Peer-to-Peer AOI Voice Chatting for Massively Multiplayer Online Games

Jehn-Ruey Jiang and Hung-Shiang Chen
Computer Science Department
National Central University
Jhongli City, Taiwan, Republic of China

Abstract

In recent years, massively multiplayer online games (MMOGs) have become more and more popular. Many techniques have been proposed to enhance the experience of using MMOGs, such as realistic graphics, vivid animations, and player communication tools, etc. However, in most MMOGs, communication between players is still based on text, which is unnatural and inconvenient. In this paper, we propose the concept of AOI voice chatting for MMOGs. The term AOI stands for the area of interest; a player in the MMOG only pays attention to his/her AOI. By AOI voice chatting, a player can easily chat by voice with other plays in the AOI. This improves the way players communicate with one another and provides a more realistic virtual environment. We also propose two peer-to-peer schemes, namely QuadCast and SectorCast, to achieve efficient AOI voice chatting for MMOGs. We perform simulation experiments to show that the proposed schemes have reasonable end-to-end delay and affordable bandwidth consumption.

1. Introduction

In recent years, massively multiplayer online games (MMOGs) have become more and more popular. For example, World of Warcraft [4], one of the most popular MMOGs, reached a record of 8.5 million subscribed players worldwide. And according to a report by ScreenDigist [5], the MMOG market broke $1 billion mark in 2006. An MMOG is a computer game which can support hundreds of thousands of players playing simultaneously in a virtual world over internet. A player in the MMOG is represented by a personalized 3D character called an *avatar*. By controlling the avatar, a player can navigate the virtual world, fight monsters for rewards, and interacts with other players, and so on.

Many techniques have been proposed to enhance the experience of using MMOGs, such as realistic graphics, vivid

animations, and player communication tools, etc. However, in most MMOGs, communication between players is still based on text, which is unnatural and inconvenient. Players *type* and *read* the text in the chat box instead of *speaking* and *listening*. Furthermore, because the mouse and the keyboard are the major input devices of most MMOGs, it is hard for a player to control the avatar and communicate with other players at the same time. When a player wants to chat with others by typing text, he/she may lose the control of the avatar for a while. Text-based chatting is inconvenient for players to use, especially for those not good at typing. As a result, players begin to seek for voice chatting solutions, such as Teamspeak[2], Ventrilo[3], and Skype[1], etc.

Teamspeak and Ventrilo are two popular VoIP applications supporting group voice chatting. They are client-server based and thus need dedicated servers. When a user of a group talks, his/her voice is transmitted to the server in the form of voice packets. The server then mixes voice contents of all group users and send the mixed contents to each group user. The client only delivers user's voice packets and receives voice packets from the server, but the server needs to receive voice packets from all clients and deliver voice packets in real time. Therefore, the number of users supported by a server is limited; it depends on the server's network bandwidth and computation power. Skype is a popular peer-to-peer based VoIP application. It not only supports telephoning over internet, but also the group voice chatting. Skype needs no dedicate server for mixing voice packets because of its peer-to-peer architecture. However, since Skype only support group voice chatting for at most five users, it is not suitable for MMOGs, which usually have more than five players chatting together. Teamspeak, Ventrilo, and Skype may be used as a voice chatting tool for MMOGs. However, they reduces the interactivity of players in an MMOG since they have static group membership (i.e., the membership of a group is fixed or seldom changed) and a player thus has to previously join a certain group to talk to someone in the group.

In this paper, we propose the concept of *AOI voice chatting* for MMOGs, which is dynamic-membership voice

chatting based on the AOIs of players in the MMOG. The term *AOI* stands for the *area of interest*; a player in the MMOG has a position in the virtual world and only pays attention to his/her AOI, which is ordinarily defined to be a circular area centered at the player [11]. By AOI voice chatting, an MMOG player can easily chat by voice with other players within her/his AOI. This improves the way players communicate with one another and provides a more realistic virtual environment. We also propose two peer-to-peer schemes, namely QuadCast and SectorCast, to achieve efficient AOI voice chatting for MMOGs. The two schemes adopt the peer-to-peer architecture to eliminate the requirement of servers and to utilize the bandwidth of all participating players. We perform simulation experiments for the two schemes to show they have reasonable end-to-end delay and affordable bandwidth consumption.

