Telecommunication Systems 25:3,4, 353–370, 2004

A Peer-to-Peer Message Exchange Scheme for Large-Scale Networked Virtual Environments *

YOSHIHIRO KAWAHARA ** and TOMONORI AOYAMA kawahara@mlab.t.u-tokyo.ac.jp
Graduate School of Information Science and Technology, The University of Tokyo, Japan

HIROYUKI MORIKAWA
Graduate School of Frontier Sciences, The University of Tokyo, Japan

Abstract. This paper presents a scalable peer-to-peer communication architecture suitable for supporting networked virtual environments that does not require costly dedicated servers or special infrastructure such as multicasting. Entities establish an overlay network based on the distance between the entities. As message exchange is handled directly by unicast between nearest-neighbor entities, the communication overhead is kept low. The proposed architecture is highly scalable and suitable for online applications such as massively multiplayer online role-playing games with hundreds of thousands of participants, where the dominant factor affecting scalability and performance is the overhead associated with exchange of update messages between users.

Keywords: networked virtual environment, peer-to-peer, 3D chat

1. Introduction

With the recent widespread household adoption of broadband Internet access services such as digital subscriber line (DSL) and cable modem services, many service providers are searching for attractive content that can take advantage of these high-speed, always-on connections. Massively multiplayer online role-playing games (MMORPGs), where users can take part in a persistent online gaming world at any time, are emerging as popular uses of such Internet connections. Individual titles can attract hundreds of thousands of registered users for online gaming, and the MMORPG market is expected to grow significantly in the future as more household become permanently connected to the Internet.

The development of online games is more complex than their standalone equivalents, requiring a significant level of network programming and, after the game has been distributed, maintenance and administration of a network infrastructure to support the online gaming community. As any down-time can have a serious impact on profits, online gaming service providers are constantly involved in up-scaling of the network implementation to ensure that the system can cope with dramatic increases in the number

* The early version of this paper has been presented at 8th IEEE International Conference on Communications Systems, November 2002.
** Corresponding author.

of users and potential failures. The cost for maintaining the huge number of servers and networks around the clock is in fact so great that only a handful of service providers can afford to launch online game services. Lowering the entry barrier for offering these services is then an important part of fostering this potentially lucrative market. One solution is to develop a new communication scheme or system architecture designed specifically for ready deployment of networked virtual environments.

The networked virtual environment (Net-VE) is a software system that allows multiple users to interact with each other in a three-dimensional virtual environment created on networked computers [Singhal and Zyda, 13]. Services such as MMORPGs can be regarded as a derivative of Net-VE. In Net-VE systems, an enormous number of participants from a range of computing environments interact with each other in one persistent virtual environment in real time. Improving the scalability of such systems is considered to be an important and interesting research challenge. The dominant factor affecting the scalability of these system is the method for exchanging update messages, which pass game information such as entity IDs, location information in the virtual environment, and the dynamic state of entities. This paper presents a peer-to-peer message exchange scheme for a scalable chat application that does not require costly servers or a dedicated infrastructure. In the proposed message exchange scheme, entities search for other entities sequentially without requiring a multicast infrastructure or centralized mechanism.

The paper is organized as follows: first, we describe conventional communication architectures used for constructing Net-VE applications. Then we show how our message exchange scheme works and evaluate through simulation. Finally, we describe a preliminary implementation as a proof of concept.

2. Related works

A number of communication architectures have been proposed to address this issue, each designed according to the requirement of specific applications. In most cases, these architectures fall into three categories of pure peer-to-peer (e.g., SIMNET, NPSNET [Macedonia et al., 9]), client-server architectures (e.g., RING [Funkhouser, 4], NetEffect [Tapas et al., 15]), and hybrid approach [Bauer et al., 2]. In this section we describe conventional schemes and other scalable communication architectures.

2.1. Pure peer-to-peer-based systems

In pure peer-to-peer-based architectures, entities send a unicast or multicast message to other entities when an entity state is updated. Conventional unicast-based approaches yield $O(N^2)$ update messages for N entities, and thus do not scale well to many participants. In contrast, multicast-based approaches scale well when combined with area-partitioning schemes such as cell-based area-of-interest management. However, optimal cell assignment becomes difficult in realistic scenario. If the cell is too large, it does not scale well. If the cell is too small, clients' overhead for switching cell becomes large. Moreover, multicast is not widely available because of problems such as manageability,

lack of a robust inter-domain multicast routing protocol, scalability, and heterogeneity. This fact renders multicast schemes unsuitable for commercial applications such as MMORPGs.

2.2. Client–server-based systems

In client–server architectures, entities send update messages to a central server, where the update messages are replicated and forwarded to the appropriate entities. In such a centralized architecture, the servers represent a potential bottleneck or single point of failure in the system. Multi-server/client architectures or hierarchical architectures address this issue by over-provisioning and load balancing, but requires accurate estimation of the total number of entities in the system in advance before the service provider invests in costly servers. Due to the high costs, providers will often under-provision initially, resulting in a reversion to the conventional up-scaling scheme and limiting the growth of these attractive applications.

Moreover, as the update messages are always disseminated by way of server, client–server-based systems suffer from higher latency, which result in clumsy movement of avatars in the virtual environment.

2.3. Hybrid approach

Thinking that neither pure peer-to-peer nor client server architecture is adequate to support a million-person real-time game, Bauer et al. proposed a hybrid approach that delegate some application functions to the network [Bauer et al., 2]. Besides servers, they deploy middleboxes called "booster boxes" at the edges of an ISP network, attached directly to the ISP's access routes. As the booster boxes are able to acquire network conditions and application requirements, it can reduce the amount of information servers forward to other servers by only sending relevant information. Though the booster box are effective way to enhance scalability of complex distributed games, it is necessary to deploy the booster box at the edges of an ISP network.

