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

Jehn-Ruey Jiang, Hung-Shiang Chen, and Chao-Wei Hung

Department of Computer Science and Information Engineering
National Central University
Jhongli City, Taiwan, R. O. C.

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 latency and affordable bandwidth consumption.

Key words: peer-to-peer, massively multiplayer online games, voice chatting, avatar

I. INTRODUCTION

In recent years, massively multiplayer online games (MMOGs) have become more and more popular. For example, World of Warcraft [1], one of the most popular MMOGs, reached a record of 8.5 million subscribed players worldwide. And according to a report by ScreenDigist [2], 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 [3], Ventrilo [4], and Skype [5], 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 sends 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 supports not only telephoning over internet, but also group voice chatting (Skype conference call), in which several players can chat together with one of the player serving the role of the host to receive, mix and deliver voice data. However, since Skype only support group voice chatting for several users, it is not suitable for MMOGs, which usually have many players chatting together. Teamspeak, Ventrilo, and Skype may be used as a voice chatting tool for MMOGs. However, their interactivity is yet to be improved since they are based on static group membership (i.e., the membership of a group is fixed or seldom changed) and a user thus has to a priori 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 [6]. 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 latency 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. Comparisons of the proposed schemes with other related work is given in Section 6. Finally, concluding remarks are drawn in Section 7.

II. 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 is 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 complex 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 AOI by exchanging messages extensively among players according to specific protocols [7, 8, 9].

2.2. Immersive Audio Systems

The paper [10] proposes an *immersive audio communication system* for MMOGs. The system allows a 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 sends the voice packet directly to other players in the hearing range. Due to the direct sending, the system provides low latency 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 latency. As shown in [10], the distributed server

architecture has shorter latency than the centralized server one, but has longer latency than the peer-to-peer one. However, it poses more complexity in the control and the coordination of distributed servers.

The paper [11] proposes a peer-to-peer based immersive audio streaming system for MMOGs. To provide with audio immersive experience, a voice from a near player would be louder than that from a farther player, and the voice would be louder when two players are talking face to face. The paper [11] uses Voronoi diagram to find out the *connecting neighbors* for a peer (i.e., player) to directly connect with. It also uses an audio mixing model for mixing audio data of connecting neighbors with consideration of neighbors' positions and audio directions. Let the strength of player *i*'s voice be S_i and the angle between the direction of player *i*'s voice and the direction from player *i* to player *j*, one of *i*'s connecting neighbors, be $\theta_{i,j}$. The voice strength sent from *i* to *j* is proportional to $S_i \times cos(\theta_{i,j} / 2)$. It is the largest when $\theta_{i,j} = 0$; it is the smallest (actually 0) when $\theta_{i,j} = \pi$. Suppose player *i* receives a voice stream of strength $S_{n,i}$ from a connecting neighbor *n* and the angle between the direction from *n* to *i* and the direction from *i* to *j* is $\theta_{n,j}$. Player *i* will also forward the voice stream just received to neighbor *j*, and the strength sent is proportional to $S_{n,i} \times cos(\theta_{n,j} / 2)$. Each peer in the system gathers the voice streams of all connecting neighbors, mixes them with its own voice stream according to the above-mentioned audio mixing model, multiplies the mixed stream by a *fading factor*, and at last sends a separate, faded, mixed stream to every neighbor per time step. Note that the fading factor for player *i* to send voice stream to player *j* is a function of the distance from *i* to *j*. A longer distance implies a smaller fading factor, which in turn makes the audio stream fade quicker.

Since the number of connecting neighbors of a player is usually small, the audio immersive system is thus scalable. However, the system has the *echo effect* that the audio signal sent by a player can loop back to itself after as few as three hops of packet transmission. Due to the voice mixing model, a voice stream may be delivered to a player far away from the voice originator. Therefore, it is hard for the system to support a definite voice transmission area (or hearing area). That is, the system cannot easily confine the transmission of a voice stream to a certain area, such as AOI, in an MMOG space. Moreover, the system also has the *multiple path effect* that a player's voice is propagated through different intermediate players to reach a certain player. For example, if player *i* has two connecting neighbors *n* and *m* which in turn have a common connecting neighbor *j*, then *j* will receive *i*'s voice stream twice, via *n* and *m*, respectively. Since the multiple path effect makes a voice stream be delivered to a node multiple times via different paths of different latency, extra audio mixing processes are caused and the resultant mixed audio may contain repeated, fading voices.

2.3. Human Conversational Speech Model

The article [12] 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. Moreover, once a person is talking in a conversation, 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

III. 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, 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 are 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 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, either server-based or peer-to-peer based, 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.