The rest of this paper is organized as follows: Section 2 introduces some background knowledge. In Section 3, we first describe how we model the system, and we then describe a basic scheme and its problem. In Section 4, we propose QuadCast and SectorCast to support AOI-voice chatting for MMOGs. We perform simulation experiments for the two schemes. The simulation results and the comparisons are given in Section 5. Finally, concluding remarks are drawn in Section 6.

2. Related work

2.1. Architecture of MMOG

Most MMOGs nowadays are based on the client-server architecture. In such an architecture, the virtual world of MMOG is maintained on a centralized server or server cluster, where players log in and start playing the game. By a centralized server or server cluster, the consistency of game states can be easily maintained and cheating between players can also be avoided. However, because the server is in charge of all event processing and message transmission, it becomes a performance bottleneck when the number of players are increasing, this constrains the scalability of the MMOG system.

To achieve better scalability, researchers propose peer-to-peer architectures, such as VON [7], Solipsis [8] and Apolo [9], for the MMOG. In the peer-to-peer architectures, every player runs a same peer program in a distributed manner without a centralized server; the peer program plays the roles of both a server and a client. In the MMOG, a player interacts only with other players in his/her AOI. Therefore, a player only exchanges messages with a limited number of players within the AOI. In this way, the peer-to-peer MMOG architecture can potentially provide better scalability than the client-server one. However, because there is no centralized server, many problems become more com-

plex to solve. For example, in the client-server architecture, finding new players in a player's AOI can be achieved by the server easily because the server has position information of all players. But in the peer-to-peer architecture, players have to discover new players in the AIO by exchanging messages extensively among players according to specific protocols [7, 8, 9].

2.2. Immersive Audio Systems

The paper [12] proposes an *immersive audio communication system* for MMOGs. The system allows an player to hear voices of all players within its "hearing range" by creating a personalized "audio scene" for every player. This personalized audio scene mixes and attenuates all voice contents from other players according to the propagation distances. The paper also examines advantages and limitations of architectures to realize the system, including the peer-to-peer, the centralized server and the distributed server architectures. In the peer-to-peer architecture, a player send the voice packet directly to other players in the hearing range. Due to the direct sending, the system provides low delay and has no single point of failure. However, if a player has a large number of players in the hearing range, the bandwidth consumption may not be affordable since a separate voice packet must be sent to each player in the hearing range in real time. In the centralized server architecture, the centralized server gathers voice streams from all players, mixes them and then sends a separate mixed stream to each player. In this architecture, the centralized server becomes the bottleneck and the single point of failure of the system. In the distributed server architecture, the whole virtual world is partitioned into multiple regions, called *locales*, and audio streams can be processed by different locale servers. A player can transmit audio streams to one of the locale servers with the shortest transmission delay. As shown in [12], the distributed server architecture has shorter delay than the centralized server one, but has longer delay than the peer-to-peer one. However, it poses more complexity in the control and the coordination of distributed servers.

The paper [10] proposes a peer-to-peer based immersive audio streaming system for MMOGs. This paper proposes to use the Voronoi diagram to find out the *connecting neighbors* for a peer (i.e. player) to connect with directly. It also proposes a model for mixing audio of connecting neighbors by taking into consideration the neighbors' positions and audio directions. Each peer in the system gathers the voice streams of all connecting neighbors, mixes them according to the audio mixing model and sends a separate mixed stream to every neighbor in each time step. Since the number of connecting neighbors is often a small constant, the system is affordable. However, the system does not sup-

port a definite hearing range and has the echo problem that a peer may hear its own voice just emitted earlier.

2.3. Human Conversational Speech Model

The article [13] describes some characteristics and statistics of human conversational speech. The human conversation can be modeled as short burst of voice signals (called *talkspurts*) separated by silence gaps (called *pauses*). The gaps occur between phrases, sentences, words or syllables when a speaker is talking. The gap may also caused by the *mutual silence*, which occurs when no one is talking. The talkspurts are contributed by either a *single talk* or a *double talk*. Statistics of temporal parameters of a conversational speech are shown in Table 1. This table shows that in a conversation a person only spends about 40% of time in speaking and keeps silent during the rest of the time. Also, we can see that in a conversation, once a person is talking, the talking rate of the other people is reduced to only 6%. In sum, a person has a probability of 40% to talk in a conversation, and people are often silent when one of the people is talking.