Similar attempts have been made in the area of "active network" [Tennenhouse et al., 16] and "programmable network" [Lazar, 8]. In an active network, the routers or switches of the network perform customized computations on the messages flowing through them. This approach facilitates new capabilities such as dynamic reallocation of resources, automated healing from malfunctions and failures, customized information processing in network devices, and easier service creation. However, because of security concern, lacks of killer application, efficiency concern, etc., there are many criticisms of both active and programmable networks for their deployment.

2.4. Modern peer-to-peer scheme

Fully distributed message exchange scheme used in file sharing application such as Gnutella [5] is one of the most successful communication schemes that realize scalable system at low-cost. To distinguish this scheme from "classic" peer-to-peer-based

system mentioned in section 2.3, now we refer to it as modern peer-to-peer scheme. The higher scalability achieved in the modern peer-to-peer system results from decentralization of server role using an overlay network in which users' client nodes are connected with each other. In Gnutella, a "ping" packet is sent to each connected node announcing the node's presence on the network. The ping packet is forwarded node to node. Each node that receives the ping sends back a "pong" packet, which contains the computer's IP address and information about the number of files being shared. Pong packets are relayed node to node back to the first node, following the same path as the initial ping. A file request query is also forwarded similarly. In this way, Gnutella makes it possible for several hundred thousand people to share huge amount of files on the Internet.

In the simplest file share application, the neighboring entity in the overlay network was chosen at random. As the overlay network is just used for file request query, the scheme cannot simply be applied to update messages exchange for Net-VE application. In general, the update messages must be delivered with minimum delay. However, it takes several seconds for query response to be returned in Gnutella.

CAN [Ratnasamy et al., 12], Chord [Stoica et al., 14], Pastry [Castro et al., 3], and Tapestry [Zhao et al., 17] are distributed systems that derive from Gnutella. In contrast to Gnutella, these overlays implement a basic Key-Based Routing (KBR) interface, that supports deterministic routing of messages to a live node that has responsibility for the destination key. They can also support higher level interfaces such as a distributed hash table (DHT) or a decentralized object location and routing (DOLR) layer. These systems scale well, and guarantee that queries find existing objects under non-failure conditions for file sharing applications. However it is not effective for exchanging update messages in Net-VE application because of the delay.

3. Our peer-to-peer message exchange scheme

Based on the discussion in previous section, we describe our peer-to-peer message exchange scheme [Kawahara et al., 6, 7].

3.1. Overview

The message exchange scheme is the most important performance-affecting factor in the design of large-scale networked virtual environment, as message exchange becomes the greatest burden on the network when many users are online simultaneously. The typical update message includes information such as entities' IDs, locations in the virtual environment, various properties, event data, and chat text. Respective types of messages have different characteristics and requirements in terms of data size, reliability, and frequency. Therefore each message must be treated according to the requirements. Location update message is the most frequent update message in typical Net-VE applications. As entities know their own location in the virtual environment, they report their own current location periodically during game play. Size of the message is small but its transmission delay has to be kept small.

To ensure scalability without special investment or a multicast infrastructure, we propose fully distributed approach for exchanging update messages. Each of the participating entities connects to other entities with unicast based on the importance of the relationship such as Euclidean distance in virtual environment, to create an overlay network over IP network. The presence (location information) and status of participating entities which is required by application are shared among appropriate entities with the support of the overlay network. As entities are directly connected to each other, the latency of this scheme is lower than in a server–client scheme, and can also be deployed without a multicast infrastructure. Moreover, as the load of each node is independent of the number of total participants, this scheme scales well.

3.2. Bootstrap method

The bootstrap is a procedure run for an entity when it connects to participate in the virtual environment for the first time. The user initially accesses one of the bootstrap servers, which authenticates the user and introduces other entities that are already in the Net-VE. The bootstrap procedure for conventional peer-to-peer systems is implemented by either a stateful approach (typical of centralized peer-to-peer file-sharing networks such as Napster [11]) or a stateless approach (decentralized, such as Gnutella). As the stateful approach always keeps track of the participating user, the server must maintain accurate information for each user and can become heavily loaded depending on the number of participating users and length of stay. In contrast, the stateless approach is basically a cache-server scheme, where the server stores the information of users who have recently entered the virtual environment. Thus, the stateless approach scales better, but at the expense of data integrity. In our message exchange scheme, we adopt stateless host cache approach for higher scalability [Matsumoto et al., 10]. We compensate for the lack of data integrity by recomposing overlay network as described below.

3.3. Composing the overlay network

After an entity has acquired information for neighboring entities via the bootstrap server, the entity establishes unicast connections to other entities to compose an overlay network. Using the host-cache approach for bootstrapping, the information stored by the server may be out of date. Therefore the overlay network will need to be recomposed using up-to-date information. To achieve this, the entity queries the other entities about "their" neighboring entities. To reconnect to more appropriate entities, each participating entity classifies the surrounding entities as either active entities (AE) in active interaction and which require tight communication, or as latent entities (LE) that have the potential for interaction but which are not currently engaged and require only existence and summarized information. For this classification, metrics such as Euclidean distance in the virtual environment can be used because entities have information of their own location. Except for Euclidean distance, application specific metrics such as information of friend or foe will be beneficial if needed. Use of bandwidth of nodes' access link will be beneficial if we adjust maximum number of active entities based on the types of

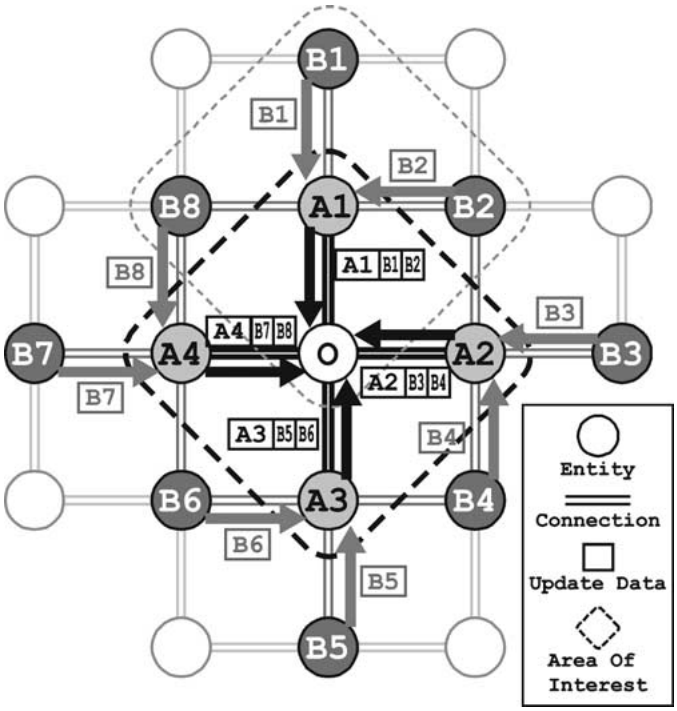

Figure 1. Relationships between entities in the virtual environment.