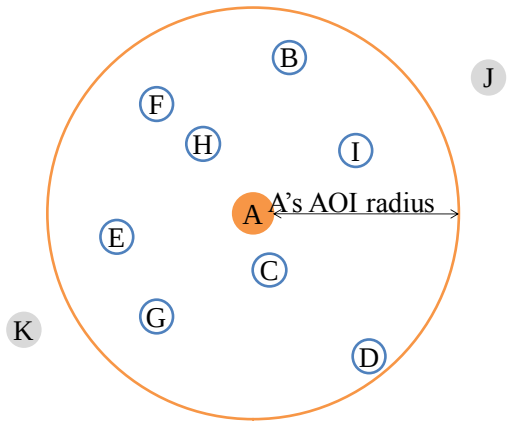


Fig 1. The AOI (Area of Interst) of a node A

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 acquires AOI neighbor information from the MMOG system. After that, the node delivers voice packets to every AOI 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.

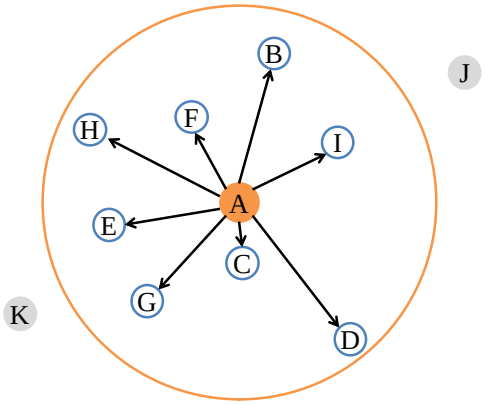


Fig. 2 Illustration of NimbusCast

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.

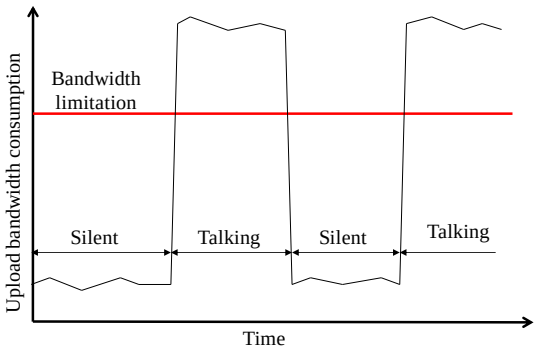


Fig 3. Network bandwidth consumption in NimbusCast

IV. 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 latency but without bandwidth overloading. Therefore, we have the following two design goals for the proposed schemes.

- ***Reasonable latency***

  ITU-T recommendation G.114 [13] provides a guideline about the one-way end-to-end (or mouth-to-ear) latency. It says that most users are satisfied with latency between 150 ms to 250 ms, while latency below 400 ms may also be tolerable by users. According to the recommendation, the latency of the proposed schemes should be less than 400 ms.

- ***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 latency longer. Thus, we should have a systematic way to perform the voice packet forwarding with reasonable latency. 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 (QuadCast)

In QuadCast, instead 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. The possibility that the bandwidth consumption of the speaking node exceeds the bandwidth limitation is thus lower. Therefore, the dropping rate is reduced 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.

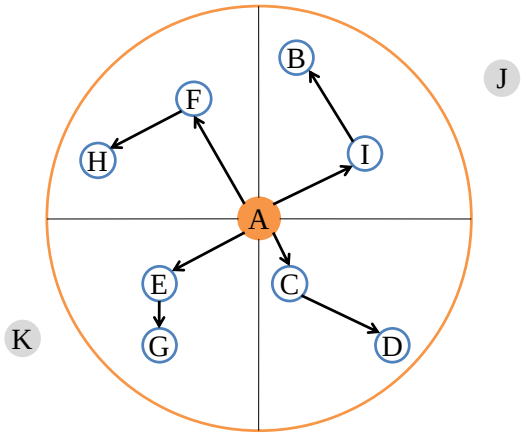
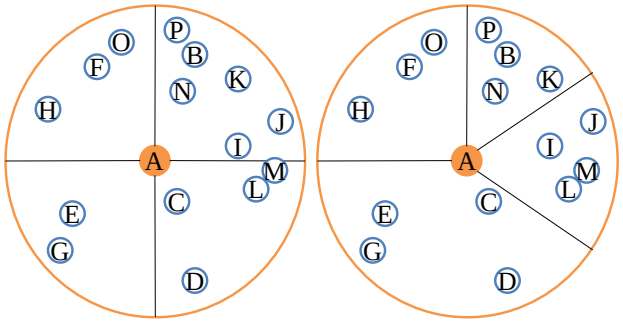


Fig 4. Quadrant-based voice packets forwarding

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 latency will be shorter and the quality of service will be better.



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

Fig 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 latency 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.