Table 1. Temporal parameters in conversational speech

Parameter	Rate (%)
Talk-spurt	38.53
Pause	61.47
Double talk	6.59
Mutual silence	22.48

3. The System Model and the Problem

3.1. The System Model

The virtual world of an MMOG is modeled as a two-dimensional plane, and the players are modeled as nodes moving on it. (Note that below we use the terms "node" and "player" exchangeably.) Each node has a unique id (ID), a coordinate (X, Y), an AOI, and some other behavior parameters. The AOI of each node defines the area in which the node can interact with others. We assume that each node's AOI is a circle centered at the node with a fixed radius and all nodes have the same AOI radius. The nodes in a node's AOI is called the node's *AOI neighbors*. For example, in Figure 1, the big circle around node A is A's AOI, and nodes B,..,I are node A's AOI neighbors.

Under the above-mentioned system model, the AOI voice chatting can be regarded as *voice packet multicast* within the AOI. When a node talks, the voice packets are
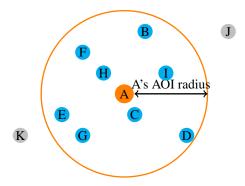

Figure 1. The AOI (Area of Interest) of a node A

multicast to its AOI neighbors. In this way, a node's AOI neighbors can hear its voice, and vice versa. As illustrated in Figure 1, when node A talks, all its AOI-neighbors should receive the voice packet, but non-AOI-neighbors should not. In order to correctly multicast the voice packets to AOI neighbors, we need a recipient list containing these AOI neighbors. In this paper, we assume the MMOG system can provide a node with the information of its AOI neighbors, such as their ids, network addresses, virtual world coordinates, etc. The assumption is practical. For example, the VON system [7] can support such information.

3.2. The Problem of a Base Scheme

In this subsection, we introduce a base AOI voice chatting scheme, NimbusCast, and its problem. When a node talks, it first requests AOI neighbor information from the MMOG system. After that, the node delivers voice packets to every AIO neighbor. For example, in Figure 2, when node A talks, it first figures out nodes B..I are its AOI neighbors, and then delivers separate voice packets to B..I. Nodes J and K do not receive the voice packets because they are outside A's AOI.
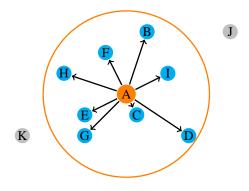

Figure 2. Illustration of NimbusCast

NimbusCast is simple; however, it has the problem of *bandwidth overloading* that the burst bandwidth consumption of a voice source node may exceed the upload bandwidth limitation. This is because voice packets should be sent in real time and when the number of a node's AOI neighbors is large, the burst bandwidth consumption is prone to exceed the bandwidth limitation. For example, if it takes $16kbps$ to support an end-to-end voice communication and the upload bandwidth limitation is $256kbps$, the burst bandwidth consumption will exceeds the bandwidth limitation of a talking node when the number of its AOI neighbors is more than 16. As shown in Figure 3, for a specific node, the problem may occur every time the node talks; however, it does not occur when the node keeps silent.
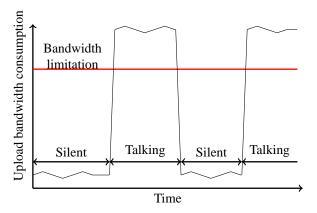

Figure 3. Network bandwidth consumption in NimbusCast

4. The Proposed Schemes

In this section, we propose two schemes, QuadCast and SectorCast, to support AOI voice chatting for MMOGs. When a node talks, it must deliver the voice packets to all its AOI neighbors with reasonable delay and without bandwidth overloading. Therefore, we have the following two design goals for the proposed schemes.

- *Reasonable delay*

    ITU-T recommendation G.114 [14] provides a guideline about the one-way end-to-end (or mouth-to-ear) delay. It says that most users are satisfied with the delay between 150ms to 250ms, while a delay below 400ms may also be tolerable by users. According to the recommendation, the end-to-end delay of the proposed schemes should be less than 400ms.

- *No bandwidth overloading*

The schemes should avoid bandwidth overloading. That is, they should prevent the burst bandwidth consumption of a node from exceeding the upload bandwidth limitation.