access connection such as modem or DSL. Figure 1 illustrates an example classification where the nearest four entities are selected as active entities. For example, active entities of entity O are {A1, A2, A3, A4} and latent entities are {B1, B2, . . . ,B8}.

In virtual environment, adopting the nearest N entities as active entities is considered reasonable, particularly in crowded circumstances where communication with more distant entities represents a significant cost. For example, when a person is in a crowd at a cocktail party, everyone is chatting so loudly that he can barely talk to people just standing next to him. When he finds someone he wants to talk to is standing a few meters away, it is reasonable for him to step forward to the person.

Localizing the communication between participating entities in this way reduces the communication bandwidth, and is referred to as area of interest management (AOIM). This concept has been adopted in a number of Net-VE applications. For example, in the NPSNET project, the virtual environment is divided into cells. Each cell is assigned a multicast address, and communication between entities in the same cell is performed by multicasting. Another implementation, NetEffect, uses a distributed server approach instead of multicasting to achieve extensibility and flexibility. However, both of these approaches require the service provider to estimate the total number of users in the application and provide sufficient resources. The advantage of the scheme presented in this study is that AOIM is performed using the overlay network created by participating users.

After active entities are determined, each entity then establishes a direct connection with N nearest active entities and exchanges update messages. Necessary and sufficient information is exchanged via this connection. As for information of latent entities, each entity truncates its active entities' information (network identifier, location information, etc.) for transmission to surrounding active entities. Latent entities are defined as those for which information is not needed immediately or when the existence of the entity is unknown. Rough or truncated information is therefore assumed to be enough in this state. When some latent entity is approaching another and more detailed information is required, they are supposed to recognize each other as active entities to establish a direct connection. For example, if entity O approaches A1 in figure 1, B1 will soon be in O's area of interest. Here, as B1's information has already traveled to O as latent entity, now O can contact B1 to ask inquire current status.

In this way, it is possible for each entity to locate other entities sequentially in a fully distributed manner. Direct exchange of update messages between active entities by unicast keeps the latency lower than in client–server schemes and eliminates the need for a multicast infrastructure. The key factor affecting the performance of the proposed network is then maintenance of optimal adjacency relationships in the composition of Net-VE over an overlay network, particularly, when the number of participants becomes large.

4. Simulation

4.1. Simulation environment

The scalability and performance of the scheme was evaluated through simulation under the following conditions.

1. Entities share a common map of the virtual environment in advance.
2. Each entity holds a list of neighboring entities (AE list).
3. When an entity enters the virtual environment for the first time, a bootstrap server informs the entity of several candidate local entities for connection, and the new entity then picks the most appropriate entities for connection.
4. When an entity enters the virtual environment, the entity communicates with the entities in its AE list, exchanging entity data and the AE list itself.
5. Using the information of neighboring entities, each entity then selects the nearest N entities as new active entities.

The simulation was conducted using a simple virtual environment in which entities are free to move around in the environment, exchanging location information and network ID with other entities. Simulation runs were performed using both the conventional client–server scheme and the proposed peer-to-peer scheme for comparison. The behavior of each entity and the relationship with other entities were shown graphically as