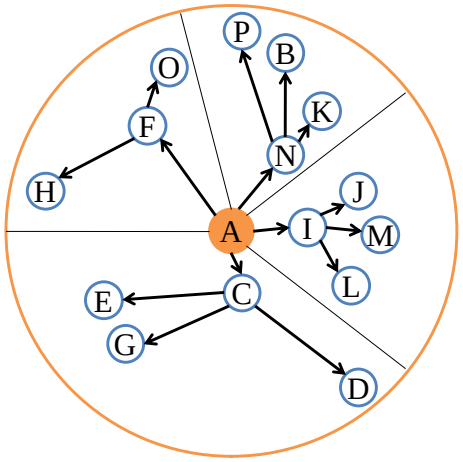


Fig 6. Sector-based voice packets forwarding

4.3. Packet Aggregation

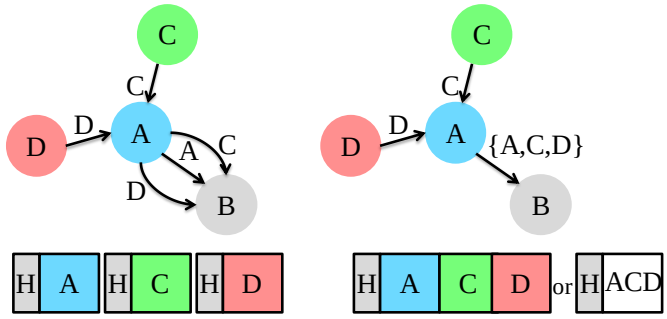
In QuadCast and SectorCast, an FA might forward different packets to a same recipient or to a same group of recipients. These packets are sent separately. However, if we can apply aggregation techniques to merge packets, the bandwidth consumption can be reduced dramatically. Below we propose two aggregation techniques: *header sharing (HS)* and *voice mixing (VM)*[1].

Time axis is divided into fixed-length (e.g., 40 ms) periods, each of which is called a *time bucket*. In every time bucket, a node gathers voice packets from all neighboring nodes and merges the packets having a same recipient or a same group of recipients to be an *aggregated packet* for delivering at the end of the time bucket. The HS scheme

[1]There is a mixer mechanism proposed in RTP which is similar to the packet aggregation techniques. As described in RFC 3550 (RTP: A Transport Protocol for Real-Time Applications), a mixer is a RTP-level entity that receives streams of RTP data packets from one or more sources, possibly changes the data format (coding), combines them into a single stream, and then forwards a new RTP packet containing the stream. Since the RTP mixer and the packet aggregation techniques are similar, we can take RTP as an implementation option.

merges packets by putting a common header for the packets, while the VM scheme merges packets by mixing their 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 aggregated packet by sharing the same header or by mixing the voice contents. As a result, packet traffic is lowered and bandwidth consumption is reduced.

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. All voice packets received successfully within a same time bucket are candidates to be aggregated. And each of the packets is assigned a unique ID of a 2-power number. 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. It is noted that an FA assigns IDs temporally and locally on the basis of time buckets. That is, each FA has its own ID assignment and IDs are re-assigned for every time bucket. Since the time bucket is usually very short, the number of packets to be aggregated is usually small. Therefore, some 2-power IDs suffice to represent all packets received in a time bucket, and an ID takes only a small number of bits for representation.



(a) Forwarding without aggregation (b) Forwarding with aggregation

Fig. 7 Example of packet aggregation

For example, in Figure 8, there are three nodes A, C, and D talking simultaneously. We assume the voice packets of these three nodes are assigned the IDs 1, 2, and 4, respectively. We also assume that node A is the FA of nodes C and D to send voice packets to nodes B and E. Node A should send the voice packets from nodes A, C, and D to node B, and thus the ID of the aggregated voice packet for B is 7, the addition of 1, 2, and 4. Node A should also send the voice packets from nodes A and D to node E, so the ID of the aggregated voice packet for E 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 according to the aggregated packet 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 the nodes of the corresponding group into the attached recipient list.

Appendix A shows a detailed example of voice packet aggregation by 2-power number addition. We can observe that the new aggregated audio packets always have shorter recipient list than the original audio packets. This is because the new recipient list is either one of the original lists or the intersection of some original lists. As we will show,

this also leads to shorter latency.

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.

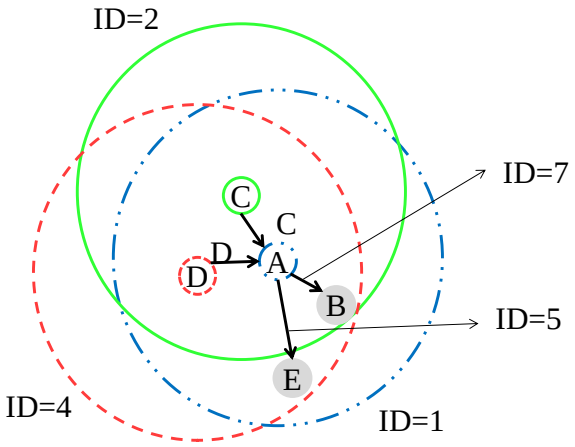


Fig. 8 The concept of packet aggregation by 2-power number addition

In QuadCast, when a node talks, it appends its ID to the voice packet and delivers 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.

4.5. Adaptive Forwarding

Both QuadCast and SectorCast are based on the concept of forwarding to reduce the burst upload bandwidth requirement. It may eliminate the bandwidth overload problem; however, it may also cause longer latency. When the latency is too long, the voice data is no more useful due to the timeliness requirement of the voice chatting.

