bytes.

The ⟨event,action⟩ pairs of the contingency field are directly provided by the user interface (not shown). The default action for each event is `notify` that keeps a VU informed for every deviations from the contract. Once an action is specified for an event, the system closely monitors that event. On the other hand, the system does not perform any monitoring in the case that **none** is associated with all events. Note that this field captures the adaptation and maintenance fields in the QoS-A service contract.

The connection field of the service contract contains information used by our *route selection* algorithm that selects a route for the establishment of a multisegment virtual channel. The specification of cost limit, max extensions, and max bandwidth are all feasibility parameters for the selection of a route. Format incompatibilities between the source and destination sites and the need for data transformations are captured by the source media formats and destination media formats. These can be found in the *captab (Capability Table)* at the source and destination. Finally, the restriction field simply allows the user to select sites or groups of sites to be excluded from consideration during the selection of a route, for whatever reason.

When the user interface requests composite data containing more than one media (i.e., a movie may contain video, audio, and textual closed captioning), the different media type may require multiple channels with different parameters. For example, the delays for audio must be more stringent than those for video, due to the human capacity for perceiving such differences. In such a case, a separate contract is created for each media and all of them must be fulfilled.

4.5 Contract Realization

After the user has fully specified his/her requirements for the system, it remains to be verified if the system can fulfill this contract. It is at this point where our route selection algorithm attempts to find a route and make reservations along that route. Unlike other routing schemes that only consider finding a path from source to destination, our algorithm also implements a mechanism that takes into consideration the hardware in the user's VU, the data representation at the data repositories, as well as the hardware resources available in the path from the source to the destination of the data.

For example, assume that a source maintains a video clip in MPEG format, the destination has no hardware MPEG decoder, and the CPU at the destination is not powerful enough to decode the video clip in software. In that case, our algorithm attempts to find another node to decode the video clip, and reserve the resources needed for the decoding and transfer of data in the format the VU can handle or as raw data (at higher traffic rates).

Thus, given a contract and a graph of possible routes generated by one of the existing routing schemes, our algorithm finds a good route in a breath-first manner from both the source and destination. An exhaustive search including all possible routes/transformations from source to destination would yield the best route but it would be too costly.

Route Selection Algorithm:

1. For each candidate route i, construct two list of formats: $\texttt{sf-list}_i$ for the source and $\texttt{df-list}_i$ for the destination, and initialize them from the information in the source media formats and destination media formats fields of the contract. Also, create two sets of sites, *extended source* (ExS_i) and *extended destination* (ExD_i) and inserts source in ExS_i and destination in ExD_i.
2. Add each route i on a candidate list, if there is a common format on the $\texttt{sf-list}_i$ and $\texttt{df-list}_i$. If the candidate-list is not empty, send the candidate list together with the values of packets, burst and interval fields of the contract to the multisegment virtual channel reservation mechanism (MVCR).[3] The MVCR attempts to make a reservation along one of the routes on the candidate list and returns either a single route that can be used to establish the desired multisegment channel or no route if none is found. If a route was not found, it means that the cost, media, or delivery parameters could not be satisfied.
3. If a route was not found in Step 2, then somewhere along the route the data must be transformed. A site for the transformation(s) is found on a route i by extending the $\texttt{sf-list}_i$ and $\texttt{df-list}_i$ with new formats that can be derived (via transformations) as follows:
   - Insert in ExS_i (ExD_i) all the sites adjacent to current sites in the ExS_i (ExD_i) in the direction to the destination (source).
   - Extend $\texttt{sf-list}_i$ ($\texttt{df-list}_i$) with all the formats that can be derived from the current $\texttt{sf-list}_i$ ($\texttt{df-list}_i$) via transformations at the newly inserted sites in ExS_i (ExD_i).
4. After an extension, step 2 and 3 above are repeated until either a route that satisfies the contract can be reserved or the contract field of maximum extensions has been reached.

When the route selection algorithm executes in the *negotiation* (interactive) mode, each time that step 2 fails to produce a good route, the algorithm initiates a negotiation of the cost threshold and maximum bandwidth.