As we have mentioned in Section 2, in a conversation, a node only spends about 40% of the time talking, and is silent for the rest 60% of the time. If the idle upload bandwidth of those silent nodes can be used to help forward other nodes' voice packets, the burst upload bandwidth consumption will be reduced, which in turn can help avoid upload bandwidth overloading. However, the forwarding of voice packets will make the end-to-end delay longer. Thus, we should have a systematic way to perform the voice packet forwarding with reasonable delay. QuadCast and SectorCast apply different strategies to divide the recipient list for efficient packet forwarding. Below, we elaborate the details of the two schemes in the following subsections, respectively.

4.1. Quadrant-Based Forwarding (Quad-Cast)

In QuadCast, in stead of directly transmitting voice packets to AOI neighbors, a node transmits voice packets only to few *forwarding assistants* (FAs). These forwarding assistants then forward the voice packets to the remaining AOI neighbors. In this way, because all nodes contribute their bandwidth resource to help forward the voice packets, the burst bandwidth consumption of the speaking node decreases, and the overall quality of conversation is improved.

To save bandwidth, we demand accurate forwarding, which means that each AOI neighbor should receive a voice packet just once. To achieve this, we attach a *recipient list* to the forwarding voice packet to indicate the recipients of the packet. The forwarding voice packet thus contains a packet header, the voice data, and a recipient list, etc. On receiving a forwarding voice packet, the FA is in charge of forwarding the voice contents to all nodes in the recipient list.

When a node talks, it divides the neighbors by their coordinates into four quadrants, and creates four recipient lists, each for a quadrant. Afterwards, for each quadrant, the node closest to the talking node is chosen as the FA for the quadrant. Note that the FA does not exist if there is no node in that quadrant. After the FA selection, the talking node creates four separate forwarding packets with corresponding recipient lists and delivers them to the four FAs. The FA applies the same procedure to forward the received packets recursively until the recipient list is empty. For example, in Figure 4, when node A talks, it first acquires AOI neighbors from the MMOG system, and then divides them into four lists according to their coordinates. It then delivers four forwarding packets to four FAs, namely I, F, E and C, in the

first, second, third, and fourth quadrants, respectively. After receiving the forwarding packets, these FAs apply the same procedure to forward voice packets to the nodes in the recipient lists by dividing them into four quadrants.
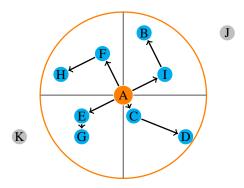

Figure 4. Quadrant-based voice package forwarding

4.2. Sector-Based Forwarding (SectorCast)

In an MMOG, players are usually clustered in some hot spots like markets, town squares or shopping malls. Thus in Quadrant-Cast, the number of players in a certain quadrant may be much greater than those in others, causing unbalanced player grouping. For example, in Figure 5(a), there are much more players in the first quadrant of player A's AOI. The message forwarding in the crowded quadrant thus has more hops, which causes more processing time and transmission delay. If we can evenly distribute AOI players into sectors as shown in Figure 5(b), the message forwarding in each sector will go through approximately the same number of hops. Therefore, the maximum end-to-end delay will be shorter and the quality of service will be better.
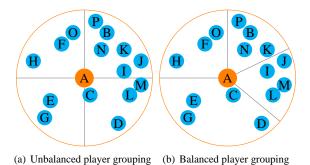

(a) Unbalanced player grouping (b) Balanced player grouping

Figure 5. Unbalanced and balanced player grouping in MMOGs

With the balanced player group in mind, we propose a sector-based AOI voice chatting scheme called SectorCast

for MMOGs. SectorCast and QuadCast are similar; they are different only in player grouping. QuadCast divides the AOI into four fixed quadrants with equal size, while SectorCast divides the AOI into four variable sectors containing approximately equal number of players. For example, in Figure 6, SectorCast divides the AOI into four sectors containing 4 or 3 players. When FAs applies the same procedure as used by the talking player, the maximum (or average) hops of packet forwarding in all sectors are almost equal because sectors have about the same number of players. SectorCast thus has shorter end-to-end delay than QuadCast. However, SectorCast has higher computation complexity than QuadCast. This is because QuadCast only needs to divide AOI neighbors into four quadrants, while SectorCast needs to sort neighbors according to their polar angles (between the X-axis and the lines from players to the voice source or FA), and divide them equally into four sectors.
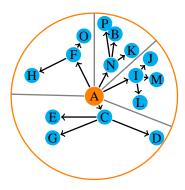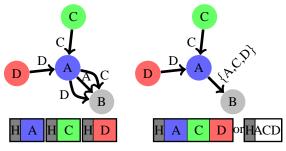

