Quality of Service for Networked Virtual Environments

Denis Gracanin, Yunxian Zhou, and Luiz A. DaSilva, Virginia Tech

ABSTRACT

A definition and measurement of quality of service for networked virtual environments (NVEs) presents many challenges, in part due to subjective and qualitative description of users' preferences for NVE-based applications. QoS can be provided, in general, as a per-application service, a network layer option, or both. Many of the proposed QoS architectures for NVE applications include the per-application service approach. Most of them focus on video and audio components with less emphasis on user interactions and user-level QoS. A review of architectures and approaches to QoS for NVEs identifies related issues. Those issues include user-level QoS and mapping between user and application-level QoS, as well as the relationship between application-level QoS and network-level requirements. An approach is described that, for a given user-level QoS value, provides a reduction in network requirements by optimizing task or application-level scenarios. An example based on the DIVE testbed illustrates the approach.

INTRODUCTION

One possible definition of a networked virtual environment (NVE) is given by Singhal and Zyda in [1], where an NVE is defined as follows:

> … a software system in which multiple users interact with each other in real time, even though those users may be located around the world.

In other words, users can share information among themselves and manipulate objects in a shared virtual environment. Because an NVE is an integrated system composed of many components (distribution, graphics, and interaction), there are multiple challenges regarding individual components or their integration. Those challenges include network bandwidth requirements, heterogeneity of applications, support for distributed interaction, real-time system design and resource management, and scalability [1]. The study of quality of service (QoS) in NVE systems has been an active research area of late.

The concept of network QoS evolved from the realization that in networks carrying heterogeneous traffic, it makes sense to treat specific classes of traffic according to their specific needs. An important trade-off is that the greater the statistical multiplexing capabilities of the network, the greater its efficiency and also the variability of achieved performance; in this sense, best effort service provides maximum efficiency with highly unpredictable service quality. The objective then becomes to come up with traffic management techniques (classification, scheduling, policing, admission control, prioritization, etc.) that allow the network provider to meet individual user requirements with maximum efficiency and flexibility to variations in traffic, topology changes, and user demand. Table 1 provides examples of QoS mechanisms that can be applied at various layers.

QoS is described by the British Standards Institute as "the totality of features and characteristics of a product or service that bears on its ability to satisfy stated or implied needs." Therefore, quality should be measurable and perceivable to the user, and service should match user needs. Performance of different applications, different flows belonging to the same application, or even subsets of packets belonging to the same flow can have distinct impacts on user perception of quality. This recognition has led to service differentiation based on traffic flow, application, or task. An application of task-based differentiation to NVEs is described in [2].

Quality perceived by a user tends to be significantly impacted by the QoS support (or lack thereof) provided at various layers of the protocol stack. The end-to-end path of a traffic flow will typically traverse multiple dissimilar networks: access networks offering a wide range of data rates, wireless segments with significantly variable channel conditions, and core networks operated by different providers. The perceived quality will clearly be a function of the QoS experienced in each segment of the path. The concepts of QoS mapping across the protocol stack and QoS concatenation are illustrated in Fig. 1.

Networked virtual environments impose some unique QoS requirements from the network point of view. The number of participants in such environments may be large, and they may be distributed around the world, so scalability is important. Stateful mechanisms, such as those that require strict reservation of resources and per-flow treatment, may not be appropriate due to both the scalability requirement and the need for the system to run over the public Internet. The high degree of interactivity in virtual environments means that most of the associated traffic flows should be treated as real time, with strict delay requirements; retransmission of lost packets is not feasible for such flows. Efficient multicasting techniques are also needed; these multicasting techniques must be appropriately integrated with service differentiation techniques at the network layer and achieve desirable QoS objectives, as well as reliability and ordering. Consistency of shared virtual objects requires a level of synchronization among a multicast group that is not present in many "traditional" network applications.

The remainder of this article is organized as follows. We review and describe QoS for NVEs. Then we describe QoS mapping and architecture, and provide an example based on the DIVE system [3]. Finally, we conclude the article and provide directions for future work.