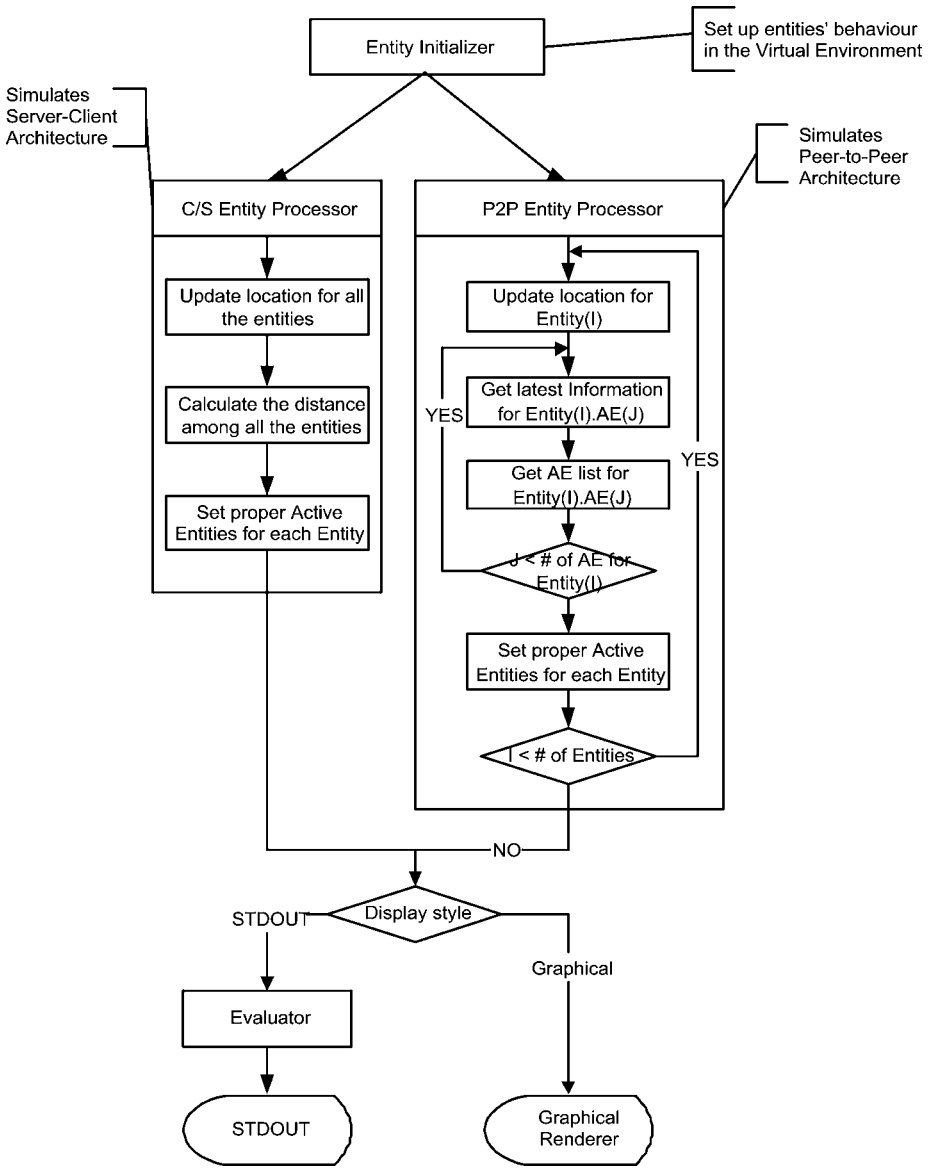

Figure 2. Simulation procedure.

the simulation result. The procedural flow for the simulation is shown in figure 2. Each entity retains information shown in table 1. The major processes are described below.

The entity initializer module creates the instance of each entity. Here, every entity's locus and speed are determined so that their location can be determined uniquely by the timestamp used in simulation. This function makes it possible to compare different message exchange schemes under the same entities' conditions in the simulator. This module also implements the mechanism of the bootstrap servers.

The C/S entity processor module simulates the client–server message exchange scheme. All entities in the virtual environment inform the message exchange server of their current location and ID, and the server replicates the message and transfers it to the appropriate clients. The detailed procedure for this module is as follows.

1. Calculate the entities location at time T.
2. Calculate the distances between all entities to generate a distance table.
3. According to the table, inform each entity of its nearest N entities as active entities.

Table 1
Entity parameters.

Field name		Description
ID		Entity identifier in the network
Location	X, Y, Z	Entity location in the VE
AEList[K]	ID, location, distance, timestamp	Cached active entity data
Num_of_AEs		Number of active entities
Timestamp		Time elapsed since the entity entered the VE

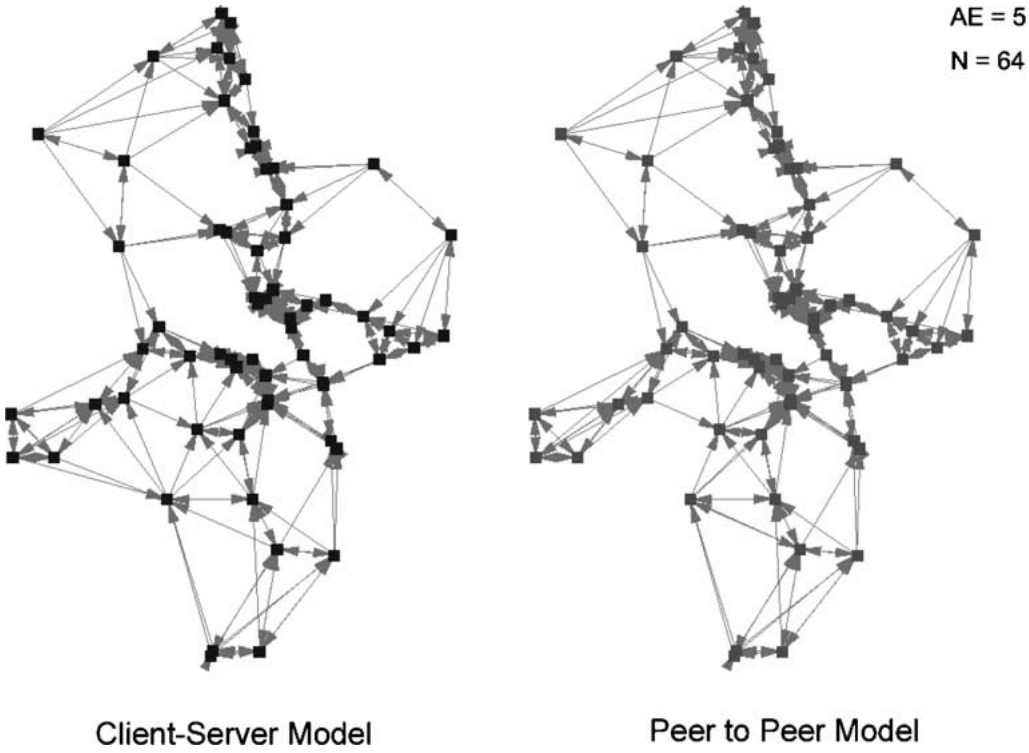

Figure 3. Relationships between entities (64 entities total, 5 active entities). Arrows point to active entities.

