ICAT 2003
December 3-5, Tokyo, JAPAN

Hierarchical QoS Architecture for Networked Virtual Dancing Environment*

Jin Ryong Kim[1], Youjip Won[1], Beomeun Kim[2]

[1] Div. of EECE, Hanyang University
17 Hangdang-dong, Sungdong-ku, Seoul, Korea
{ jessekim | yjwon }@ece.hanyang.ac.kr

[2] Advanced Development Group, Samsung Electronics
416 Maetan-dong, Paldal-ku, Suwon, Korea
bekim@samsung.com

Abstract

In this paper, we present the hierarchical QoS architecture for the virtual dancing environment. In this system, geographically distributed users share the virtual dancing hall and interact to each other. The participating object can be a graphical avatar or live video stream. It allows the coexistence of graphic objects and real stream in a shared virtual space. One of the main technical challenges in developing distribute virtual environment is to handle excessive network traffic. In an effort to effectively reduce the network traffic, we propose to adjust the QoS of each object with respect to the distance from the observer in the virtual space. The server maintains the QoS vector for each client's shared space. The server controls the packet traffic to individual clients based on its QoS vectors. We develop proto-type virtual dancing environment. Java based development enables the client to be platform independent. The result of experiment shows that the adoption of hierarchical QoS architectures significantly reduces the overall network traffic.

Key words: QoS, DVE, Multimedia Streaming, Multicast, Mixed Reality

1. Introduction

1.1 Motivation

The distributed virtual environment(DVE) is a distributed system where the clients are located in different parts of the network and the participants concurrently explore and interact with each other under a high-resolution, 3D graphical environment. The application of this technology ranges from education, medical operations to military training. DVE is setting forth a new set of challenges in the management of system resources in a distributed environment.

* This work is funded by KOSEF grants number R11-2000-073-01000-0.

In DVE, it is crucial that the participant's state needs to be updated in a shared space promptly so that the new state of the participant becomes visible to the rest. There are two approaches in maintaining this consistency: *server-client* and *peer-to-peer*. In the server-client approach for consistency management, the server is responsible for updating the state of individual participants in the shared space and for distributing the updated state to each client. In this approach, the server can easily be overloaded when the number of client becomes high and also can be a single point of failure. In the peer-to-peer approach, each client machine symmetrically updates the view on the shared space. It resolves the burden on the server in the server-client approach.

Another important issue in distributed virtual environment is the load on the *network*. Each client informs the server(or the rest) about its state change. It is possible that the network traffic may increase combinatorially with the increment in the participants. Following methods are used to reduce the network traffic: (i) packet compression; (ii) packet integration for collecting various packets to transmit as one packet; and (iii) method for reducing the frequency of the packet transmission. Also, the network traffic can be reduced using the multicast protocol.

In this work, we intend to augment the virtual environment with live streams from the participants in the server-client approach. A participating object, the *dancer* can be a graphic avatar whose motion is controlled by the keyboard input, motion detector, or etc. or can be a live streaming video of the dancer which is captured and downstreamed from the respective site. In capturing the image of the dancer, it is not possible to capture the image of only the dancer, but the background image accompanies. The chroma-key based processing is mandatory to extract the image of dancer in each frame.

The design objectives of our virtual environment can be summarized as follows: (i) Users participate in the

identical virtual dancing environment; (ii) Participant can be real video stream or 3D avatar; (iii) Each participant interacts with each other; (iv) there is no minimum network bandwidth requirement for participants. We put the requirement (iv) since the network environment becomes more diverse with the technology advancement. It is important to provide a single unified framework which can incorporate the users with variable network bandwidth seamlessly.

The co-existence of multiple streaming sessions generates non-trivial amount of network traffic. However, the network traffic generated by the user in the high speed connection cannot be delivered to the client connected via low speed network line, e.g. modem. Further, if one object is located in the distant place from the observer in the virtual space, the observer may not be interested in or may not be able to see the detailed action of the distant object,

In this work, we present the QoS framework which incorporates the distance between the observer and the object in determining the QoS of the respective object in the virtual space.

1.2 Related Work

The DIVE system[1-4] focuses on minimizing the network delay large scale in distributed virtual environment. In the DIVE system, data is transmitted as peer-to-peer using multicast streaming. To reduce the retransmission of the lost packets, the DIVE system proposes a simple dead-reckoning algorithm based on the linear velocity movement of an object. This algorithm does not apply for the complex motion of the object.