To avoid latency from getting too long while retaining the benefit of the forwarding concept, we propose the *adaptive forwarding* mechanism, which has 5 steps and is shown below. Some notations for representing the mechanism are introduced in Table 2. The basic concept of adaptive forwarding is to send a voice packet to all recipients directly if bandwidth is affordable. The mechanism just needs to determine which packets should be sent directly and which packets should be sent via the forwarding of FAs. Actually, packets with 4 or less recipients are certainly sent directly. And packets with smaller sizes are to be sent directly with higher priority; the recipient list length is used to break ties when packet sizes are the same. This calls for the sorting in Step 1. Note that a packet may be directly sent to partial recipients when available bandwidth is not so high. That is, a packet is sent to some recipients directly and sent to FAs for forwarding it to other recipients. Also note that when the available bandwidth is inadequate (i.e., when *RB* in Step 2 is less than 0) to send packets to all FAs, the original QuadCast or SectorCast schemes will be used to send the packet and adaptive forwarding is not adopted.

Step 1: **Sort** all aggregated packets in a time bucket

according to the lexicographic order of the two-tuple (packet size, recipient list size). Suppose that the sorted packets are P_1, P_2,…, P_k. (Packets with smaller sizes and shorter recipient lists precede.)

Step 2: **Calculate** $RB=B-\sum_{i=1}^{k} \text{FA}(P_i)\times\text{size}(P_i)$

Step 3: **FOR** i=1 to k **DO**

IF (list(P_i) ≤ 4) **THEN Mark** P_i as *DirectSending*;

ELSE IF ((list(P_i) − FA(P_i))×size(P_i) < RB) **THEN**

Mark P_i as *DirectSending*;

$RB=RB-(\text{list}(P_i)-\text{FA}(P_i))\times\text{size}(P_i)$;

IF (RB=0) **THEN Break** loop;

Step 4: **IF** (RB>0) **THEN**

FOR i=1 to k **DO**

IF (P_i is unmarked and extra(P_i) ≥ 1) **THEN**

Mark P_i as *PartiallyDirectSending*;

Remove extra(P_i) recipients from the recipient list to the direct sending list of P_i;

$RB=RB-\text{extra}(P_i)\times\text{size}(P_i)$;

IF (RB=0) **THEN Break** loop;

Step 5: **Send** a *DirectSending* packet to each of its recipients;

Send a *ParitiallyDirectSending* packet to each member of its direct sending list, and to FAs for relaying the packet to all members in the recipient list;

Send an unmarked packet to its FAs for relaying it to all members in the recipient list;

Table 2. Notations for representing the adaptive forwarding mechanism

Notation	Meaning
RB	a variable for storing the *remaining bandwidth*
B	standing for the *bandwidth limitation*
size(P_i)	a function returning the size of packet P_i
list(P_i)	a function returning the number of recipients (in the recipient list) of packet P_i
FA(P_i)	a function returning the number of FAs of packet P_i (Note that FA(P_i) will return list(P_i) if list(P_i) is 4 or less. Otherwise, FA(P_i) will return the number of FAs according to QuadCast or SectorCast. In this way, the remaining bandwidth *RB* can be calculated accurately in Step 2.)
extra(P_i)	extra(P_i) = *RB* / size(P_i)

V. Evaluation

In this section, we perform simulation experiments for comparing the proposed AOI voice chatting schemes, QuadCast and SectorCast, with the base scheme — NimbusCast. 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; the time bucket length is set to be 40 ms and each step length is also set to be 40 ms. In each step, every node moves along a random direction by a distance of 4. 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 40 bytes (say, composed of four ITU-T G.729 frames, each with a 10-byte size for 10 ms duration [14]).

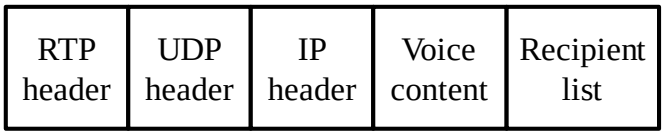


Fig. 9 Format of a forwarding voice packet

We first perform experiments for NimbusCast, QuadCast, and SectorCast under the assumption that each node has an upload bandwidth limitation of 32k bytes/sec (256 kbps). 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 10. In Figure 10 and the following figures, "-HS" in the legend stands for the aggregation method of header sharing; "-VM", voice mixing. We can see that QuadCast and SectorCast consume more upload bandwidth than Nimbus- Cast, especially when the number of nodes is more than 600. This is because the burst bandwidth consumption of NimbusCast exceeds the bandwidth limitation frequently and the so-called bandwidth overloading problem occurs, while the problem does not occur so frequently for the forwarding based schemes, QuadCast and SectorCast. We can also see that by applying packet aggregation, the bandwidth consumption of QuadCast and SectorCast is reduced, and the reduction is proportional to the number of nodes.