MEASURING QUALITY OF SERVICE

There are many different services provided to a user in NVE-based applications. Those services can be defined at the user preference, user performance, and task performance levels. Evaluation and measurements are based on subjective and qualitative descriptions of user preferences. Mapping user perception of quality to objective metrics that can be observed and controlled at the network or data link layer poses a significant challenge. It is clear that user satisfaction will correlate with end-to-end delay, jitter, packet loss rate, and throughput. The sensitivity of individual applications to each of these factors, however, may vary significantly. For instance, packet losses (say, due to playback deadline misses) in video transmission, if kept at a reasonably low level, may go virtually undetected by users, while the same losses would significantly impair interactivity if they occur on control packets in an interactive game application.

Layer	Examples of QoS mechanisms
Application	Caching, prefetching, task prioritization
Middleware	Marking, requests for resources, reservation, classification of traffic flows/subflows
Network	Scheduling, prioritization, shaping, admission control, flow control
Data link	Mediation of link access, prioritization
Physical	Forward error correction, code/slot assignment

Table 1. *Examples of QoS mechanisms.*

Subjective testing is a well-established method of determining user reaction to network conditions, and has been widely used in traditional telephony. Such tests can result in a long, costly, cumbersome process, especially if applied in a collaborative application context. One approach to mitigate this problem in an NVE is to test subjective reactions of one user to *N* simulated users. The parameter space must be tightly controlled to achieve generalizable results. There have been experiences with using psychometric methods to assess subjective quality in networked applications, as well as some large-scale experiments with testing user reaction to service level and pricing. While these help shed some light on the correlations between perceived quality and network conditions, results to date have generalized rather poorly. Innovative visualization techniques would also help in determining appropriate subjective mapping to the variety of network parameters impacting quality.

QoS mechanisms present in the various layers of the protocol stack need to be integrated to achieve the desired user-level quality objectives. For instance, prediction, prefetching, and caching are techniques typically associated with the application layer; these can be combined with scheduling and prioritization mechanisms

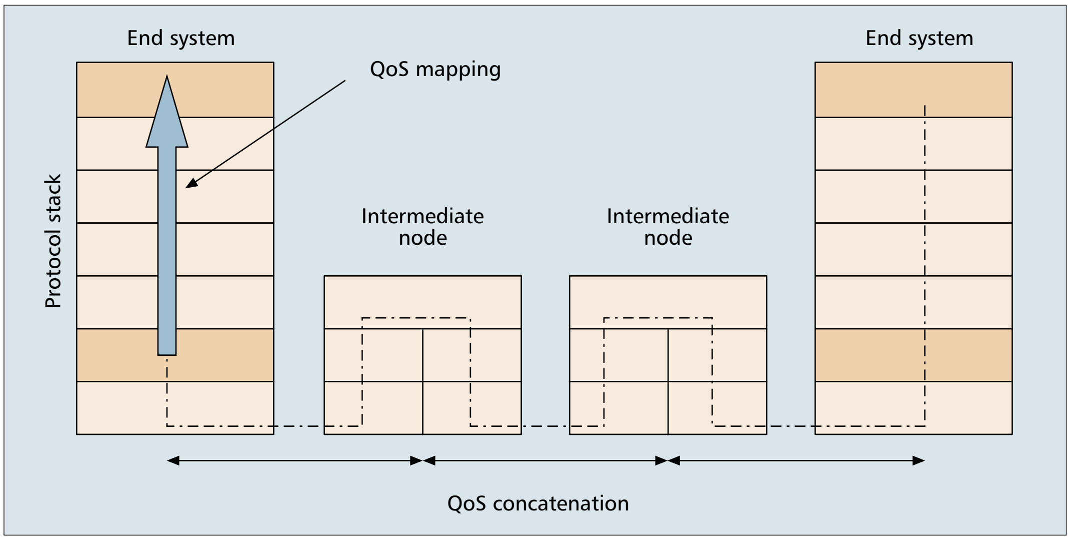


Figure 1. *An illustration of QoS mapping and concatenation.*

The QoS metrics at the user level are NVE features experienced or sensed by the user during active participation in the NVE system. They are mostly subjective and qualitative, such as immersion, presence, comfort, learning time, and overall satisfaction.

implemented at the network and medium access control layers to increase interactivity, inclusion, and other objectives of NVEs. Such cross-layer approaches are still insufficiently explored in the literature.

