peerTalk: A Peer-to-Peer Multi-Party Voice-Over-IP System

Xiaohui Gu Zhen Wen Philip S. Yu Zon-Yin Shae

***Abstract*— Multi-party voice-over-IP (MVoIP) services allow a group of people to freely communicate with each other via Internet, which have many important applications such as on-line gaming and tele-conferencing. In this paper, we present a peer-to-peer MVoIP system called peerTalk. Compared to traditional approaches such as server-based mixing, peerTalk achieves better scalability and failure resilience by dynamically distributing stream processing workload among different peers. Particularly, peerTalk decouples the MVoIP service delivery into two phases: mixing phase and distribution phase. The decoupled model allows us to explore the asymmetric property of MVoIP services (e.g., distinct speaking/listening activities, unequal in-bound/out-bound bandwidths) so that the system can better adapt to distinct stream mixing and distribution requirements. To overcome arbitrary peer departures/failures, peerTalk provides light-weight backup schemes to achieve fast failure recovery. We have implemented a prototype of the peerTalk system and evaluated its performance using both large-scale simulation testbed and real Internet environment. Our initial implementation demonstrates the feasibility of our approach and shows promising results: peerTalk can outperform existing approaches such as P2P overlay multicast and coupled distributed processing for providing MVoIP services.**

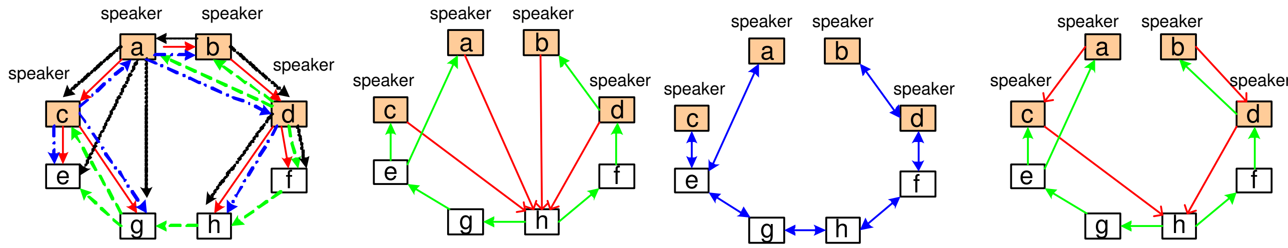
***Index Terms*— Peer-to-Peer Streaming, Voice-Over-IP, Adaptive System, Service Overlay Network, Quality-of-Service, Failure Resilience**

I. Introduction

Recent Internet advancement has made large-scale live streaming a reality [37]. Although previous work has studied the feasibility of supporting stream content delivery using peer-to-peer (P2P) architectures (e.g., [15], [14], [7], [21], [13], [12]), little is known whether it is feasible to provide large-scale multi-party voice-over-IP (MVoIP) services using application end-points such as peer hosts. The MVoIP service allows a group of people to freely communicate with each other via Internet, which can be used in many important applications such as massively multi-player on-line gaming [10], [20], tele-chorus, and online stock trading. Different from conventional conferencing systems that impose explicit or implicit floor controls, we strive to provide a more flexible MVoIP service that allows any participant to "speak" at anytime. By speaking, we mean not only uttering words, but also nonverbal activities such as shouting, singing, cheering, and laughing that are common in interactive and spontaneous applications such as on-line gaming. For example, in the Internet gaming application, MVoIP services allow game players to easily communicate with each other for deploying strategies, and game spectators to cheer up players. The emerging collaborative distributed virtual environment applications such as inhabited television [28] and digital virtual world (e.g., Second Life [1]) can support large online communities and highly interactive social events where it is common to have overlapping audio transmissions from multiple participants.