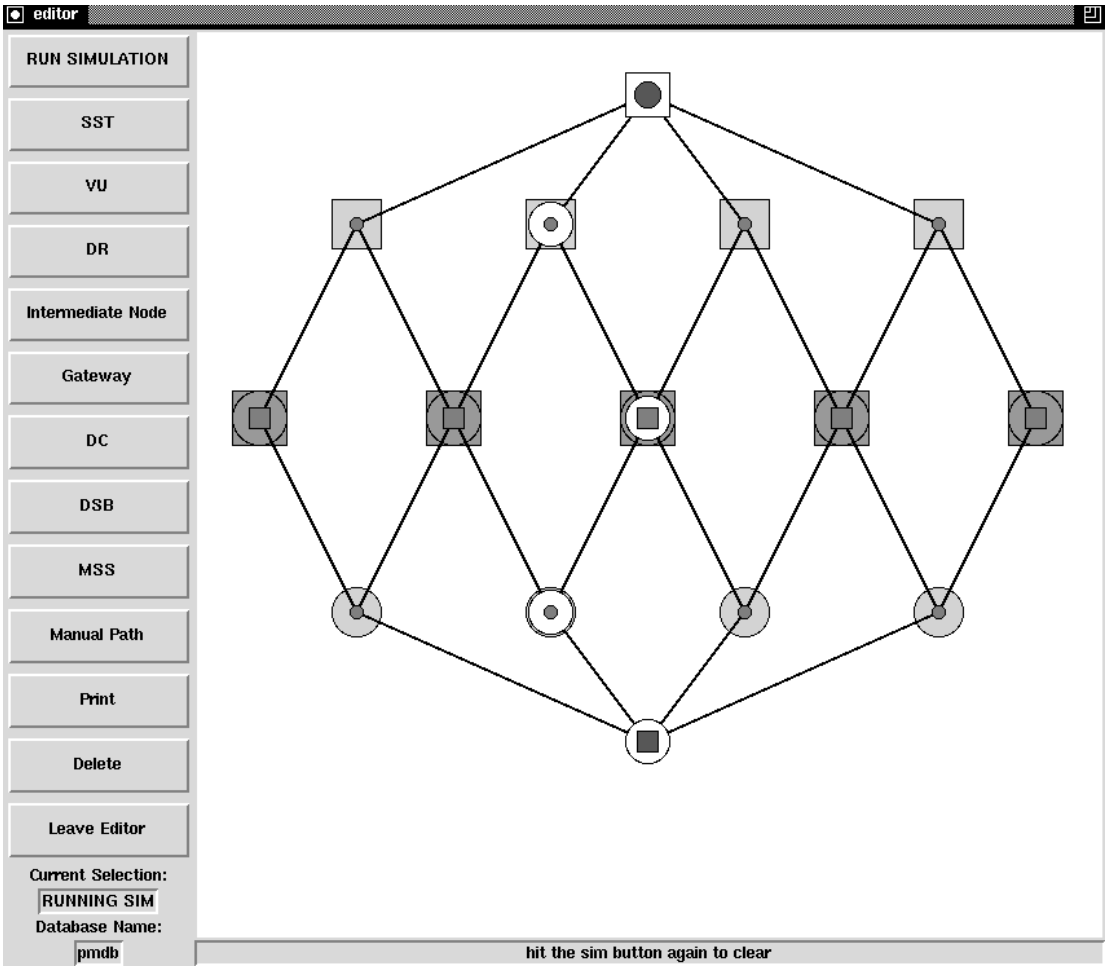


Figure 3: Network Configuration Editor

Similarly, after step 4 terminates without discovery of a good route, it prompts for larger maximum extension value.

Figure 3 shows our configuration editor visualizing and animating a simulation of our route selection algorithm. We selected the bottom-center node to be the source DR and the top center node as the destination VU. The DR format was specified as MPEG and the VU node was made to support raw data. The only node in the network capable of any transformations is the middle node (a SST), which can transform MPEG to raw. The light grey squares (circles) around the nodes are the first extension of the source (destination) and the darker grey represents the second extension. The trail of white circles leading from DR to VU represents the route that the simulation returned.

5 STATUS AND CONCLUSION

In this paper, we reviewed our system architecture, NETWORLD, and discussed how QoS is specified and supported within NETWORLD. Users specify their QoS requirements by means of a simple, yet flexible, user interface, suitable for both expert and non-expert users. We then discussed how user QoS specifications are formulated

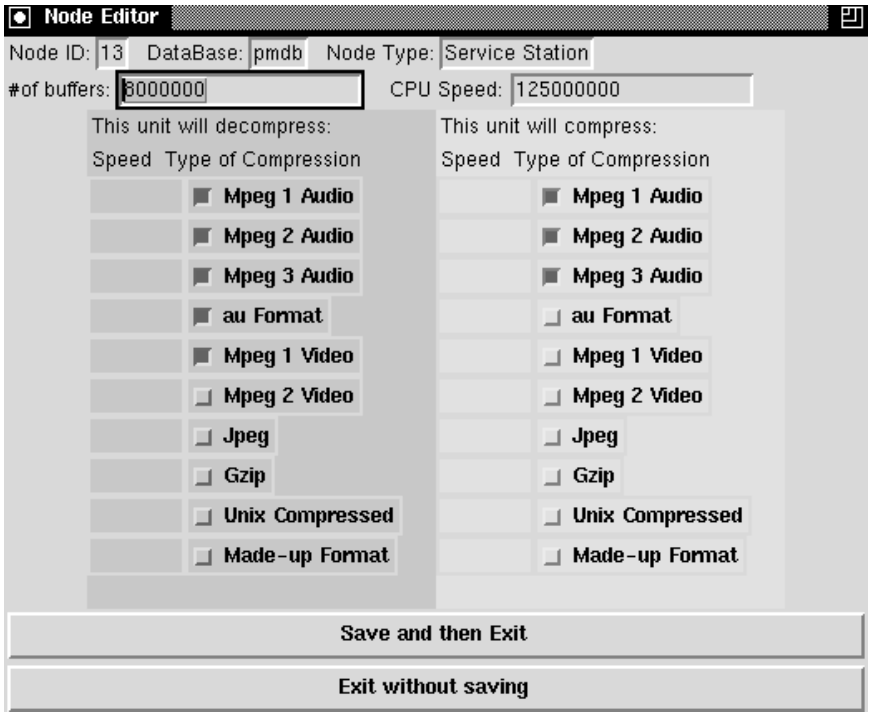


Figure 4: Capability Table Editor

into a QoS contract that the system is expected to fulfill. Finally, we presented a route selection algorithm that validates a contract for a specific set of routes between a source and a destination.