Figure 6. Sector-based voice packet forwarding

4.3. Packet Aggregation

In QuadCast and SectorCast, an FA might forward different packets to a same recipient. These packets are sent separately. However, if we can apply aggregation techniques to merge packets, the bandwidth consumption can be reduced dramatically. Below we propose two aggregation techniques: *header sharing (HS)* and *voice mixing (VM)*. In the former packets are merged by sharing a same header, while in the latter packets are merged by mixing the voice contents. For example, in Figure 7, node A is the FA of node C and D; node A is in charge of forwarding their voice packets to the recipient, node B. At the same time, node A also needs to deliver its own voice packet to node B. As shown in Figure 7(a), without aggregation, three voice packets containing the same header and different voice contents are delivered to node B. However, as shown in Figure 7(b), the three packets can be merged to be one packet by sharing the same header or by mixing the voice contents. As

a result, packet traffic is lowered and bandwidth consumption is reduced.

(a) Forwarding without aggregation

(b) Forwarding with aggregation

Figure 7. Examples of packet aggregation

Since an FA node forwards voice packets to many recipients on behalf of many source nodes, it must have a way to aggregate voice packets for every recipient properly. Below, we model the aggregation of packets as 2-power number addition. A voice packet is assigned a unique ID of a 2-power number. And the ID of the aggregated voice packet is set to be the addition of the IDs of packets from different sources. In this manner, an ID corresponds to a unique combination of packets from specific sources. For example, in Figure 8, there are three nodes A, C, and D talking simultaneously. Assume the voice packets of these three nodes are assigned the IDs 1, 2, and 4, respectively. Also assume that node A is the FA of nodes C and D to send voice packets to nodes B and E. Node B should receive via node A the voice packets from nodes A, C, and D, and thus the ID of the aggregated voice packet for B is 7, the addition of 1, 2, and 4. Node D should receive via node A the voice packets from nodes A and D, so the ID of the aggregated voice packet for D is 5, the addition of 1 and 4. To perform packet aggregation, an FA calculates the ID of the aggregated voice packet for each recipient. Recipients are then grouped by the calculated IDs; they are put in a same group if they are to receive a same aggregated packet. Finally, the FA sends each aggregated voice packet to corresponding recipients by putting nodes of the corresponding group into the attached recipient list.

4.4. Alternatives of the Recipient List

In QuadCast and SectorCast, a recipient list is appended to a voice packet for transmitting the packet only to proper recipients. However, delivering the recipient list consumes a lot of bandwidth. We would like to trade computation complexity with network bandwidth. When a voice packet is forwarded, instead of appending a whole recipient list, we only append the ID of the current FA for the next FA to calculate the recipients.
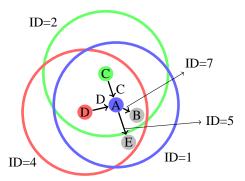

Figure 8. The concept of packet aggregation by 2-power number addition

In QuadCast, when a node talks, it appends its ID to the voice packet and deliver the voice packet to the FAs. The FA can acquire the position and the AOI of the talking node by the ID and then figure out the forwarding area of the voice packet. Once the forwarding area is specified, the FA can select the recipients from its AOI neighbors properly. Similarly, the FA also has to append its ID to the forwarding voice packet in order to allow the next FA to specify the forwarding area. In SectorCast, besides IDs, the source and the FA need to further append to the voice packets the begin and end angles of the forwarding sector for the next FA to calculate the forwarding area to choose recipients from its AOI neighbors properly.