In NPSNET[5], the virtual environment are divided into the hexagonal regions. Each region is assigned to one of the multicast group so that an avatar receives information from the total of seven multicast groups, including avatar itself and all of the regions that is adjacent to it. Whenever an avatar moves through the virtual space, it joins the same number of the multicast groups as the number of the multicast groups it leaves. The group-per-region allocation method of NPSNET can decrease not only the overhead of the host processor but also the network traffic, because data that is sent or received is limited to specified regions. However, if the boundary of the cell is located in the middle of the hexagonal regionals where the objects are passing the boundary frequently, the multicast groups of avatar moving along the boundary are required to be changed frequently[]. Subsequently, the network load increases.

The MASSIVE system[6-10] determines the levels of mutual awareness between objects using the concept of *focus* and *nimbus*. The mutual awareness is a measurement of the mutual interest between an observing object and an observed object. The term *focus* represents the observer's region of interest and the term *nimbus* represents the observer's region of Influence. That is, MASSIVE proposes the QoS architecture based on the observer's awareness. The QoS architecture controls data over network such as volume of audio, level of graphics, and quality of video by the levels of mutual awareness. MASSIVE is designed for the virtual environment of the first person's point of view, therefore, the client himself cannot see his avatar in the virtual environment. That is, the viewpoint of the client and his avatar is identical. Hence, the concept of MASSIVE is not appropriate for the virtual environment of the third party's viewpoint. In addition, because each avatar expresses its face with the video stream and the body with the 3D figure, it is difficult to express the motion of each avatar. In another word, MASSIVE defines the static avatar model rather than the dynamic avatar model and the QoS architecture for the dynamic model is not proposed at all.

The DVE system[11-12] addresses the scalability problem by developing the partitioning algorithm. The partitioning algorithm is based on the linear optimization technique and is shown to be computationally efficient. It can effectively partition the workload evenly among the servers and, at the same time, can reduce the communication overhead. The partitioning algorithm also illustrates how it can partition a very large scale DVE system. However, this algorithm does not propose the hierarchical QoS architecture to reduce the amount of data transmitted to the clients and is not applicable for this project.

2. Overview of TIE System

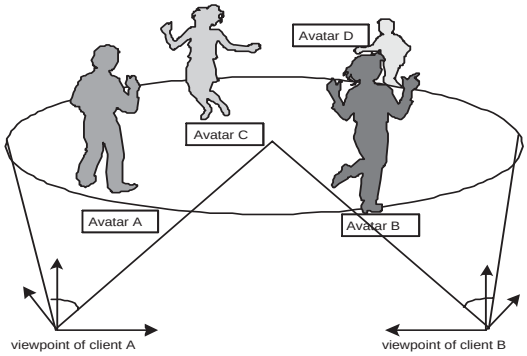
The objective of TIE is to develop a virtual dancing environment where the clients share the same virtual space and communicates with each other. In TIE system, the real video streams of the clients and the 3D avatars co-exist in the shared virtual environment.

The TIE system consists of three parts: *TIE server*, *TIE client*, and *network transport*. Depending on the distance from the primary viewer, we need to adjust the visual and aural quality of the respective object. The TIE server is a program entity which maintains state of the shared virtual environment and position of each object in the virtual environment and updates the QoS information for each object for individual client application. The client application is a program entity which shows the view of dancing studio from the respective user's point of view. The multicast routing protocol based on the adaptive and hierarchical QoS architecture is used. The TIE system is developed based on *restricted* third party's view.

Before to proceed further, we like to clarify the notion of *client* and the *user*. *Client* denotes the application program which renders the image of dancing floor and participating objects. *User* is the human entity which participates in the shared virtual space as an avatar or as real video stream. It is important to note that each user

has its own client application. Let us briefly explain the concept of hierarchical QoS architecture. Although all users share the same virtual environment, what is seen in each user's client terminal is different from each other.

Fig.1. illustrates schematic diagram of the virtual dancing environment of the TIE system. The TIE system is based on server-client architecture.



QoS Table of client A

A vatar	QoS Level
A	high
C	middle
B, D	not shown

QoS Table of client B

A vatar	QoS Level
B	high
C	low
D	low
A	not shown

Fig. 1 Shared Virtual Dance Hall and QoS table for individual client

In this Figure, let client A be the client application running in the user A's local site. Client A sees the avatars A and C. From the observer's position, avatar A is located more closely than the avatar B. In three dimensional space, the object in the farther distance looks smaller. We carefully argue that the object in the farther position is not required to maintain the same level of QoS as the one in the closer distance in each client application. Thus, in client A, the QoS level of avatar A can be higher than the avatar C. Same principle applies in determining the QoS level for each avatar in client B. Also, the viewpoint is not symmetric. Even though avatar B appears in client A, avatar A may not be visible in user B's window. The quality of the live image object can be governed by the frame rate, frame size, and encoding method.