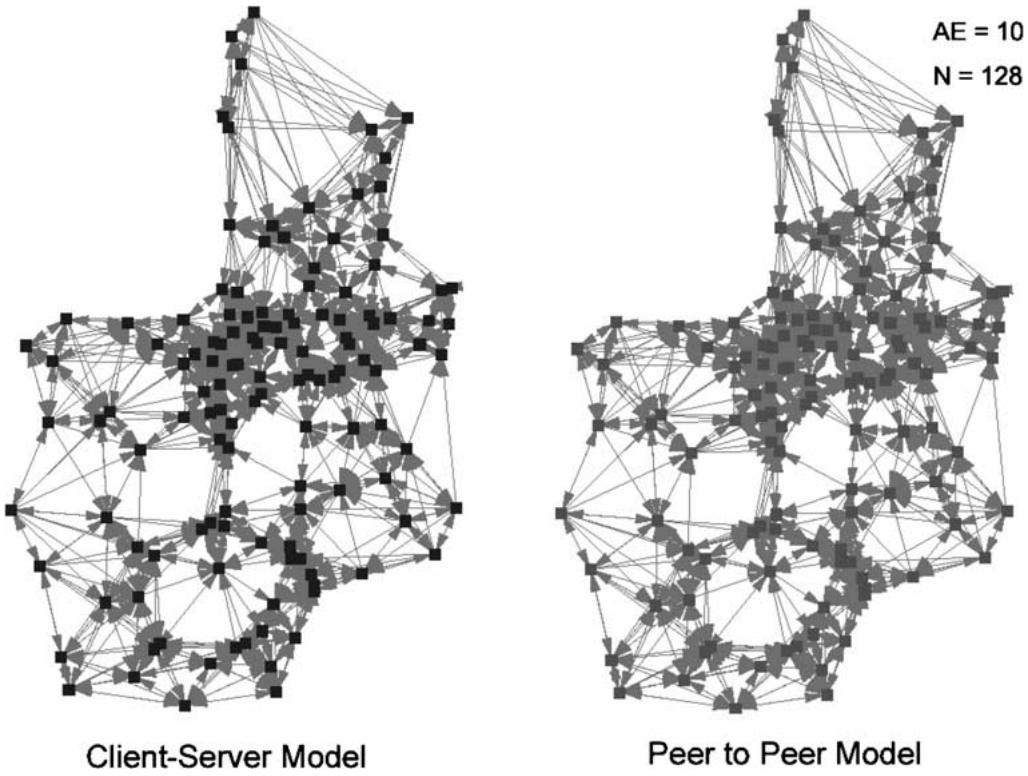

Figure 4. Relationships between entities (128 entities total, 10 active entities). Arrows point to active entities.

The P2P entity processor module simulates the proposed peer-to-peer message exchange scheme. The detailed procedure for this module is as follows.

1. Update the location of entity I (Entity(I)) at time T.
2. Query the neighboring entities of Entity I (Entity(I).AE(J)) for their current locations.
3. Query the neighboring entities of Entity I for their neighboring entities (Entity(Entity(I).AE(J).ID).AE(K))).
4. Calculate the distance between entities based on the information acquired in the previous step and sort entities by distance.
5. Set the nearest N entities of Entity I as its new active entities.

4.2. Results

The networks formed under two simulation conditions are shown in figures 3 and 4. The conditions were set in terms of the total number of entities in the virtual environment and the number of active entities assigned to each entity. The relationships formed under

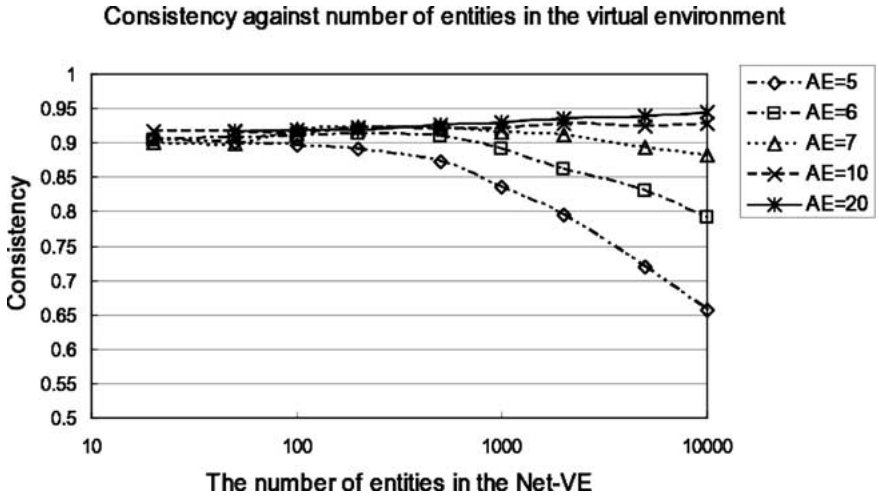

Figure 5. Consistency against number of entities in the virtual environment.

both models, the client–server-based model and the peer-to-peer model, are qualitatively very similar, demonstrating that the proposed peer-to-peer scheme is capable of building optimal adjacency relationships as in the client–server model.

In order to evaluate the system quantitatively, the concept of consistency is defined as a measure of the optimality of the adjacency relationships of the proposed scheme with respect to the client–server model. Consistency is measured as the similarity between the AE lists of the two models, as follows:

$$\text{Consistency} = \frac{1}{N}\sum_{i=1}^{N}\frac{P(i)}{Q(i)}, \tag{1}$$

where N is the number of entities in the virtual environment, $P(i)$ is the number of entities in the AE list of entity i in both models, and $Q(i)$ is the number of active entities for entity i.

Figure 5 shows the consistency of the proposed scheme with respect to the number of active entities assigned to each entity (in reference to the client–server model). The consistency of the peer-to-peer message exchange scheme is greater than 0.9 when a sufficiently large number of active entities is chosen. The drop in performance at lower numbers of active entities arises from division of the overlay network, as discussed below.

4.3. Effect of overlay network division

Figure 6 shows a schematic of the division of the overlay network in the simulation. In this case, the overlay network is divided into three independent groups (1, 2 and 3). Once the network is divided, each entity is able to communicate only with other entities in the same group, and entities cannot obtain information of entities in other groups. When

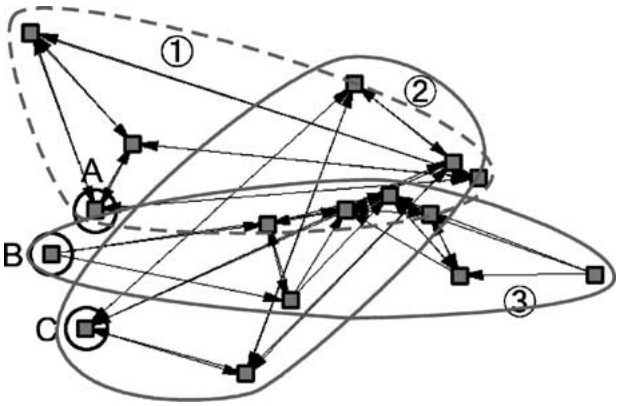

Figure 6. Schematic of overlay network division.