5. Evaluation

In this section, we perform simulation experiments for comparing the proposed AOI voice chatting schemes, QuadCast and SectorCast, with the base scheme. We place 200, 400, ..., 1000 nodes in a 1000×1000 area to simulate different node density scenarios. Nodes are assumed to have arbitrary initial positions. All nodes have the same AOI radius of 100. According to Table 1, each node is assumed to have a 40% probability of talking and 60% of keeping silent. Each experiment case lasts for 1000 discrete time-steps, and each step has an interval of 100ms. In each step, every node moves along a random direction by a distance of 10. The voice packet is assumed to have the format as shown in Figure 9. The header size of a voice packet is 40 bytes ($12(RTP) + 8(UDP) + 20(IP)$), and the voice contents occupy 100 bytes.

We first perform experiments under the assumption that there is no bandwidth limitation. Figure 10 shows the per-node total upload bandwidth consumption for NimbusCast, QuadCast, and SectorCast without bandwidth limitation. In Figure 10 and the following figures, "-HS" in the legend stands for the aggregation method of header sharing ; "-

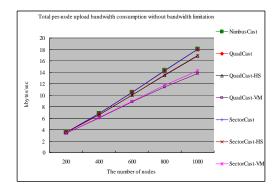| RTP header	UDP header	IP header	Voice content	Recipient list

Figure 9. Format of a forwarding voice packet

Figure 10. Total per-node upload bandwidth consumption without bandwidth limitation for AIO voice chatting schemes

Figure 11. Total per-node upload bandwidth consumption with bandwidth limitation for AIO voice chatting schemes

Figure 12. Drooping rate for AOI voice chatting schemes with bandwidth limitation

VM", the voice mixing. Without packet aggregation, the total bandwidth consumption of the three schemes are the same. This is because all AOI neighbors of a node should receive every voice packet of the node no matter which node sends out the packet (the packet may be directly sent by the node itself or by a certain FA node). However, by applying packet aggregation, the bandwidth consumption of Quad-Cast and SectorCast is reduced, and the reduction is proportional to the number of nodes. When the number of nodes increases, the probability of performing aggregation and the degree of aggregation (i.e., the number of voice packets to be aggregated) also increase. This leads to less bandwidth consumption.

In practice, every node has a bandwidth limitation. Below, we perform simulations with considering the bandwidth limitation. We assume each node has an upload bandwidth limitation of 32k bytes/sec. Under the bandwidth limitation, when a node delivers a voice packet, the packet is delivered normally if there is still enough bandwidth for packet transmission. However, if the upload bandwidth consumption exceeds the limitation, the packet will be dropped. The total per-node bandwidth consumption for the three schemes with considering bandwidth limitation is shown in Figure 11. We can see that QuadCast and SectorCast consumes more upload bandwidth than NimbusCast, especially when the number of nodes is more than 800. This is because the burst bandwidth consumption of NimbusCast exceeds the bandwidth limitation frequently and the so-called band-
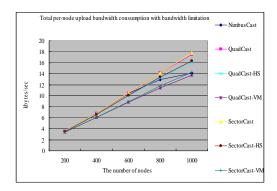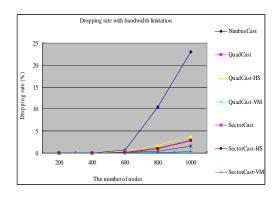
width overloading problem occurs, while the problem does not occur so frequently for the forwarding based schemes, QuadCast and SectorCast.

Figure 12 shows the packet dropping rates of all three schemes. We can observe that in this figure, the dropping rate of NimbusCast is overe 20% when the number of nodes in the system reaches 1000, while the dropping rate of other schemes are below 5%. In [6], the authors summarize that it is acceptable when the dropping rate of voice packets is lower than 5% in the internet voice chatting. The proposed forwarding based schemes, QuadCast and SectorCast, do fulfill this requirement in our simulation setting.

We also measure the average end-to-end delay for all schemes when bandwidth limitation is considered. We adopt the assumption of end-to-end delay proposed in [15]. That is, the packet propagation delay between two directly connected nodes and the packet processing time are assumed to be 70ms and 30ms, respectively. This means
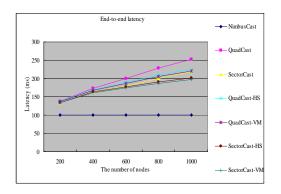
Figure 13. Average end-to-end delay for AOI voice chatting schemes with bandwidth limitation

that when a voice packet is forwarded through one more hop, the end-to-end delay is increased by 100ms. Figure 13 shows the simulation results for the end-to-end delay of the three schemes. As shown in the figure, NimbusCast has the shortest end-to-end delay 100ms because a node delivers all voice packets directly to all AOI neighbors. QuadCast has about 250ms end-to-end delay when the number of nodes is 1000. SectorCast has shorter delay than QuadCast because it evenly divides the AOI neighbors into four sectors, which yields even shorter delay. As we have mentioned in Section 3, the end-to-end delay of a conversation should not exceed 400ms, and a delay below 250ms is considered to be of good quality. We can conclude that QuadCast and SectorCast have reasonable end-to-end delay in our simulation setting.