In the TIE server, the QoS level of the avatar is determined by the distance from observer's position in the virtual space. For the live video object, the TIE server dynamically adjusts the frame rate with respect to the distance. The objective of the hierarchical QoS architecture is to reduce the network traffic by properly incorporating the distance between the observer and the object in each user's view.

3. Hierarchical QoS Architecture

3.1 Resource Requirement

It is important to properly estimate the total amount of system resources to maintain the given virtual space. Eq.1 in the Information Principle[13] provides insightful guideline for this.

$$\text{Resources} \approx M \times H \times B \times T \times P \qquad (1)$$

where M is number of messages transmitted in the virtual environment, H is average number of destination hosts for each message, B is average amount of network bandwidth required for a message to each destination, T is timeliness with which the network must deliver the packets to each destination (large values of T implies that the packets may be delivered with longer delays), and P is number of processor cycles required to receive and process each message. Because the TIE system supports both the real video streams and the 3D avatars in the same virtual environment, more resources are required for dynamic adjustment of frame rate, rendering of mixed type objects, etc. The hierarchical QoS architecture tries to reduce the resource by minimizing H and B in Eq.1.

A client's avatar participated in its own view on virtual dancing environment is called *local avatar* and other client's avatar in the same view is called *remote avatar*. The avatar is a representative of the client. If the observer's viewpoint is the same as the local avatar's viewpoint, it is called the first person's viewpoint. Otherwise, it is defined as the third person's viewpoint. In case of the first person's viewpoint, the user cannot see his own avatar. In case of the third person's viewpoint, the viewer can see his local avatar on the screen.

In the TIE system, we introduce the *restricted third person's viewpoint*. In the restricted third person's viewpoint, the distance between the local avatar and the viewer is bounded. Thus, when the local avatar changes its location and the distance from the observer goes beyond the predefined upper bound, the observer's view point is required to be changed accordingly. The local avatar is always visible from the viewer, i.e. in the client terminal.

3.2 Visual QoS

Fig. 2. illustrates the location of the viewer and the avatar in the third person's viewpoint of the virtual dancing environment. The user observes the virtual environment at the viewer's position and his object, which can be an avatar or live stream, appears at the local avatar's position. The objects near the observer's view point look more clear and larger in the client screen.

There are two important principles in visual QoS: (i) The QoS level of each object depends on the distance from the observer's view point; and (ii) For local avatar, the distance from the viewer should be less than its upper bound. Since there is no restriction on the location of the

avatar in the shared virtual space, the observer's view point needs to be updated in accordance with the local avatar's move.

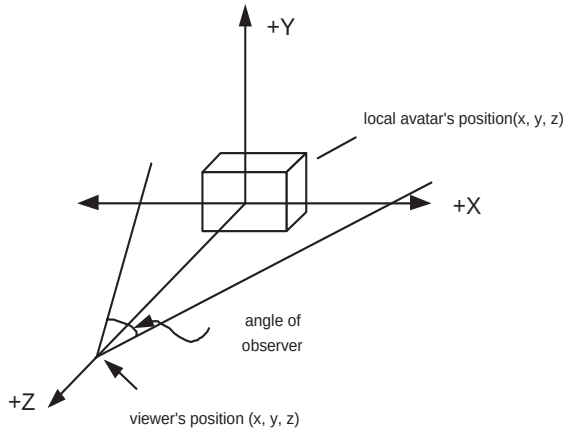


Fig. 2 Location of viewer and avatar

Let (*x, y, z*) and (*x′, y′, z′*) be the local avatar's and the viewer's position, respectively in three dimensional space. The distance *D* between the local avatar's position and the viewer's position is computed as in Eq. 2.

$$D = |z' - z| \tag{2}$$

We need to obtain the QoS level for each avatar based on the distance *D* in Eq. 2. For this purpose, we introduce the QoS mapping function as in Eq. 3.

$$f(D) = O(\frac{1}{D^{1/c}}) \tag{3}$$

This QoS mapping curve has the following characteristics. The avatar's QoS level is inversely proportional to the square of the distance between the local avatar and the viewer. Function *f* maps the distance to one of the set of integers in {1,…,n_{QoS}}. n_{QoS} is the number of distinct QoS levels provided in the system. For example, there are three different QoS levels for avatar, e.g. *high*, *middle*, and *low*. In this case, n_{QoS} corresponds to 3.