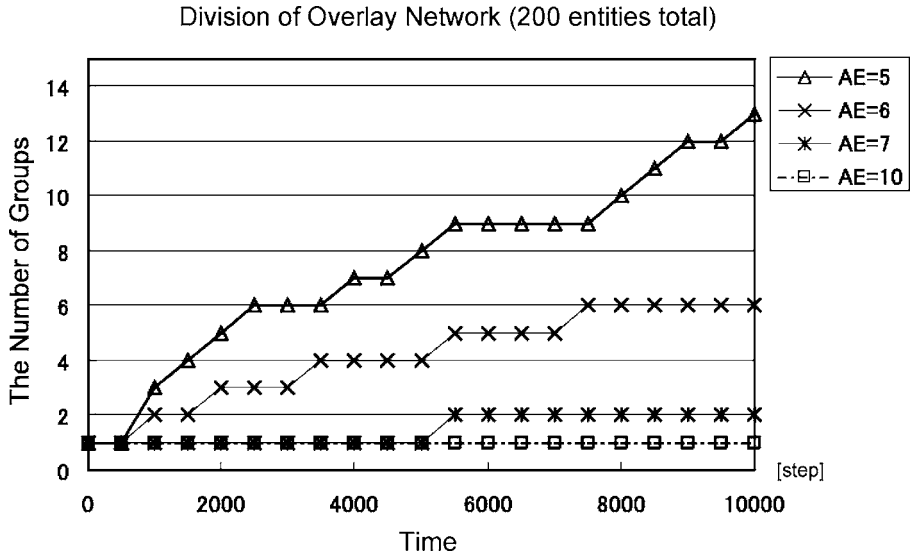

Figure 7. Division of overlay network.

entities exchange their own update information with only the neighboring N entities, division of the overlay networks into subgroups that includes at least $N + 1$ entities is inevitable. Figure 7 shows how the number of subgroups increased as the simulation progressed. In the current simulation, the network was not subdivided when the number of active entities was 8 or more. However, for less than 7 active entities, the overlay network is divided into multiple subgroups as time passes. Even if the number of active entity is sufficient, it only lessens the probability of the division: the divided subgroups are never to reunite. In a realistic application scenario, participants' nodes (e.g., computers) are heterogeneous in terms of network connectivity and/or computational capability, the number of active entities can be accidentally decreased. Therefore it is not realistic

to expect every node to retain a sufficient number of active entities. If the number of active entity is insufficient, it can cause network division, resulting in inconsistency of the virtual world. Therefore some mechanism to avoid this problem is necessary.

If it is possible to guarantee that there is a route from some entity to all the other entities in the virtual environment, the overlay network is guaranteed to be consistent. However, this is not guaranteed in the distributed scheme because every entity's location information cannot be traced without central scheme. In addition, division of the overlay network itself cannot be detected without a central monitoring system because such detection requires global knowledge of entity locations.

To address this issue, the divided groups are reorganized by a probabilistic scheme called random introduction. Random introduction is a means of disseminating update information between isolated subgroups. This scheme can be implemented as follows. At arbitrary intervals, entities reconnect to the bootstrap server to request arbitral entities' information. The entity then connects to these entities and exchanges update information regardless of the location in the virtual environment. If the introduced entity is chosen at random, then it is possible for the information from different subgroups to be exchanged at this random level.

Figure 8 shows the evolution of the number of subgroups with elapsed simulation time using random introduction. Each series indicates a different interval to reconnect to the server. For example, 'Interval = 100' indicates that entities reconnect to the bootstrap server after it has exchanged update information with neighboring entities 100 times. The simulation results show that the random introduction scheme is effective

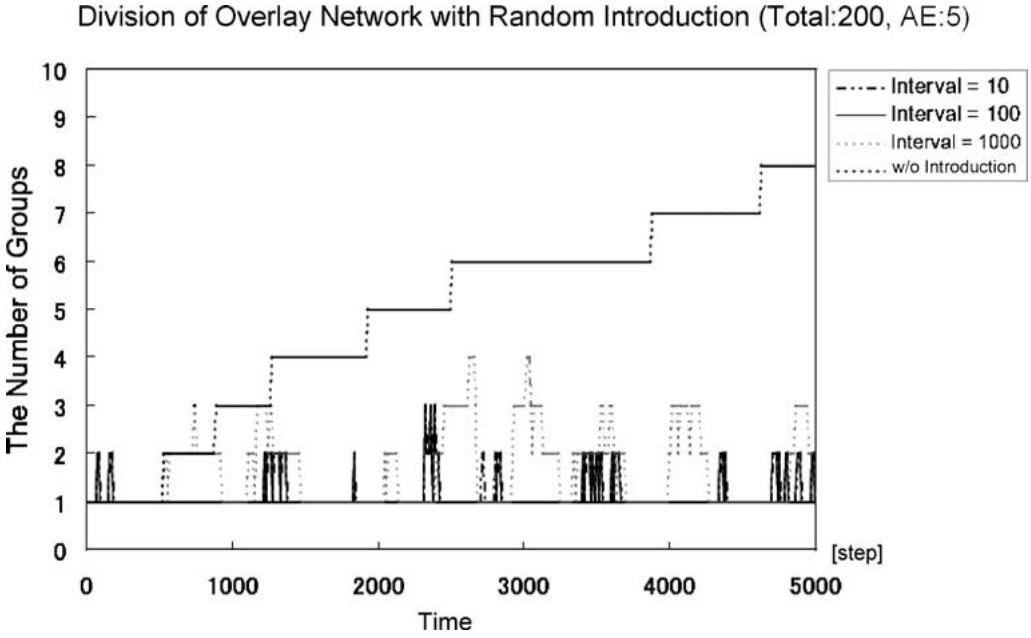

Figure 8. Division of overlay network with random introduction.