Traditional multi-party conferencing systems employ either multicast (e.g., [16], [15], [14], [7]) illustrated by Figure 1 (a), or server-based centralized audio mixing (e.g., H.323 multi-point control units) illustrated by Figure 1 (b). Using the multicast approach, the system needs to distribute multiple audio streams concurrently from all active speakers to all participants. Although multicast is well suited for broadcast applications that usually involve one active speaker, it becomes inefficient for interactive and spontaneous applications (e.g., on-line gaming) that often include many simultaneous speakers. The system can be overloaded by processing many audio streams concurrently. Moreover, since any participant is allowed to produce audio streams at any time, we need to maintain a large number of multicast trees for all participants, which can incur a lot of maintenance overhead especially in dynamic P2P environments where peers can dynamically leave or join the system. The audio mixing scheme can effectively reduce the number of concurrent streams, which first mixes the audio streams of all active speakers into a single stream and then distribute the mixed stream to all participants. However, centralized audio mixing lacks the scalability desired by P2P applications that often have large groups and many concurrent VoIP sessions. For example, the existing most popular VoIP system Skype [2] can only support conferencing sessions with at most five people. Previous work (e.g., [28], [22], [10]) has proposed coupled distributed processing (CDP) approach that uses the same tree for

Xiaohui Gu, Zhen Wen, Philip Yu, and Zon-Yin Shae are with IBM T. J. Watson Research Center. Email: {xiaohui, zhenwen,psyu, zshae}@us.ibm.com.



(a) Overlay multicast (b) Centralized mixing (c) Coupled distributed processing (d) Decoupled distributed procesing

Fig. 1. Design alternatives of multi-party voice-over-IP services.

both stream mixing and distribution, illustrated by Figure 1 (c). However, we observe that MVoIP services present asymmetric properties: (1) the number of active speakers (i.e., stream sources) is often different from the number of listeners (e.g., stream receivers), and (2) the in-bound bandwidth of a peer can be different from its out-bound bandwidth (e.g., cable network). Thus, CDP can be sub-optimal by using the same tree for both mixing and distribution.

In this paper, we present the design and implementation of the first P2P MVoIP system called peerTalk. Compared to previous work, our solution presents three unique features. First, peerTalk provides the first *decoupled distributed processing* (DDP) model for MVoIP services, illustrated by Figure 1 (d). The DDP model partitions the multi-stream audio delivery into two phases: (1) *mixing phase* that mixes audio streams of all active speakers into a single stream via a *mixing tree*; and (2) *distribution phase* that distributes the mixed audio stream to all listeners via a *distribution tree*. The decoupled processing model can better match the asymmetric property of the MVoIP application, which allows us to optimize and adapt to distinct stream processing operations (i.e., mixing or distribution) more efficiently. Second, peerTalk is *fully distributed* and *self-organizing*, which does not require any specialized servers or IP multicast support. The system provides scalable MVoIP services by efficiently distributing stream processing load among different peers. Thus, peerTalk can naturally scale up as more peers join the system. Third, peerTalk is *adaptive*, which can dynamically grow or shrink the mixing tree based on the current number of active speakers. During a MVoIP session, the number of active speakers can dynamically change over time. For example, in a P2P gaming application, there can be many active speakers at exciting moments while less speakers during quiet periods. Any static solution (e.g., predetermined aggregation tree at setup time) can either be over-sufficient that wastes system resources or under-sufficient that fails to meet workload requirements. Thus, peerTalk performs *continuous* optimization to adaptively optimize the quality of the MVoIP service in dynamic P2P environments.

The peerTalk system aims at supporting P2P applications (e.g., P2P gaming [20]) where MVoIP services are mostly applicable. However, compared to conventional distributed systems, P2P environments present more challenges due to higher failure frequency and arbitrary peer departures. The peerTalk system provides *failure-resilient* MVoIP services using a set of light-weight failure recovery schemes. First, the system maintains a number of backups for each mixer on the mixing tree by utilizing *redundant* resources in P2P environments. Thus, we can achieve fast failure recovery for time-sensitive VoIP applications by avoiding constructing a new mixing tree on-the-fly as much as possible. Second, similar to previous work [15], [6], [36], peerTalk adopts an overlay-based approach for failure resilience. We first connect peer hosts into an overlay mesh on top of IP network. The mixing tree and distribution tree are then built on top of the overlay mesh. Finally, we assume cooperative P2P environments where peers are willing to share resources with each other. The P2P VoIP service provides natural incentives for participants to share resources since they want to receive high-quality VoIP services with low cost.