We can assign different frame rate for each QoS level, e.g. 15 frames/sec, 10 frames/sec, and 5 frames/sec, respectively. In case of the 3D avatar, there is no notion of frame rate. The motion of avatar is changed based upon the control signals generated from the respective user. For graphic avatar, level of details(*LOD*) is adjusted.

Different QoS mapping function is used for local avatar than the remote avatar, as shown in Fig. 3. Whenever the local avatar is out of the permitted view area, the viewer's position is changed in order to keep track of the local avatar. The reason of tracing the local avatar is that the local avatar is the most important object in the viewer's point of view. This policy of maintaining view can easily changed to other ones. Let Q_T be the lower bound of QoS for local avatar. In the QoS mapping function for local avatar, QoS level must be greater than Q_T, as in Fig. 3.

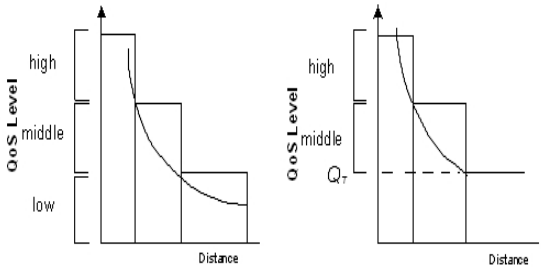


Fig. 3 QoS mapping function

The important issue for mapping function is to determine the range of *D* values for each QoS level. *D* can be equally paced, or the range of *D* for each QoS levels can be set up differently. Session with higher QoS level generates more network traffic and put more load on the client application.

The QoS level of each client's avatar is saved in $m \times n$ matrix, *Q*. m and n denotes the number of avatars, and the number of clients, respectively. In the TIE virtual environment, the number of the clients is the same as the number of the avatars.

Fig. 4 illustrates the QoS matrix. Q_{ij} denotes the QoS level of avatar *i* in the client *j*'s view. When an avatar *i* updates its position, all the elements in a row *i* need to be updated. For each client *j*, the streams are sent based on the information given in column *j*.

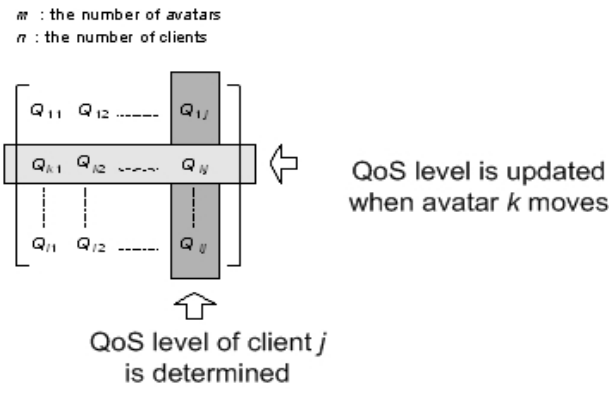


Fig. 4 $m \times n$ QoS matrix

3.3 Audio QoS

The local avatar is not able to hear all the sound or voice of all the other avatars in the virtual environment. To

address this issue, we introduce the concept of *influence region of avatar interaction*. When an object is within a certain distance from the local avatar, the client can hear its sound. The volume of sound is inversely proportional to the distance between the two objects. When an object is beyond a certain threshold distance, the local avatar is not able to hear it. Eq.4. denotes the relationship between the distance and the sound.

$$R_a + R_b \geq d \rightarrow \text{sound is transmitted}$$
$$R_a + R_b < d \rightarrow \text{sound is not transmitted} \quad (4)$$