Figure 11 shows the packet dropping rates of all three schemes. We can observe that in this figure, the dropping rate of NimbusCast is over 40% when the number of nodes in the system reaches 1000, while the dropping rates of other schemes are below 5%. In [15], 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.

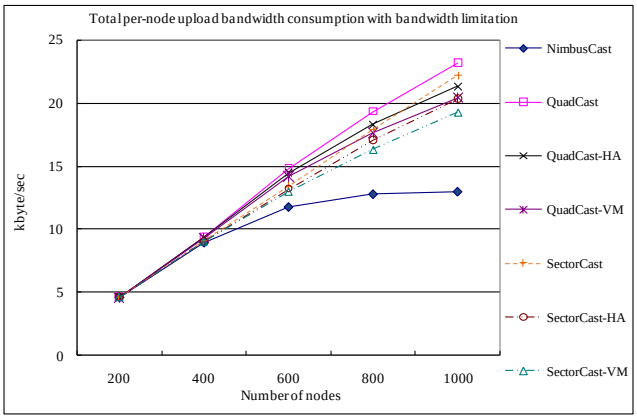


Fig. 10 Total per-node upload bandwidth consumption with bandwidth limitation for AOI voice chatting schemes

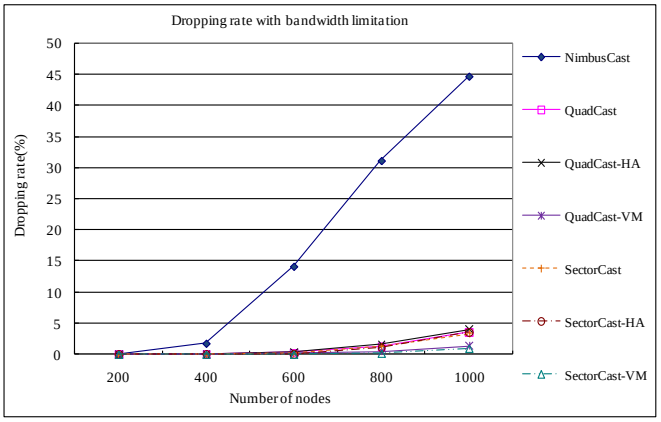


Fig. 11 Dropping rates for AOI voice chatting schemes with bandwidth limitation

We also measure the latency for all schemes when bandwidth limitation is considered. The latency is regarded to be the packet propagation time plus the packet transmission time. We adopt the assumption of end-to-end delay proposed in [16]. That is, the packet propagation delay between two directly connected nodes and the packet processing time are assumed to be 70ms and 30ms, respectively. Besides, the 40 ms time bucket period for voice packet collection is also considered. To be more precise, the one-hop packet delay contains the waiting time in a time bucket (between 0 ms and 40 ms), the packet propagation delay (70 ms), and the

packet processing time (30 ms). This means that when a voice packet is forwarded through one more hop, the latency is increased by 120 ms in average (the waiting time in a time bucket is 20 ms in average). Figure 12 shows the results for the average latency of the three schemes. NimbusCast has the shortest average latency about 120 ms and QuadCast has about 210 ms latency when the number of nodes is 1000. SectorCast has shorter latency than QuadCast because it evenly divides the AOI neighbors into four sectors, which yields shorter latency. We can also observe that the packet aggregation mechanism indeed helps reduce the latency of QuadCast and SectorCast schemes. As shown in Section 4, packet aggregation makes recipient lists shorter. It is noted that a packet is transmitted directly to each recipient if the recipient list has less than or equal to 4 nodes. Therefore, the latency is reduced when the recipient list is shorter.

Figure 13 shows the simulation results of the maximum latency of the three schemes. As shown in the figure, NimbusCast has the shortest maximum latency (140 ms) because a node may deliver a voice packet directly to an AOI neighbor at the end of the time bucket in the worst case. QuadCast has the longest maximum latency, with some cases having latency larger than 400 ms. Fortunately, the ratio of packets undergoing such latency is very small. Figure 14 shows the latency distribution of QuadCast. We can easily check that most packets are transmitted with latency less than 400 ms. As we have mentioned in Section 3, the latency of a conversation should not exceed 400 ms, and latency below 250 ms is considered to be of good quality. We can conclude that QuadCast and SectorCast have acceptable latency by our simulation results.