Increasingly, participants in an NVE may be mobile, connecting to the network through wireless links. The dynamic nature of the wireless channel, subject to fading, shadowing, and interference, will impact quality, and mobility adds its own set of challenges. Address redirection, re-authentication and authorization, context transfer, and session maintenance are some of the issues that must be addressed to allow seamless participation by mobile nodes in a virtual environment.

A robust distributed framework for network security and QoS management is another requirement. This framework must support authentication and authorization of users, and differentiated treatment based on pre-established policies. Policy-based network management is a strong candidate, and recent progress has been made in establishing the protocols and procedures for this in both the Internet and more dynamic environments such as mobile ad hoc networks. In an NVE scenario, there are different QoS metrics corresponding to distribution, graphics, and interaction components. We identify and summarize QoS metrics at three different levels (user, application, and communication) following a top-down approach.

The User Level

The QoS metrics at the user level are NVE features experienced or sensed by the user during active participation in the NVE system. They are mostly subjective and qualitative, like immersion, presence, comfort, learning time, and overall satisfaction.

Immersion characterizes the extent to which users feel part of an NVE. It can be evaluated through some objective parameters, such as extensiveness, surround, inclusion, vividness, and match.

Presence is the most important user-level metric. It is a psychological feeling of "being there," that is, the extent to which the user is in the center of the virtual world. Virtual stimuli will have a synthetic impact on human beings. The stimuli help to construct a mental model of the physical environment. Many factors contribute to presence, including degree, immediacy and mode of control, anticipation of events, physical environment and modifiability, sensory factors, distraction factors, and realism factors such as scene realism, information consistent with the objective world, meaningfulness of experience, and separation anxiety. There are questionnaires and physiological parameters for measuring presence; these include heart rate, sweaty palms, and breathing.

The Application Level

The QoS metrics at the application level can be classified into four categories.

The first category relates to system or task performance, which studies performance of the connection between user and application. That performance is described through total latency, robustness (fault tolerance), spatial/temporal synchronization, task completion time (efficiency), task accuracy, and application-dependent metrics.

The second category includes performance metrics related to graphics and output technologies that are part of human-computer interaction. Those include image rendering, acoustic rendering, tactile and haptic feedback, and other related techniques (e.g., olfactory display).

The third category includes performance metrics for input devices. Virtual reality systems are highly interactive, and the system simulation is based on the data from input devices. The performance of the input devices is important and sometimes crucial to the overall system performance. Position/orientation tracking devices are attached to a user for visual and auditory updates according to head and body position and orientation. The desired QoS metrics for positional trackers include resolution, accuracy, lag, update rate, robustness, quality of registration (calibration), working volume, and sociability and encumbrance. Data gloves use bend sensing technologies to detect hand postures. The QoS metrics, similar to position tracking, include robustness, accuracy and resolution, update rate and latency, calibration requirements, hygiene, and encumbrance. Locomotive systems let users pedal or walk using devices like stationary bicycles, stair climbers, and passive treadmills, then sense and use these activities to navigate in a virtual world. QoS metrics include ease and freedom of operation, responsiveness, strength required to operate the device, and other metrics similar to those for tracking systems. Additional input devices include eye tracking, biosignal sensors, haptic devices, and gesture and speech recognition.

The fourth category includes architectural or distribution considerations. It represents the application as a whole and the connection between the application and the underlying network. Since an NVE system is a highly integrated system with various parts including distribution, simulation, and interaction, its architectural characteristics have a great influence on overall system performance. Some architectural considerations common to networked systems include heterogeneity, scalability, and openness (extendibility, standardization), fault tolerance, transparency, and adaptability.

The Network/Communication Level

The QoS metrics at the communication level can be defined based on bandwidth, latency, jitter, packet loss rate, reliability, ordering, throughput, and related parameters [4]. The network communication includes video, audio, and other (VE) data. Typically the video/audio data flows occupy large network bandwidth, while VE data are small but frequent. Previous research on NVE systems has focused on decreasing network communication costs [5]. One approach is to prioritize object (data) transfer based on importance of presence (IOP), which is related to the distance, angle, and direction between user and object [6]. Another approach uses a QoS architecture management of streamed video within collaborative virtual environments (CVEs) [7] and a spatial awareness model (nimbus, focus, and third-party objects).

NVE QoS Mapping and Architecture

A general QoS framework includes QoS principles that govern the construction of the framework, QoS specification that captures application-level QoS requirements, and QoS mechanisms that realize the desired application end-to-end behavior.