4. TIE System Architecture

4.1 Implementation of Hierarchical QoS Architecture

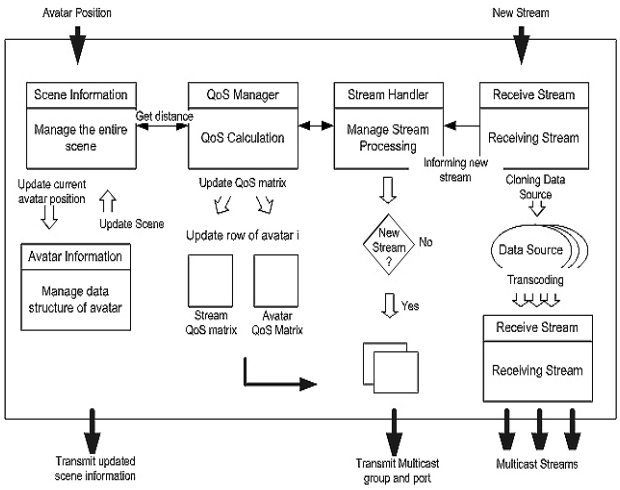


Fig. 5 TIE system architecture for virtual dancing environment

Fig. 5 illustrates the TIE server architecture. *SceneInformation* class manages the entire scene information and *QoSManager* class determines the levels of QoS. *StreamHandler* class is responsible for transmitting and receiving streams. These three classes are used to determine the levels of QoS for each avatar and to manage the consistency of virtual dancing hall. When the new avatar joins the virtual dancing environment, *AvatarInformation* class is created which manages information of the avatar.

When the client sends the state information for its local avatar, the server updates the position, re-computes the QoS level, and transmits the appropriate streams to each client. *QoSManager* class receives the user A's new position from *SceneInformation* and each client recalculates the avatar A's QoS level. The QoS level of live video stream denotes the frame rate and the QoS level of the avatar denotes the LOD of the 3D avatar.

Whenever the new streams are received from the client, the data source of the streams is copied and transcoded into high, middle, and low of the QoS levels. After transcoding, classified streams are multicasted to other multicast groups. *StreamHandler* class catches all avatars' QoS information corresponding to the current client in *QoSManager* class when *StreamHandler* class receives a new stream from the client. This QoS information is stored in the column of the QoS matrix.

4.2 Multicast

Adjusting the frame rate based on client's QoS level is as follows. For each object in the virtual environment, a number of multicast channels are created. The number of mutlicast channels corresponds to the number of QoS levels in the system. When there are three QoS levels, e.g. *high*, *middle*, and *low*, three multicast channels are created with the respective avatar or streaming object, e.g. T_h, T_m and T_l. To get *middle* QoS level stream, the client need to subscribe for T_m and T_l multicast channels for the respective avatar. As the distance between the object and the viewer changes, the client dynamically joins and leaves the multicast channel.

In case of wireless environment, there will be a network bandwidth constraint. In IEEE 802.11b, the maximum network bandwidth is 11Mbps. By transcoding and multicast, we can support both wired and wireless network environments.

4.3 Client System

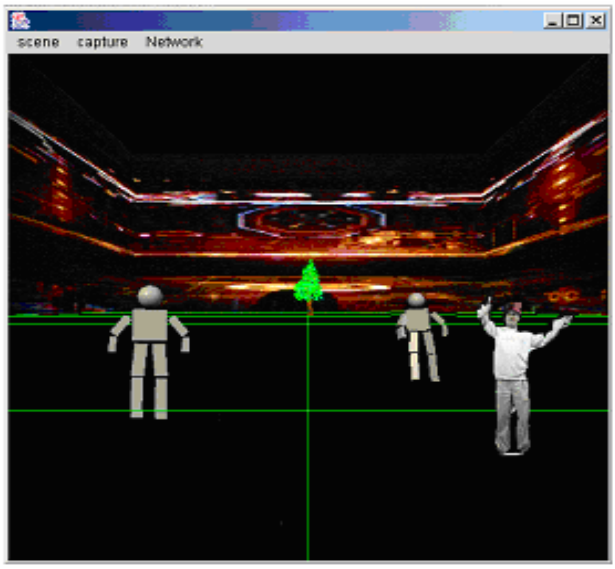


Fig. 6 Client system

Fig. 6 shows how the real video avatar and 3D CG avatars explore and interact with each other in the shared virtual dancing studio. The TIE client system is designed to render CG and real video stream simultaneously using JMF and Java3D. The client connects to the server to enter the virtual space in forms of either a video or a 3D CG avatar. If the client joins the virtual space as a video avatar, the real video streams of the user are captured by the user's camera. The pre-processed video stream is

encoded with *H.263* format and transmitted to the server. Each client in the shared virtual space receives the video stream using RTP stream. JMF processor transforms the received video stream to 2D image.