6 Conclusion

In this paper, we first describe the concept of AOI voice chatting for MMOGs. By AOI voice chatting, a player in the MMOG can chat by voice with other players within his/her AOI. We then introduce NimbusCast as a base scheme, in which each node directly delivers all voice packets to each of its AOI neighbors. This scheme has the shortest end-to-end delay; however, when the number of AOI neighbors increases, the burst upload bandwidth consumption frequently exceeds the bandwidth limitation, which leads to the bandwidth overloading problem and causes a high packet dropping rate.

We propose the QuadCast and SectorCast schemes for avoiding the bandwidth overloading problem. In QuadCast, a speaking player divides the AOI neighbors into four lists according to the quadrants they reside. It then selects a forwarding assistant (FA) for each quadrant, and sends voice packets to the FAs only. Each FA then helps forward voice

packets to the remaining AOI neighbors in the corresponding quadrant. SectorCast is similar to QuadCast. The major difference is that in SectorCast, a speaking player divides the AOI into four sectors with nearly the same number of AOI neighbors. Both the two schemes can deal with the bandwidth overloading problem properly and thus have low packet dropping rate. However, they have longer end-to-end delay. Fortunately, as shown by the simulation results, the two schemes both have reasonable delay. QuadCast and SectorCast can run on either a client/server or a peer-to-peer based MMOG, only if the MMOG can provide proper AOI neighbor information. We are planning to implement QuadCast and SectorCast on top of VON [7] in the near future.

References

[1] Skype, http://www.skype.com/.
[2] Teamspeak, http://www.goteamspeak.com/.
[3] Ventrilo, http://www.ventrilo.com/.
[4] World of warcraft, http://www.worldofwarcraft.com/.
[5] Western world mmog market: 2006 review and forecasts to 2011, http://www.screendigest.com/ reports/07westworldmmog/ readmore/view.html/.
[6] L. Ding and R. Goubran. Assessment of effects of packet loss on speech quality in VoIP. *Proceedings of the 2nd IEEE Internatioal Workshop on Haptic, Audio and Visual Environments and Their Applications (HAVE 2003)*, pages 49–54, 2003.
[7] S. Hu, J. Chen, and T. Chen. VON: a scalable peer-to-peer network for virtual environments. *IEEE Network*, 20(4):22–31, 2006.
[8] J. Keller and G. Simon. Solipsis: A massively multi-participant virtual world. *Proc. of PDPTA*, pages 262–268, 2003.
[9] J. Lee, H. Lee, S. Ihm, T. Gim, and J. Song. Apolo: Ad-hoc peer-to-peer overlay network for massively multi-player online games. Technical report, 2005.
[10] L. Liu and R. Zimmermann. Immersive peer-to-peer audio streaming platform for massive online games. *Proceedings of 3rd IEEE Consumer Communications and Networking Conference (CCNC 2006)*, 2, 2006.
[11] K. Morse, L. Bic, and M. Dillencourt. Interest management in large-scale virtual environments. *Presence: Teleoperators and Virtual Environments*, 9(1):52–68, 2000.
[12] C. Nguyen, F. Safaei, and D. Platt. On the provision of immersive audio communication to massively multi-player online games. *Proceedings of Ninth International Symposium on Computers and Communications*, 2, 2004.
[13] I. T. Union. *ITU-T Recommendation P.59 Artificial conversational speech*, 1993.
[14] I. T. Union. *ITU-T Recommendation G.114 One-way transmission time*, 2003.
[15] R. Zimmermann and L. Liu. ACTIVE: adaptive low-latency peer-to-peer streaming. *Proceedings of SPIE*, 5680:26–37, 2005.