For experimenting with QoS of multimedia services, we have been developing a prototype of our NETWORLD on Unix Workstations running X-Windows using *C* and Tcl/Tk[11] (with Postgres extensions). Internally, Postgres[10,12] is used to store all the data in the system, allowing distributed access.

We have completed the implementation of our graphical user interface (described in Section 4.3 and shown in Figure 2), the Network Configuration Editor (NCE) (Figure 3), and the route selection algorithm, and we are currently working on the multisegment virtual channel reservation mechanism which is based on $\mathcal{V}$–NET.

Using NCE, we can visually lay out the topology of a NETWORLD that we would like to experiment with. This can be easily done by placing nodes of particular type selected from a menu in the editor's windows and appropriately connecting the nodes with communication links. (In this respect NCE behaves similar to a graph editor). By clicking on a site, a menu-driven Capability Table Editor (Figure 4) pops up to allow the specification of the CPU speed, the memory size, and the transformations the site is capable of performing. Similarly, by clicking on link, a menu-driven editor is poped up that allows for the specification of the bandwidth of the link. As mentioned above, NCE stores all the configuration parameters in a database managed by Postgres.

The NCE offers also a *run simulation* option which animates the execution of the route selection algorithm using QoS parameters and routes stored in the database. The NCE supports the selection of specific routes in a topology to be used for an experimentation.

The user interface is our vision of how users of future distributed multimedia applications will formulate their QoS requests. Currently, we are incorporating in our user interface the notion of a *user profile* to make access as easy as possible. A user-profile simplifies the interface, since the user will not need to enter the same information

repeatedly. Further, we investigate the use of user profiles to effectively retrieve data in a periodic fashion.

With respect to negotiation of QoS parameters, we are currently investigating ways to enhance the negotiation mode of our route selection algorithm. Specifically, we explore methods to decrease the cost of a service if a user, for example, is willing to negotiate the time the data is to be delivered, or if the data can be delivered in advance and stored locally.

6 ACKNOWLEDGMENTS

This work was performed when the first three authors were undergraduate research assistants funded by REU awards associated with NSF RIA grants CCR-9308886 and IRI-9210588.

7 REFERENCES

[1] D. P. Anderson, R. Wahbe S. Tzou and, R. Govindan, and M. Andrews. "Support for Continuous Media in the Dash System," *Proc. of the 10th International Conference on Distributed Computing Systems*, pages 54–61, May 1990.

[2] A. Campbell, G. Coulson, and D. Hutchison. "A Quality of Service Architecture," *Computer Communications Review*, pages 6–27, April 1994.

[3] P. Chrysanthis and D. Mossé. "Management and Delivery of Multimedia Traffic", *Proc. of the Second IEEE International Workshop on Community Networking*, pp. 189–196, June 1995.

[4] D. C. Clark, S. Shenker, and L. Zhang. "Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism," *SIGCOMM '92*, pages 14–26, 1992.

[5] B. Field and T. Znati. "α-channel: A Network Framework to Support Appliction Real-Time Performamce Guarantees," *IEEE Journal on Selected Areas in Communications*, 11(3):1317–1329, October 1993.

[6] B. Field, T. Znati, and D. Mossé. "$\mathcal{V}-$NET: A Framework for a Versatile Network Architecture to Support Real-Time Communication Performance Guarantees," *Proc. of InfoComm*, April 1995.

[7] B. Mah. *A Mechanism for the Administration of Real–Time Channels.* Master's Thesis, University of California at Berkeley and International Computer Science Institute, 1993.

[8] K. Nahrstedt and J. M. Smith. "The QOS Broker," *IEEE Multimedia*, pages 53–67, Spring 1995.

[9] OMG ORBTF (Object Request Broker Task Force). "Common Object Request Broker Architecture," *Technical Report, Object Management Group*, 1992.

[10] The POSTGRES Group *The POSTGRES User Manual* Computer Science Division, Department of EECS University of California at Berkeley, 1994.

[11] J. K. Ousterhout. *Tcl and the Tk Toolkit* Addison-Wesley, 1994.

[12] L. Rowe and M. Stonebraker. The POSTGRES Data Model Proc. 1987 VLDB Conference

[13] D. Verma, H. Zhang, and D. Ferrari. "Delay Jitter Control for Real-Time Communication in a Packet Switching Network," *Proceedings of TriComm '91*, pages 35–43, 1991.