5. Experiment

For validity and novelty of our work, we conducted the experiment. The purpose is to study the effectiveness of the hierarchical QoS architecture when there are a large number of clients are in a shared virtual environment. In this experiment, we measure the network traffic in the TIE system. We conduct the experiment using two real video streams and five CG avatars in the shared virtual environments. Each QoS level is mapped to its own value so that high, middle, and low quality levels are set to 15fps, 10fps, and 5fps, respectively. In this experiment, we demonstrate the performance of the proposed system by comparing it with the system that QoS architecture is not applied. The experiment is set under local area network environment and we intentionally change the QoS level of each avatar by moving them around the virtual worlds to see how it influences the system. In case of video streams, the real streams of the users are received at the server from respective clients, transcoded into *high*, *middle*, and *low* of the QoS levels, and multicasted to the clients. We use the range from 233.0.0.0 to 233.0.0.255 for multicast addresses

5.1 Load on the Client

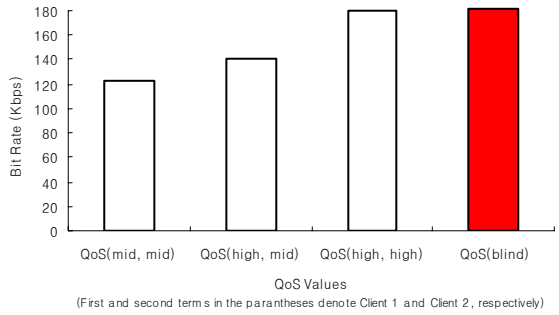


Fig. 7 Network traffic at client 1

In this experiment, we captured the packets from clients and evaluated them. Fig. 7 shows the effectiveness of our proposed QoS algorithm. We adjusted two video streams to have different QoS values at each case by changing their positions in the virtual worlds. In this Figure, each bar represents the stream data receiving rate at the first client site. The first three bars represent the cases that our hierarchical QoS architecture is applied. The receiving rate of the first case where the qualities of video streams are both *middle* is around 120 Kbps. The receiving rate of the second case where the quality of the first video stream is *high* while the second one is *middle* is around 140 Kbps. However, the receiving rate of the last case where QoS is not applied is 180 Kbps. Since the QoS management is not applied at all, both of the video streams are streamed at 90 Kbps, for each. Thus, it is clear that the amount of network packets can be reduced by applying our QoS architecture.

Fig. 8 is the result from another client. In the same manner, the first three bars represent the stream transmission rates that QoS is applied while the last one represents the stream transmission rate without applying QoS architecture. In this Figure, the receiving rate of the first case where the qualities of both video streams are *low* is around 60 Kbps. This value is one third of the last case, which is 180 Kbps. Thus, it proves that our hierarchical QoS architecture can reduce the network traffic by 1/3 at most, compared to non-hierarchical QoS architecture.

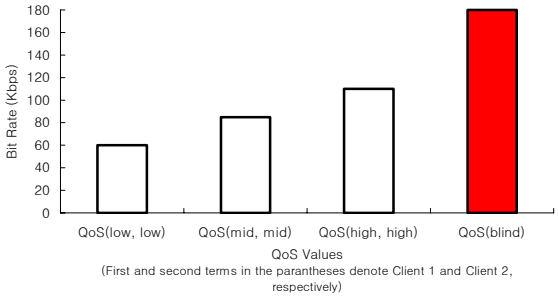


Fig. 8 Network Traffica at client 2

5.2 Packet Transmission on the Network

In this experiment, we captured packets on the network. We set all seven clients(two for video avatars and five for CG avatars) to receive two streams from video avatars for each. To validate our QoS algorithm, we compared our QoS based system to non-QoS system. For non-QoS system, we used unicast for the video delivery.

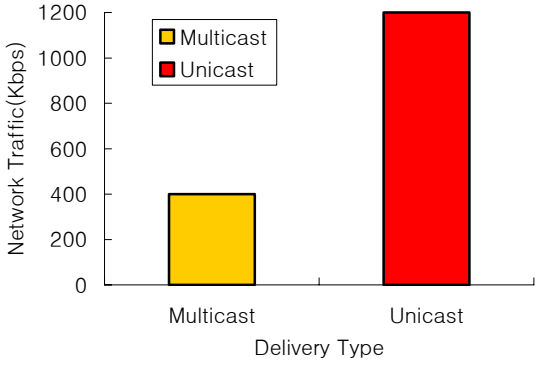


Fig. 9 Packet transmission on the network

Fig. 9 is the network load in the TIE system. Since our algorithm uses the hierarchical QoS architecture based

on the multicast transmission scheme, the load is much lower than the unicast system. In the unicast system, all the streams transmitting to each client are the highest qualities of frames and each stream has to be delivered to its own destination. On the other hands, in the multicast system using our hierarchical QoS architecture, the clients simply join their respective multicast groups depending on their positions to receive video streams. Therefore, by adaptively changing the QoS level of each avatar in the shared virtual environment, we are able to significantly reduce the overall network traffic.