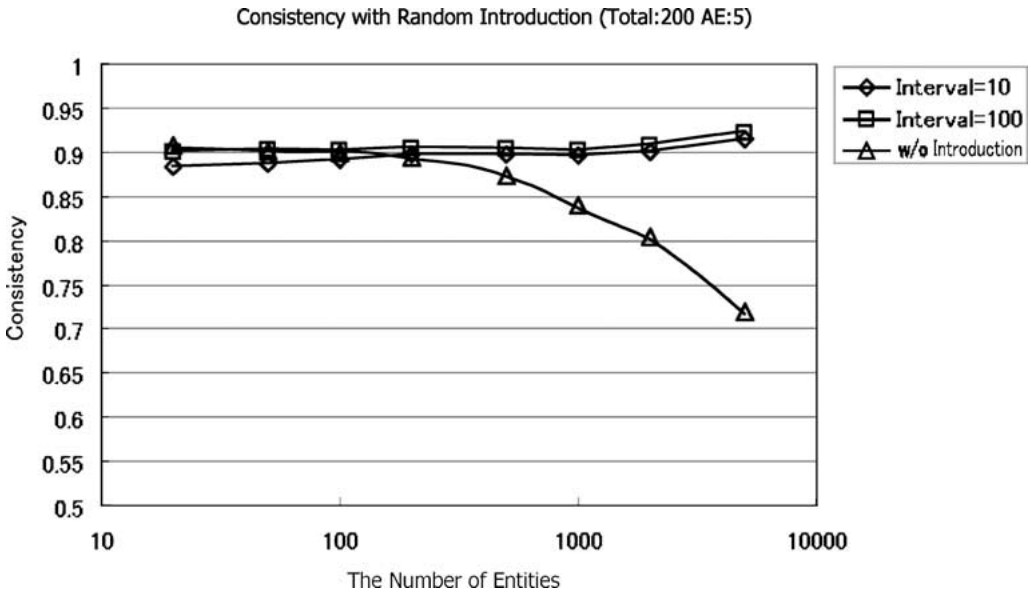

Figure 9. Consistency with random introduction.

for reconnecting subgroups. Figure 9 shows the consistency of the overlay network with random introduction. A much better result is achieved than without random introduction, particularly at lower numbers of active entities. In general, the probability of reconnection of the subgroups increases as the random introduction interval increases. However, as excessive reconnection operations will overload the server, an optimal interval should be determined under real network conditions considering server performance and entity behavior.

5. Implementation

5.1. Application design

As a further investigation of the feasibility of the proposed scheme, the peer-to-peer architecture was implemented for a simple three-dimensional environment in which participants navigate freely and exchange text messages when the interact with other participants. This environment features all of the key components of online virtual environments, and is implemented over TCP/IP.

Figure 10 shows the architecture of the chat application. The main components of the application are the bootstrap server and client software. The function of each component and the overall procedure are described below. The text messenger, input controller and 3D render controller form the user interface of this application, and will not be discussed in detail here.

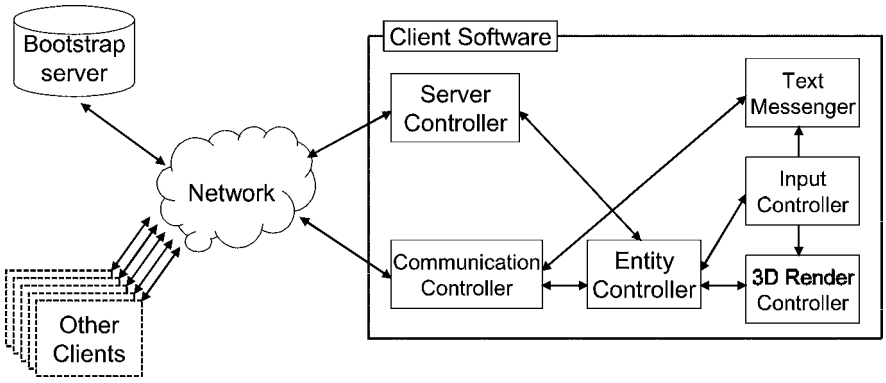

Figure 10. Application architecture.

Packet size	Protocol version	Message type	Source entity ID	Destination entity ID	Message body
2 bytes	*1 byte*	*1 byte*	*4 bytes*	*4 bytes*	*Variable*

Figure 11. Message format.

Server controller. The server controller controls communication between the bootstrap server and the client application. When the client application is initialized, the server controller connects to the bootstrap server to report its local IP address, port number and entity ID. If the bootstrap server receives the information successfully, it notifies the client of the IP address, port number and entity ID of appropriate entities. The server controller is also used to implement the random introduction process by periodically connecting to the bootstrap server to acquire optional entity information after the appropriate absence.

Communication controller. The communication controller is used to exchange messages between entities by distributing the packet to other controllers such as the entity controller and text messenger according to the message type. The communication controller also receives messages from other controllers and reforms the messages into an appropriate format before sending to the relevant entity. Message format used in the application is shown in table 2 and figure 11.

Entity controller. The entity controller is the most important function in the prototype system, controlling the information of the associated entity and the surrounding entities in the virtual environment. When the controller receives an active entity information message from another entity, it calculates the Euclidean distance to that new active entity. In the current implementation, the nearest N entities are defined as active entities, and the rest are defined as latent entities. N is configured as an application parameter. For latent entities, expired information is removed from the list. The entity controller also calculates and updates the entity location in the virtual environment in response to user actions. This updated information is then sent as a unicast message to the active

Table 2
Message format.