We have implemented a prototype of the peerTalk system and conducted extensive experiments in both wide-area networks PlanetLab [27] and simulated P2P networks. Our experiments validate the feasibility of supporting MVoIP service using P2P systems and demonstrate the performance advantages of our approach compared to existing schemes. More specifically, our results show that (1) peerTalk can greatly reduce resource contentions in P2P environments compared to the overlay multicast approach, especially for MVoIP sessions with large group sizes and heavy workloads (i.e., many active speakers); (2) peerTalk achieves much lower service delay than the CDP approach by using separate trees; and (3) peerTalk can quickly recover MVoIP service failures while maintaining low resource contention and service delay among live peers. The rest of this paper is organized as follows. Section II introduces the peerTalk system model. Section III presents the detailed design and algorithms for P2P MVoIP service provisioning. Section IV presents the failure resilience management schemes. Section V presents the experimental results and analysis. Section VI discusses related work. Finally, the paper concludes in Section VII.

II. System Model

In this section, we introduce the peerTalk system model. First, we describe the MVoIP service model and its applications. Second, we present the overlay-based P2P VoIP system architecture. Third, we provide an overview of our approach to providing MVoIP services using a P2P system.

A. Multi-party VoIP Service Model

Multi-party VoIP services allow geographically dispersed participants to communicate with each other in a more natural way than other alternative solutions such as instant messaging. The basic MVoIP service model considered in this paper is that each participant is allowed to speak at anytime and should be able to hear the voices of all other active speakers. Different from conventional conferencing systems that often impose explicit or implicit floor control, the MVoIP service does not limit the number of participants who can "speak" and the time when participants can "speak". By speaking, we mean that participants produce any audio signals that could be not only uttering words, but also nonverbal activities such as singing, cheering, and laughing, or some background sound in a virtual environment (e.g., music). The MVoIP service has many interesting applications. For example, in increasingly popular multi-player Internet game applications [20], the MVoIP service allows both players and spectators to communicate naturally in realtime [35]. The players can better coordinate with each other for deploying strategies using audio than using instant text messaging. Moreover, the MVoIP service allows the game spectators to cheer up the players in more personalized ways [10]. Other important applications include Interactive Internet TV, Tele-immersions, audio-enabled tele-auctions, and collaborative virtual environments. All of the above applications have a common property that many participants can produce audio streams simultaneously. Moreover, the number of active speakers can change over time as the session's activeness changes.

B. Overlay-based System Architecture

The peerTalk system adopts an overlay-based approach for quality-of-service (QoS) management and failure resilience. Instead of constructing the mixing and distribution trees directly, peerTalk first connects peer hosts into an overlay mesh on top of existing IP network. The mixing and distribution trees are then constructed on top of the overlay mesh. Each peer is connected with a number of peers called neighbors via application-level virtual links called *overlay links*. Each overlay link between two peer hosts v_i and v_j, denoted by $l_{i,j}$, can be mapped to the IP network path between v_i and v_j. The number of neighbors to which a peer host can be connected is called the out-bound degree of the peer host, which is limited by the out-bound bandwidth at the peer host. Similarly, the in-bound degree of the peer host is constrained by its in-bound bandwidth. The overlay topology can dynamically change while each peer selects different neighbor peers. Specifically, to construct an overlay mesh with node degree k, each peer selects $[k/2]$ nearby peers as neighbors for network locality, and $[k/2]$ random peers as neighbors for failure resilience [31]. Remote random peers allow the overlay network to better survive correlated failures.

Each peer sends heartbeat messages to its neighbors to indicate its liveness and current stream processing performance (e.g., processing time and throughput). Each peer can keep up-to-date neighbor list and the neighbors' information based on the heartbeat messages. Each peer also periodically monitors the network delay to its neighbors and the bandwidth of the corresponding links using active probing [19]. Each peer maintains the routing cost (i.e., network delay) to every other peer and the path that leads to such a cost. The distribution tree rooted at each peer is constructed from the reverse shortest paths in similar fashion to DVMRP [16]. The mixing tree is dynamically constructed using the adaptive P2P mixing algorithm presented in Section III. The rationale behind the overlay-based approach include: (1) allowing each peer to maintain QoS information (e.g., CPU load) about its neighbors and the network QoS (e.g., network delay, data loss) of its adjacent overlay links from itself to its neighbors; (2) reducing tree repairing frequency by leveraging the resilience property of the overlay mesh that contains multiple redundant paths between every pair of peer hosts; and (3) leveraging previous overlay multicast solutions (e.g., [15]) for building the distribution tree.