5.3 Simulation

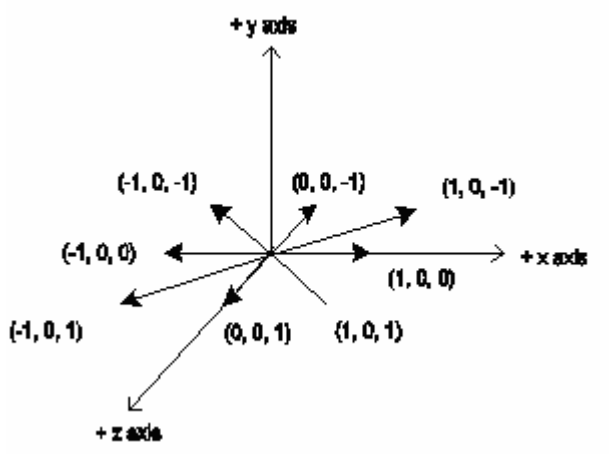


Fig. 10 Direction of avatar's movement

We design the simulation program to find the effectiveness of the hierarchical QoS architecture. In this simulation, the initial position of the avatar is set randomly and the avatar moves in one of the eight directions by one step from the current position as shown in Fig. 10. The avatar's position is updated every second. Whenever the avatar's position is out of the viewer's permitted view area, the viewer's position is updated. The viewer's permitted view area is the maximum distance between the viewer's and the avatar's positions, and it is calculated by $y = Q_T$ in the QoS mapping curve.

In this simulation, the following variables are defined: number of clients participated in the virtual dancing environment, elapsed time, number of QoS levels, and density of avatars in the selected region. The default values of the variables are defined as follows: number of clients participated in the virtual dancing environment is 10, elapsed time is 100 seconds, number of QoS levels is 3(0, 3, and 7), and density of the avatars in the selected region is 0%. In this experiment, we assume that network is not bottleneck point.

In Figure 11, the number of packets that the clients receive for 1 second is measured using JMStudio in JMF. The average packet size is approximately 907 bytes/packet. We find that the system is more scalable from the aspect of network traffic when the shared virtual environment is controlled by the hierarchical QoS architecture. By applying the hierarchical QoS architecture, the number of packets transmitted over the network decreases by 1/3 compared to non-hierarchical QoS architecture. With the increase in the granularity of QoS level, we can achieve further decrease in the network traffic. Thus, it is important to select the suitable number of QoS level depending on how the virtual dancing environment is designed.

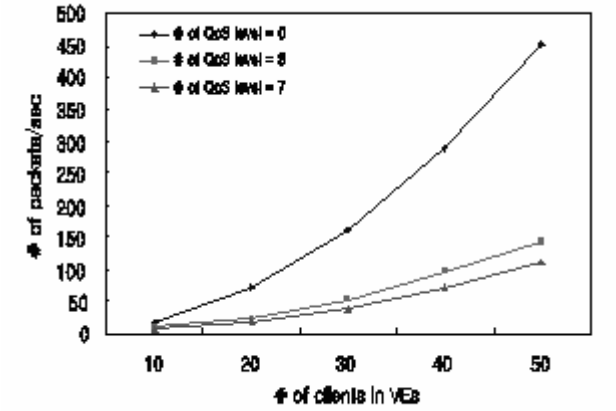


Fig. 11 Number of clients vs. network traffic

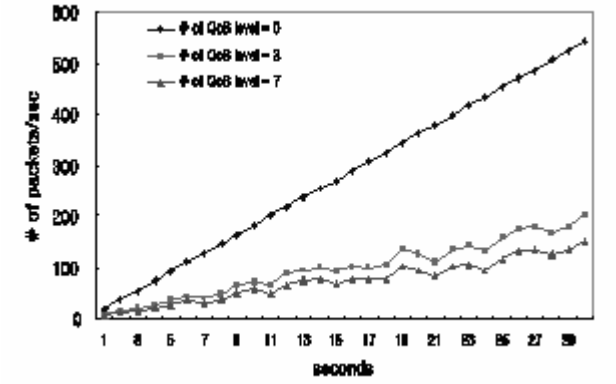


Fig. 12 Density of avatars vs. network traffic
(# of avatars = 30)