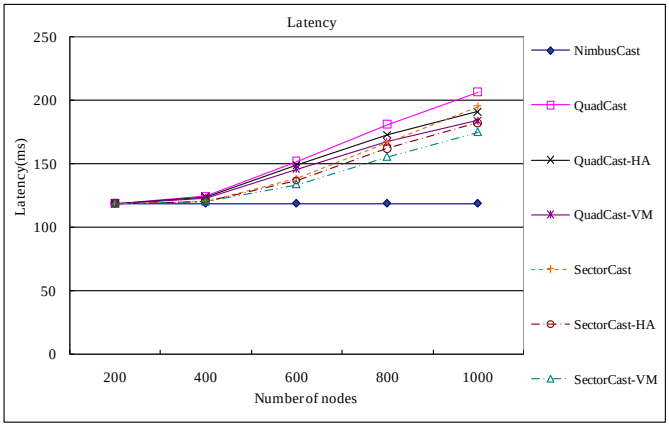


Fig. 12 Average latency for AOI voice chatting schemes with bandwidth limitation

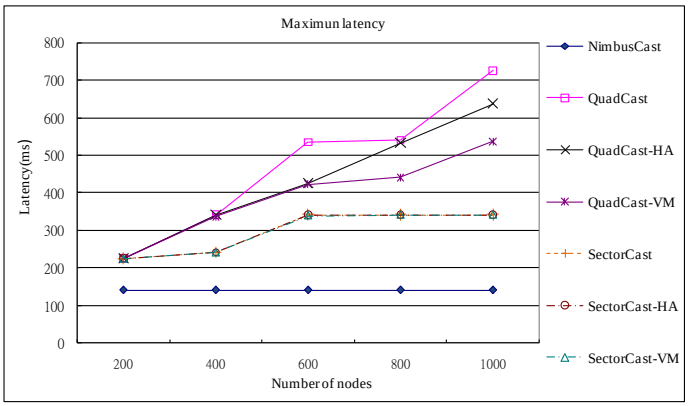


Fig. 13 The maximum latency for AOI voice chatting schemes with bandwidth limitation

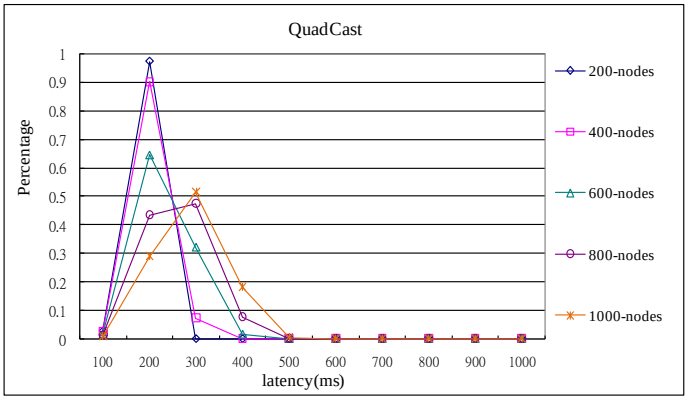


Fig. 14 Latency distribution for QuadCast with bandwidth limitation

VI. COMPARISON

In this section, we compare the proposed schemes, QuadCast and SectorCast, with Teamspeak [2], Ventrilo [3], Skeype [1] and the immersive audio system [10]. The comparison results are shown in Table 3. Both Teamspeak and Ventrilo support group voice chatting and are based on the client/server architecture. When a user of a group talks, voice packets are transmitted to the server, which mixes voice contents of all group users and sends the mixed contents to each group user. Since the server receives voice packets from all clients, mixes packets and then sends the mixed packets to all clients in real time, the number of users supported is limited. Moreover, Teamspeak and Ventrilo deliver voice on the basis of static group membership; they cannot confine the delivery area of the voice according to user positions in an MMOG. Skype is peer-to-peer based; it needs no dedicate server for mixing voice packets, and supports not only telephoning but also static group voice chatting (Skype conference call), in which several players can chat together with one of the player serving the role of the *host* to receive, mix and deliver voice data. Skype does not support definite voice transmission area in the MMOG space, neither.

The immersive audio system [10] is peer-to-peer based. It uses Voronoi diagram to find out connecting neighbors for a peer to directly connect with. For every time step, a peer collects audio streams from all connecting neighbors, mixes them with its own voice stream according to an audio mixing model. The peer then multiply the mixed stream by a fading factor, and sends a separate mixed audio stream to every neighbor per time step. The audio immersive system is scalable since the number of connecting neighbors of a peer is usually small. However, the system has the echo effect and the multiple path effect and it is hard for the system to support a definite voice transmission area (or hearing area). Below, we use an example in Figure 15 to illustrate the echo effect and the multiple path effect and to explain the reason why the system fails to support the definite voice transmission area.

Figure 15 partially shows the voice packet relaying scenario of the immersive audio system [10]. In the figure, we assume that node A talks and sends voice data V_{AB} and V_{AC} to nodes B and C, respectively. We also assume that the voice transmission area is confined to AOI when voice data is forwarded by intermediate nodes; therefore, node B will not deliver A's voice data to node D since D is not A's AOI neighbor. When B receives the voice data from A and finds that C is also A's AOI neighbor, B mixes its own voice data with A's (i.e., V_{AB}) and sends the mixed voice data V_{BC} to C. We can see that C obtains A's voice data from both the path A→C and the path A→B→C, which illustrates the multiple path effect. On receiving V_{AC} and V_{BC}, node C mixes them with its own voice data and sends the mixed data V_{CA} and V_{CD} to nodes A and D, respectively. On receiving V_{CA}, node A obtains its own faded voice data, which illustrates the echo effect. We can see that node D can also obtain voice data of A by receiving V_{CD}, which is a case that the system fails to achieve definite voice transmission area even though nodes indeed check sender's (speaker's) AOI area before forwarding voice data.