QoS Mapping

Network requirements can be assessed and met by mapping between different NVE layers [8]. QoS translation is the process of translating QoS requirements (delay, throughput, losses, etc.) into specific resources (buffers, bandwidth) needed to meet such requirements.

In a layered approach, QoS mapping consists of two aspects: first, how to set QoS objectives in a lower layer to satisfy higher-layer QoS requirements; second, given the expected performance at a lower layer, determine its impact on higher-layer QoS. A preliminary study of QoS (delay, loss) mapping based on segmentation/fragmentation is presented in [8]. QoS mapping for a virtual reality system and mapping of costs to user satisfaction are also discussed in [9, 10].

QoS mapping for the Networked Virtual Reality (NVR) system [9] provides a QoS mapping mechanism from application level to network level:

•The QoS adjuster calculates the IOP values based on the distance and angle from user to object for both graphic objects and video objects.

•The QoS controller assigns a level of detail for graphic objects, and assigns units of CPU time and network bandwidth or temporal/spatial resolution for video objects in proportion to their IOP values.

•The system monitor observes CPU resources at both server and client sides, as well as network bandwidth, then sends feedback to the QoS adjuster. User-level QoS mapping from a single parameter [10] maps the cost into user satisfaction and then configures the system accordingly. Some of the characteristics of that approach include:

•This study adopts a three-layer architecture. The user layer targets cost and satisfaction, the application layer contains frame rate, image resolution, coding distortion, and so on, and the resource layer contains network bandwidth, processor, and other metrics.

•User to application mapping is one to many. A logarithmic function is used for the relationship between single-application QoS and individual component satisfaction because the same increase in application-level QoS will have a greater impact on user satisfaction at lower levels of performance than at higher levels.

•A set of application-level QoS values is selected using a regularizing function to achieve a specific satisfaction level.

There are also some mathematical relationships among the different QoS metrics in an NVE system. Generally, this relationship can be expressed as

$$QoS\_Metric = F(Contributing\ QoS\ Parameters).$$

The function can be estimated based on empirical analysis. The coefficient values for the functions can be calculated from measurement data.

The Proposed QoS Architecture

Several QoS architectures have been proposed in the literature. A host QoS architecture integrating spatial model awareness is described in [7]. A QoS architecture based on IOP calculation is provided in [9]. Most of the architectures focus on audio/video streams, while the user presence, user preference, and task performance are less emphasized. A new QoS architecture (Fig. 2) is proposed that takes into account both "traditional" multimedia factors and user-related factors, and focuses on user interactions and user-level QoS.

A typical NVE system involves users, NVE application, and network. The proposed architecture has, accordingly, three main segments: *NetVE User*, *NetVE Application*, and *NetVE Network*.

Within the NetVE User segment, the *User/Avatar* module describes user control of an avatar in a virtual world. The sense of presence and network awareness are improved by providing some autonomy in lower-level behaviors to the *Avatar*. That autonomy can be used to simulate a large number of users and test the scalability of the application. Three types of profile are considered:

- *Avatar profile* contains information such as age, gender, character set, skill set, and location.
- *Group profile* defines group character, group skill, group role, and so on.
- *Task profile* specifies the complexity, goals, and requirements of given tasks.

The *Action model* is built on action rules, which are derived from avatar/task/group profiles. It can be modeled as a finite state machine or Petri net.

The NetVE Application segment includes basic components of a NetVE application. The *Human Computer Interaction* module handles input/output devices. The *Graphics/Multimedia* module is responsible for generating multimedia display content. The *Distribution* module manages network connections with other participants and maintains a dynamic shared state [1].

The NetVE Network segment handles user-, application-, and network-level QoS. The *User Survey* module collects data from user feedback and application parameters. The *Network Trace* module logs events and network packet transmission from the Distribution module. It parses the log into formatted data to calculate packet statistics such as payload distribution, interarrival distribution, loss, and error. Data from the User Survey and Network Trace modules are sent to the *QoS Mapping/Modeling* module that filters and classifies data for further analysis. The QoS Mapping/Modeling module then calculates mapping models based on refined data, and

A general QoS framework includes QoS principles that govern the construction of the framework, QoS specification that captures application-level QoS requirements, and QoS mechanisms that realize the desired application end-to-end behavior.