Fig. 12 illustrates the relationship between the packet traffic and the density of avatars when 30 clients participate in the virtual dancing environment. The density of avatars is a ratio of the number of avatars in the selected region over the total number of avatars in the virtual dancing space. The network traffic generated from the densely populated region is much larger than the network traffic generated from the less densely populated region. However, we find that the overall traffic becomes smaller with hierarchical QoS architecture when the objects are focused on smaller

fraction of the shared virtual space. We divide the virtual dancing environment into 9 regions with the same dimension. The density of avatars ranges from 0 to 100. For each density value, we locate a number of avatars to particular region to maintain the given density value. The other avatars are dispersed to the rest of the virtual space randomly. We increase the density of avatars in the selected region gradually and measure the number of packets transmitted in the virtual environment. In Fig. 12, the number of packets transmitted over the network is the same regardless of the density when the hierarchical QoS architecture(number of QoS level is 0) is not applied. On the other hand, the number of packets transmitted over the network decreases with the hierarchical QoS architecture.

5. Conclusion

In this paper, we propose the hierarchical QoS architecture of the TIE system to reduce the number of packets transmitted over the network in the virtual dancing environment. The hierarchical QoS architecture has several advantages: (i) it controls the frame rate of the stream by the QoS level based on the distance between the viewer and the avatar, (ii) the client receives the packets for only the objects in its visible region. This can significantly reduce the amount of network traffic, (iii) the quality of audio can be improved by applying the concept of influence region, and (iv) the hierarchical QoS architecture manifests itself when the participating objects are clustered in relatively smaller region rather than in the evenly populated in the virtual environment. The result of experiment shows that the hierarchical QoS architecture developed in this work successfully reduce the amount of network traffic involved in the distributed virtual environment.

References

[1] C. Carlsson and O. Hagsand: Dive-a Multi-user Virtual Reality System. In *Virtual Reality Annual International Symposium*, pages 394-400. IEEE, September 1993.

[2] E. Frecon and M. Stenius. Dive: A Scaleable Network Architecture for Distributed Virtual Environments. *Distributed Systems Engineering Journal*, 5(3):91-100, September. 1993.

[3] O. Hagsand. Interactive Multiuser in the Dive System. *IEEE Multimedia Magazine*, 3(1):30-39, Spring 1996.

[4] A. Steed, J. Mortensen, and E. Frecon. Spelunking: Experiences using the Dive Systems on Cave-like Platforms. In Proceedings of *Immersive Projection Technologies and Virtual Environments 2001*, pages 153-164. Springer-Verlag, May 2001.

[5] M. R. Macedonia, M. J. Z. Pratt, D. R. Brutzman, D. P. Barham, Paul T. Exploiting Reality with Multicast Groups: A Network Architecture for Large Scale Virtual Environments. In Proceedings of *the 1995 IEEE Virtual Reality Annual Symposium*, North Carolina. (1995).

[6] S. Benford and C. Greenhalgh. Massive: A Collaborative Virtual Environment for Teleconferencing. *ACM Transactions on Computer-Human Interaction*, 2(3):239-261, September 1995.

[7] S. Benford, C. Greenhalgh, G. Reynard, C. Brown, and B. Koleva. Understanding and constructing shared spaces with mixed-reality boundaries. *ACM Transactions on Computer-Human Interaction*, 5(3):182-223, September 1998.

[8] C. Greenhalgh, J. Pubrick, and D. Snowdon. Inside massive-3: Flexible Support for Data Consistency and World Structuring. In Proceedings of *the Third International Conference on Collaborative Virtual Environments*, pages 119-127. ACM, September 2000.

[9] S. Benford, C. Greenhalgh, D. Snowdon, and A. Bullock. Staging a Public Poetry Performance in a Collaborative Virtual Environment. In Proceedings of *the 5th European Conference on CSCW '97*, pages 125-140. Dordrecht: Kluwer Academic Publishers, September 1995.

[10] C. Greenhalgh, S. Benford, and G. Reynard. A QoS Architecture for Collaborative Virtual Environments. In Proceedings on the Seventh ACM International Conference on Multimedia (Part 1), pages 121-130. ACM, October 1999.

[11] J.C.S. Lui, M.F. Chan: An efficient partitioning algorithm for distributed virtual environment systems. *IEEE Transactions on Parallel and Distributed Systems*, Volume: 13 Issue: 3, March 2002: Page(s): 193 -211.

[12] J.C.S, Lui: Constructing Communication Subgraphs and Deriving an Optimal Synchronization Interval for Distributed Virtual Environment Systems, *IEEE Transactions on Knowledge and Data Engineering*, Volume: 13 Issue: 5, September-October 2001: Pages: 778-792.

[13] Sandeep Singhal, M.Z.: Networked Virtual Environments : *Design and Implementation*. (1999): Addison-Wesley/ACM Press.