Packet size	Length of packet in bytes [12–65535] (network byte order)
Protocol version	Protocol version [1] (for future expansion)
Message type	Types of the message [1–6] 1: Chat text message 2: Entity information message 3: Active entity information 4: Request for entity information 5: Request for active entity introduction 6: Leave message
Source entity ID	Entity ID of message sender
Destination entity ID	Entity ID of message recipient
Message body	Only used for delivering the entity information and chat text

entities. No message is sent if the entity status remains unchanged, but a heart beat message is sent periodically regardless of the entity state to advertise that the entity is still connected. Requests for messages from the active entities and their active entities are also generated periodically.

5.2. Verification

Although large-scale feasibility tests have yet to be conducted, the fundamental behavior of the application with 8 users was examined over a 100 Mbps local area network. We have implemented this 3D chat application on the Microsoft Windows 2000 using Visual C++ 6.0 with DirectX 8.0 (figure 12). All the computer nodes used for verification were equipped with Pentium III 1.2 GHz processor with 512 Mbytes RAM.

In the experiment, 8 users entered the virtual environment and 5 neighboring entities were selected as active entity. The round-trip time of message delivery was measured using an echo function to return a sent message to the originators display. Chat messages of 100 characters were delivered in 2 ms on average, demonstrating the real-time communication capacity of the proposed scheme. Video frame rate always exceeded 30 fps, demonstrating the implementation of this message exchange scheme does not load computers.

5.3. Prospective applications

At the moment, we have implemented 3D chat application using this message exchange scheme. When we apply this scheme to other applications such as MMORPG, some mechanism that guarantees the fairness of the users would be needed. It results from that current peer-to-peer-based approach cannot detect “cheating” or foul play.

However, we believe this problem can be avoided using asynchronous synchronization scheme developed by Baughman and Levin [Baughman and Levin, 1]. The main idea used in the scheme is as follows. Each player decides but does not announce its next turn (in game clock). Each player instead announces a cryptographically secure

Figure 12. User interface.

one-way hash of its decision as a commitment, including randomized padding if necessary to avoid recognizable hashed decisions and avoid collisions. Once all players have announced their commitments, players then reveal their decisions in plaintext. In this way, hosts can easily verify revealed decisions by comparing hashes of plaintext to the previously sent committed value. Even if each host advances in time asynchronously from the other hosts, it can enter into a lockstep-style mode when secure interaction is required. With this scheme, correct playout and fairness are guaranteed. Therefore our peer-to-peer communication scheme has prospect of being applied to applications other than simple 3D chat.

6. Conclusion

A fully distributed message exchange scheme for networked virtual environments was proposed in which entities locate entities in the system systematically by exchanging update messages with immediate neighbors. Simulation results showed that this message exchange scheme forms a network of entities in a virtual environment in a scalable man-

ner, making it suitable for large-scale Net-VE applications without the need for special infrastructure. A trial implementation of the scheme was also shown.

The current peer-to-peer scheme does not include provision for detecting cheating. In order for the architecture to be applied to consumer applications such as online gaming, mechanisms for guaranteeing security and fairness will need to be implemented.

References

[1] N.E. Baughman and B.N. Levine, Cheat proof playout for centralized and distributed online games, in: *Proc. of IEEE INFOCOM 2001*, 2001.

[2] D. Bauer, S. Rooney and P. Scotton, Network infrastructure for massively distributed games, in: *Proc. of the 1st Workshop on Network and System Support for Games*, 2002, pp. 36–43.

[3] M. Castro, P. Druschel, Y.C. Hu and A. Rowstron, Exploiting network proximity in peer-to-peer overlay networks, Technical Report MSR-TR-2002-82 (2002).

[4] T.A. Funkhouser, RING: A client–server system for multi-user virtual environments, in: *1995 SIGGRAPH Symposium on Interactive 3D Graphics*, Computer Graphics (1995) pp. 85–92.

[5] Gnutella, `http://gnutella.wego.com/`.

[6] Y. Kawahara, H. Morikawa and T. Aoyama, A distributed communication architecture for networked virtual environments, Technical Report of IEICE, IN-2001-229 (2002) (in Japanese).

[7] Y. Kawahara, H. Morikawa and T. Aoyama, A peer-to-peer message exchange scheme for large scale networked virtual environments, in: *Proc. of the 8th IEEE Internat. Conf. on Communications Systems (ICCS '2002)*, 2002, 3A-04-04.

[8] A.A. Lazar, Programming telecommunication networks, IEEE Networks 11(5) (1997) 8–18.

[9] M. Macedonia, M. Zyda, D. Pratt, D. Brutzman and P. Barham, Exploiting reality with multicast groups: A network architecture for large-scale virtual environments, Proceedings of IEEE Computer Graphics and Applications (September 1995) 38–45.

[10] N. Matsumoto, Y. Kawahara, H. Morikawa and T. Aoyama, An interest-oriented bootstrap method for large-scale networked virtual environments, IEICE Technical Report IN 2002 (2002) (in Japanese).

[11] Napster, `http://www.napster.com/`.

[12] S. Ratnasamy, P. Francis, M. Handley, R. Karp and S. Shenker, A scalable content-addressable network, in: *Proc. of ACM SIGCOMM '2001*, 2001.

[13] S. Singhal and M. Zyda, *Networked Virtual Environments: Design and Implementation* (ACM Press, New York, 1999).

[14] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek and H. Balakrishnan, Chord: A scalable peer-to-peer lookup service for Internet applications, in: *Proc. of ACM SIGCOMM '2001*, 2001.

[15] K.D. Tapas, G. Singh, A. Mitchell, P. Senthil Kumar and K. McGee, Developing social virtual worlds using neteffect, in: *Proc. of the 6th Workshop on Enabling Technologies Infrastructure for Collaborative Enterprises*, 1997, pp. 148–153.

[16] D. Tennenhouse and D. Wetheral, Towards an active network architecture, ACM Computer Communications Review 26(2) (1996) 5–18.

[17] B.Y. Zhao, L. Huang, J.R. Stribling, C. Sean, A.D. Joseph and J. Kubiatowicz, Tapestry: A resilient global-scale overlay for service deployment, IEEE Journal on Selected Areas in Communications (2003) in press.