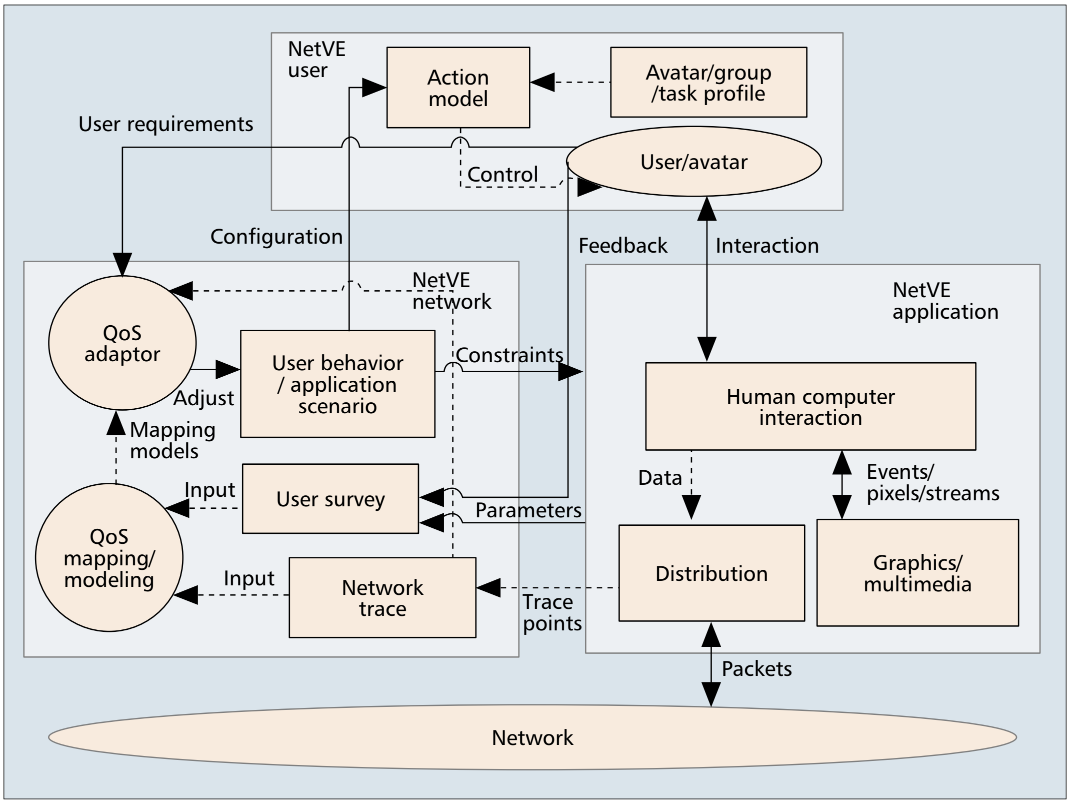


Figure 2. *QoS architecture.*

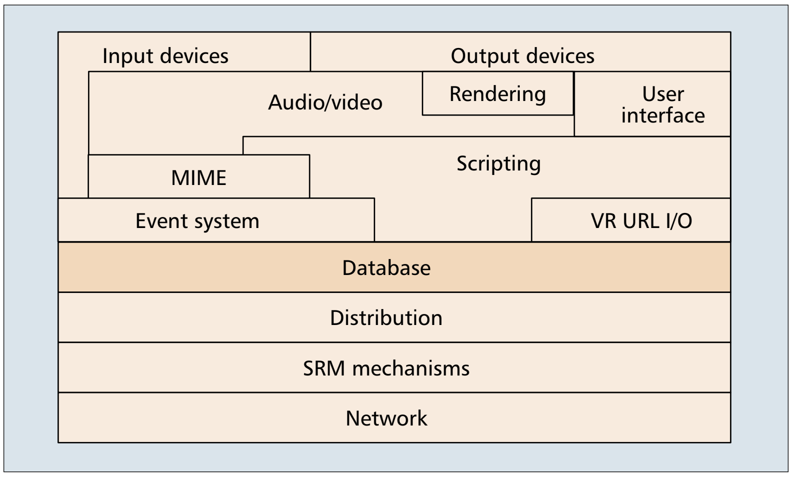


Figure 3. *DIVE architecture [3].*

establishes mathematical relationships among user, application, and network.