Table 3. Qualitative comparison of the proposed schemes with others

Scheme	System Architecture	Scalability	Nodes Doing Mixing	Definite Voice Area	Multiple Path Effect	Echo Effect
Teamspeak [3]	client/server	low	server	no	no	no
Ventrilo [4]	client/server	low	server	no	no	no
Skype Conference Call [5]	peer-to-peer	low	host node	no	no	no
Immersive Audio System [11]	peer-to-peer	high	all nodes	no	yes	yes
QuadCast	peer-to-peer	high	speaker and chosen FAs	yes	no	no
SectorCast	peer-to-peer	high	speaker and chosen FAs	yes	no	no

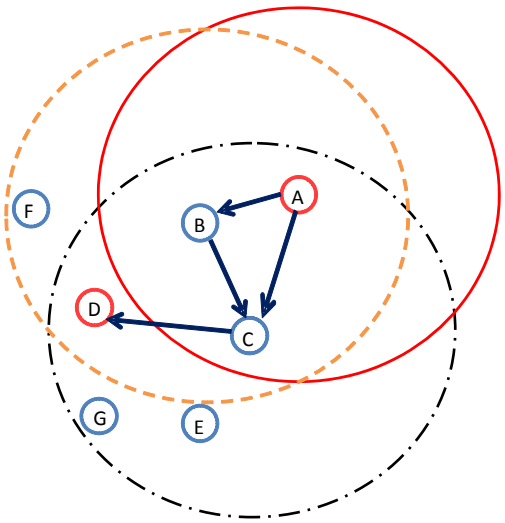


Fig. 15 An example of immersive audio system [11]

VII. 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 latency; 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. They can run on either a client/server or a peer-to-peer based MMOG, only if the MMOG can provide proper AOI neighbor information. 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. When several packets are to be sent to a same recipient, the packets can further be merged by the header sharing (HS) or by the voice mixing (VM) mechanisms to save bandwidth. 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 cause longer latency. To avoid latency from getting too long while retaining the benefit of the two schemes, we propose the adaptive forwarding mechanism, which is to send a voice packet to all recipients directly if bandwidth is affordable. As shown by the simulation results, this mechanism can reduce the latency dramatically.

In the original QuadCast and SectorCast design, a node closest to the talking node (or sender node) in the MMOG plane is chosen as the FA for the quadrant or sector. However, the MMOG plane topology is different from the underlying network topology. If we can take network topology into consideration and make a speaking (or sending) node choose nodes with shorter round trip time (RTT) as FAs, the latency can definitely be reduced. The RTT between two nodes can be measured and stored when the nodes enter each other's AOI for the first time. The RTT between two nodes usually varies slightly for near time instances, so the stored RTT records may be valid for a while. Therefore, a node usually has RTT information for all AOI neighbors whenever it is to select FAs for forwarding voice packets. We are planning to implement QuadCast and SectorCast on top of VON [7] with consideration of physical network topology in the near future.

REFERENCES

[1] World of warcraft, http://www.worldofwarcraft.com/.

[2] Western world mmog market: 2006 review and forecasts to 2011, http://www.screendigest.com/reports/07westworldmmog/readmore/view.html/.

[3] Teamspeak, http://www.goteamspeak.com/.

[4] Ventrilo, http://www.ventrilo.com/.

[5] Skype, http://www.skype.com/.

[6] K. Morse, L. Bic, and M. Dillencourt. "Interest management in large-scale virtual environments," *Presence: Teleoperators and Virtual Environments*, vol. 9, no. 1, pp. 52–68, 2000.

[7] S.Y. Hu, J.F. Chen, and T.H. Chen. "VON: a scalable peer-to-peer network for virtual environments," *IEEE Network*, vol. 20, no. 4, Jul./Aug. 2006, pp. 22-31.

[8] J. Keller and G. Simon. "Solipsis: A massively multi-participant virtual world," *Proc. of PDPTA*, pp. 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] 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.

[11] 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.

[12] I. T. Union. *ITU-T Recommendation P.59, Artificial conversational speech*, 1993.

[13] I. T. Union. *ITU-T Recommendation G.114, One-way transmission time*, 2003.

[14] I. T. Union. *ITU-T Recommendation G.729, Coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited*

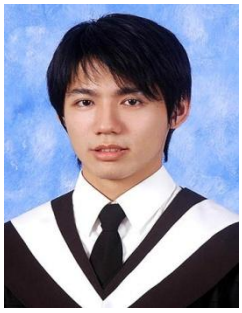
linear prediction, 1996.