The *QoS Adaptor* module collects data from the user (user requirements), QoS mapping (mapping functions), and network (network configurations) modules, provides related solution sets, and adjusts the *User Behavior/Application Scenario* module.

The User Behavior/Application Scenario module enforces constraints on both the Action Model and NVE application, which indirectly control whether user actions are enabled. The Action Model module imposes constraints on avatar state changes. The User Behavior/Application Scenario module can be considered as constraints on application-level QoS parameters such as rendering interval and distribution rate. This QoS architecture contains two types of mapping: user vs. application and user vs. network. Fluctuations in network traffic (due to dynamics in replicated data change) can affect the consistency of shared data state. Network statistics and application state changes caused by user actions are used to calculate the mapping models. User or application behavior scenarios are then dynamically adjusted based on current network conditions.

Practical Issues

Instead of developing a new NVE application or system, publicly available and mature software platforms like DIVE [3] were considered (Fig. 3).

A software architecture, application-level QoS metrics, and network traffic characteristics were studied to establish the empirical relationships between the testbed application-level configuration and the user-evel QoS metric. Current network configuration including bandwidth, error rate, and network type must be also taken into account.

Implementation depends on the underlying software platform. For example, in DIVE, Tcl/Tk is used to write a user interface. A data analysis tool like Matlab or SAS can be used to derive different translation functions. Network traffic can be captured using tools like Sniffer or Ethereal.

The general process based on the proposed QoS architecture is as follows:

1) Identify application-level QoS metrics including parameters that can be dynamically configured and those that can be added to the

existing software. DIVE provides runtime options, and many of them contribute to user satisfaction. Those options include `token_bucket_size`, `loss_rate_send`, `render_fov`, and `render_lod`.

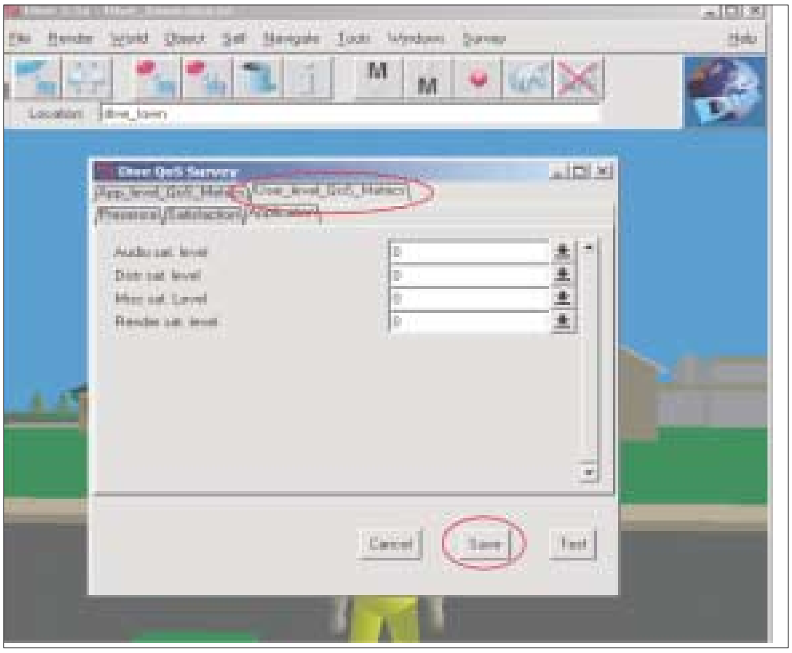
2) Create a user performance survey interface on top of the testbed for recording the QoS values from both the application and the user. Figure 4 shows part of the user interface for the survey, written in Tcl/Tk and based on the original DIVE code. The user first tests a set of application-level QoS metrics from the Audio, Distribution, Render sections. The user then records his/her satisfaction or presence level. The above steps are repeated with different combinations of QoS values and different users until enough samples are collected. To make the process more efficient, we may first single out the most relevant parameters, then automatically generate a set of parameter values either in a random way or in some order for users.

3) Create network trace client/server modules that record the generated packets statistics from the Distribution module, user actions, and network performance statistics.

4) Develop mathematical models (based on the data) to translate QoS values among user, application, and network levels. One user QoS value can be mapped to different sets of configuration values. This is because user-level QoS is a subjective metric resulting from a synthesis of different sets of application or network parameters.

5) For a given user QoS value, find different solution sets of application-level QoS values, and then measure the system resource consumption based on selected sets of values. Find the optimal set that requires the least system resource consumption while satisfying the same user QoS value. Similarly, identify the user behavior scenario that can achieve the best network performance. Sometimes constraints on user behaviors may affect other user-level QoS metrics like user satisfaction; then we need a trade-off mechanism to achieve the proper balance.

The analysis of collected measurements includes deriving functions for the mapping between the overall user QoS value and the specific user QoS values, and between the specific user QoS value and the application, as shown in Fig. 5. The three-level QoS translation connects the overall user QoS with the individual runtime options.



Audio sat. level 0
Dist sat. level 0
Misc sat. Level 0
Render sat. level 0

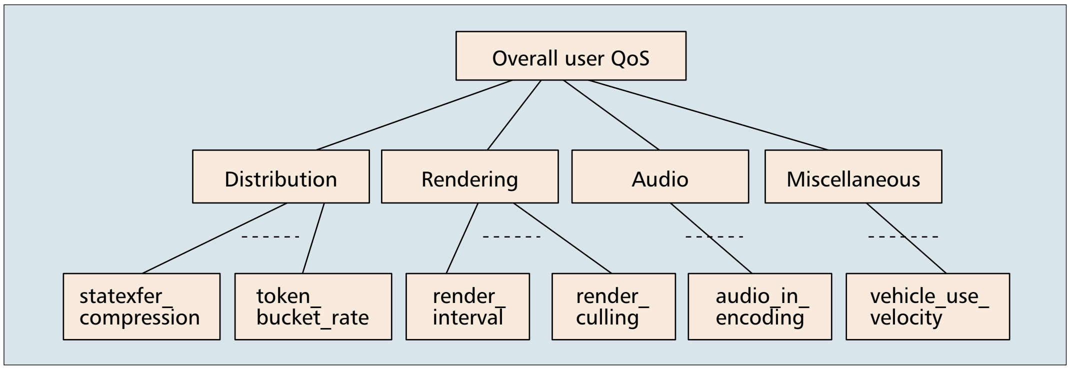
■ **Figure 4.** *The user performance survey.*

As an example, six runtime options have been selected, four from the distribution section and two from the rendering section. Variables x_1 to x_6 are `statexfer_compression` (compression level for sending state to the requesting peer), `token_bucket_size`, `token_bucket_rate`, `loss_rate_recv`, `render_fov`, and `render interval`. The user QoS metric is the user satisfaction level ranging from 0 to 9, and a linear approximation function is

$$y = 7.55 - 0.34\,x_1 + 0.0000277\,x_2 + 0.000391 \times x_3 - 0.0492\,x_4 - 0.02\,x_5 - 0.0333\,x_6.$$

Other forms of functions can be used using polynomial or nonlinear fit models. A more complex function does not necessarily imply better results. What really counts is the similarity of our assumption to the reality. In this example there is not much difference between the estimated value and the recorded value when the linear model is used. They differ only within a relatively small range. In this case the linear function is sufficient for the QoS mapping.

The QoS adaptation process starts by collecting functions mapping from different application QoS value sets to overall user quality. For a



■ **Figure 5.** *Hierarchical QoS translation.*

Other adjustable parameters such as interaction and collision frequency will be included and analyzed. Surveys can then be used to model user behavior and simulate NVE applications with a large number of simultaneous users.

given user QoS value, there may be multiple sets of corresponding application QoS values. In the previous example, for a given y value, there will be multiple sets of $x_1 \ldots x_6$ solutions. The results of the mapping process can be confirmed by comparing the result with the default setting and other settings, while also comparing the generated network traffic.

It may be possible to find optimal sets of QoS values that can require minimum network traffic or other resources from NVE components by testing applications and using a network trace tool. For example, the total latency has contributing factors such as input devices lag, state computation, network latency, graphics modeling, and image display. Task accuracy would be affected by input/output device accuracy, network delay, and network error/loss rate. Input device lag includes data generation lag, transmission lag, and network delay, which in turn contribute to image lag. Generally, the total latency should be kept less than 100 ms. Task accuracy and input/output device fidelity will affect sense of presence or satisfaction.