[15] L. Ding and R. Goubran. "Assessment of effects of packet loss on speech quality in VoIP," *Proceedings of the 2nd IEEE International Workshop on Haptic, Audio and Visual Environments and Their Applications (HAVE 2003)*, pp. 49–54, 2003.

[16] R. Zimmermann and L. Liu. "ACTIVE: adaptive low-latency peer-to-peer streaming," *Proceedings of SPIE*, vol. 5680, pp. 26–37, 2005.

Jehn-Ruey Jiang received his Ph. D. degree in Computer Science in 1995 from National Tsing-Hua University, Taiwan, R.O.C. He joined Chung-Yuan Christian University as an Associate Professor in 1995. He joined Hsuan-Chuang University in 1998 and became a full Professor in 2004. He is currently with the Department of Computer Science and Information Engineering, National Central University, Taiwan. He was a recipient of Best Paper Award of the 32nd International Conference on Parallel Processing, 2003, and editors of Journal of Information Science and Engineering and International Journal of Ad Hoc and Ubiquitous Computing in 2004 and 2005, respectively. He has organized the 1st, 2nd and 3rd International Conference on Peer-to-Peer Networked Virtual Environments in 2007, 2008 and 2009, respectively. His research interests include distributed algorithms, algorithms for peer-to-peer networks, algorithms for mobile ad hoc networks and algorithms for wireless sensor networks.

Hung-Shiang Chen received his B.S. degree in Electrical and Control Engineering from the National Chiao-Tung University, Taiwan, and received the M.S. degree in Computer Science and Information Engineering from the National Center University, Taiwan, R.O.C., in 2007. His research interests include peer-to-peer networks.



Chao-Wei Hung received his B.S. degree in Mathematics form National Center University, Taiwan, R.O.C., in 2008. He is currently a M.S. student in the Department of Computer Science and Information Engineering of the National Center University, Taiwan, since 2008. His research interests include peer-to-peer networks.

Appendix A

In the Appendix, we show a detailed example of the voice packet aggregation by 2-power number addition. In the example, we assume that a forwarding assistant (FA) node Z needs to forward six voice packets, V1,..., V6 which have IDs of 2-power numbers 1, 2, 4, 8, 16 and 32, and that each packet has a specific recipient list as shown in Table A-1.

Table A-1. The voice packets, their IDs and recipient lists

Voice packet	ID	Recipient list
V1	1_{hex} (1_{bin})	{A, C, E, G, H}
V2	2_{hex} (10_{bin})	{B, F, K, M, N}
V3	4_{hex} (100_{bin})	{A, C, E, G, H, I, O}
V4	8_{hex} (1000_{bin})	{A, C, D, E, G, H, J, L}
V5	16_{hex} (10000_{bin})	{B, D, F, I, J, K, M, N, O}
V6	32_{hex} (100000_{bin})	{D, I, J, L, O}

For each recipient X, FA node Z scans all voice packets that X should receive and add up the IDs of the packets (see Table A-2). For example, node D should receive voice packets V4, V5 and V6, which have IDs of 8, 16 and 32, and the summation of the packets' IDs is 56.

Table A-2. Recipients and the ID summation of the voice packets they should receive

Recipient	Voice Packets to the Recipient	Summation of the packet IDs
A	V1, V3, V4	13_{hex} (001101_{bin})
B	V2, V5	18_{hex} (010010_{bin})
C	V1, V3, V4	13_{hex} (001101_{bin})
D	V4, V5, V6	56_{hex} (111000_{bin})
E	V1, V3, V4	13_{hex} (001101_{bin})
F	V2, V5	18_{hex} (010010_{bin})
G	V1, V3, V4	13_{hex} (001101_{bin})
H	V1, V3, V4	13_{hex} (001101_{bin})
I	V3, V5, V6	52_{hex} (110100_{bin})
J	V4, V5, V6	56_{hex} (111000_{bin})
K	V2, V5	18_{hex} (010010_{bin})
L	V4, V6	40_{hex} (101000_{bin})
M	V2, V5	18_{hex} (010010_{bin})
N	V2, V5	18_{hex} (010010_{bin})
O	V3, V5, V6	52_{hex} (110100_{bin})

Afterwards, node Z gathers recipients of the same ID summation to establish a new aggregated voice packet (see Table A-3). For example, nodes D and J have ID summation of 56, and they both need to receive voice packets V4, V5, V6. Therefore, a new audio packet NV3 that aggregates V4, V5 and V6 is built, and such a voice packet is with the recipient list {D, J}.

Table A-3. New voice packets, their IDs and recipient lists

New voice packet	ID	Recipient list
NV1=(V1, V3, V4)	13_{hex} (001101_{bin})	{A, C, E, G, H}
NV2=(V2, V5)	18_{hex} (010010_{bin})	{B, F, K, M, N}
NV3=(V4, V5, V6)	56_{hex} (111000_{bin})	{D, J}
NV4=(V3, V5, V6)	52_{hex} (110100_{bin})	{I, O}
NV5=(V4, V6)	40_{hex} (101000_{bin})	{L}