Conclusion and Future Work

Quality of service for networked virtual environments presents many challenges, from both the application development and application deployment/use points of view. New NVE applications should support users' preferences that are very subjective and qualitative. Mapping/translation among user, application, and network levels should be supported using an appropriate QoS architecture for the application. For already developed NVE applications, such an architecture can be, to some extent, added.

The described approach to QoS architecture and QoS mapping is based on the concept of QoS translation and mapping among user, application, and network layers. The mapping of QoS metrics between user and application levels is achieved by using summative evaluation techniques and analyzing the statistical results. In some cases a simple linear function is sufficient for QoS mapping. For a given user-level QoS value, network requirements can be reduced by optimizing task or application-level scenarios. The derived mapping function provides a set of application-level QoS values for a given user-level QoS value.

Our current work focuses on formalization of this approach and its use for various networked virtual environment software platforms. We will attempt to conduct the survey strictly based on the data collection steps provided in order to get more accurate values. Other adjustable parameters such as interaction and collision frequency will be included and analyzed. Surveys can then be used to model user behavior and simulate NVE applications with a large number of simultaneous users.

References

[1] S. Singhal and M. Zyda, *Networked Virtual Environments: Design and Implementation*, Addison-Wesley, 1999.

[2] M. V. Capps, "The QUICK Framework for Task-Specific Asset Prioritization in Distributed Virtual Environments," *Proc. IEEE Virtual Reality*, Mar. 2000, pp. 143–50.

[3] E. Frecon *et al.*, "The DiveBone — An Application-Level Network Architecture for Internet-Based CVEs," *Proc. ACM Symp. Virtual Reality Software and Tech.*, Dec. 1999, pp. 58–65.

[4] M. Matijasevic *et al.*, "A Framework for Multi-User Distributed Virtual Environments," *IEEE Trans. Sys., Man, Cybern. B*, vol. 32, no. 4, Aug. 2002, pp. 416–29.

[5] W. Broll, "Distributed Virtual Reality for Everyone — A Framework for Networked VR on the Internet," *Virtual Reality Annual Int'l. Symp.*, Mar. 1997, pp. 121–28.

[6] K. Fujikawa *et al.*, "C3: A Scalable Networked Virtual Reality System," *Proc. 17th IASTED Int'l. Conf. Applied Informatics*, Feb. 1999, pp. 268–71.

[7] C. Greenhalgh, *Large Scale Collaborative Virtual Environments*, Springer, 1999.

[8] L. A. DaSilva, "QoS Mapping along the Protocol Stack: Discussion and Preliminary Results," *Proc. ICC 2000*, vol. 2, June 2000, pp. 713–17.

[9] S. Oh, D. Kado and K. Fujikawa, "QoS Mapping for Networked Virtual Reality System," *Proc. SPIE Conf. Performance and Control of Network Sys.*, Nov. 1997, pp. 18–26.

[10] A. Richards *et al.*, "Mapping User Level QoS from a Single Parameter," *Proc. 2nd IFIP/IEEE Int'l. Conf. Management of Multimedia Net. and Services*, Nov. 1998.

Biographies

Denis Gracanin [SM] (gracanin@vt.edu) is an assistant professor of computer science at Virginia Tech. His research interests include distributed virtual environments and related network, application, and user issues. Current and recent research sponsors include NSF, NIH, DARPA, and Microsoft Research, among others. He is a senior member of IEEE and a member of AAAI, ACM, APS, SCS, and SIAM.

Yunxian Zhou (yuzhou1@vt.edu) is a computer science Ph.D. student at Virginia Tech. She received an M.S. degree from Fudan University, China, with a focus on multimedia communications. Her current research focuses on intelligent collaborative virtual environments. She participates in several funded research projects working on topics like software visualization (CISC), user interface and intelligent agents (DARPA), and Web services for senior citizens (NIH).

Luiz A. DaSilva[SM] (ldasilva@vt.edu) joined Virginia Tech's Bradley Department of Electrical and Computer Engineering in 1998. He previously worked for IBM for six years. His research focuses on performance and resource management in wireless mobile networks and QoS. Current and recent research sponsors include NSF, the Office for Naval Research, the U.S. Customs Services, Intel, and Microsoft Research, among others. He is a member of ASEE.