C. Approach Overview

The peerTalk system provides the MVoIP service using a new P2P stream processing approach, which decouples the audio stream mixing from the audio stream distribution. Figure 2 shows a P2P MVoIP session with eight participants. Unlike conventional schemes (e.g., centralized mixing), peerTalk does not require any special servers and uses only end-systems of all participants,

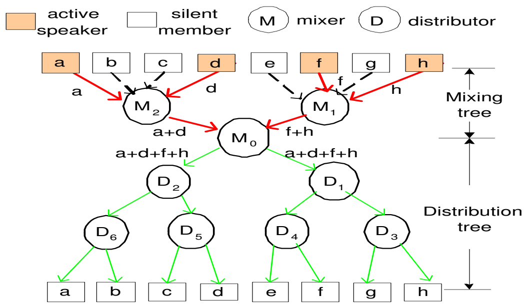


Fig. 2. Decoupled MVoIP service delivery model.

called peer hosts, to perform audio stream processing in a fully distributed and self-organizing fashion. Each MVoIP service session employs a set of audio stream processing components called *mixers* and *distributors*. The mixers and distributors are dynamically instantiated on different peer hosts based on their load conditions. Each mixer, denoted by M_i, has multiple input ports and a single output port. The mixer periodically aggregates the audio samples arrived at all input ports into one audio sample and normalizes the result to generate a mixed audio sample packet that is sent out via the output port. The mixer is the basic building-block in the mixing phase of the decoupled stream processing. In contrast, each distributor, denoted by D_i, has a single input port and multiple output ports. The distributor replicates each input audio packet into multiple copies that are sent out via the output ports. The distributor provides the basic function for the distribution phase.

Different from traditional client-server system, P2P system consists of end-system hosts, denoted by v_i. The peer host often has constrained resources such as limited memory for buffering audio packets received from networks, and low out-bound bandwidth (e.g., cable/DSL networks). However, the multi-stream audio processing is often resource intensive (e.g., large buffer requirements for many streams, large bandwidth requirement for sending/receiving packets), especially for large-scale MVoIP service sessions involving many participants. Thus, centralized stream processing becomes inapplicable in P2P environments since no single peer host can meet the resource requirements. To address the problem, the peerTalk system employs multiple peer hosts to collectively fulfill the task of audio stream processing. The peerTalk system first connects a number of mixers into a *mixing tree*, illustrated by the upper-level tree in Figure 2. The leaf nodes of the mixing tree consist of all participating peer hosts. We assume that each peer host performs silence suppression to save resources. If a peer host is a leaf node in the mixing tree, it generates audio stream only if the local participant produces any sound. The internal nodes of the mixing tree consist of serving peer hosts that provide audio mixing functions. Since the number of active speakers can dynamically change, the audio mixing workload varies over time. The peerTalk system can dynamically grow or shrink the mixing tree to adapt to the number of active speakers. For the distribution phase, we leverage the existing overlay multicast solution (e.g., [14], [15]) to construct a *distribution tree* to disseminate the mixed audio stream from the root of the mixing tree to all listening participants, shown by the lower-level tree in Figure 2. Note that the internal nodes M_i and D_i in the mixing tree and distribution tree can be instantiated on peer hosts that belong to different VoIP sessions. In this paper, we assume that peers are willing to share their resources when they join the system. Some research work has addressed the problem of enforcing fair resource sharing [25], [11] in P2P systems, which however is not the focus of this paper. We also assume that peer hosts are trust-worthy and secure audio transmissions can be achieved using cryptography schemes.

Compared to the multicast approach, our scheme has an extra mixing delay. However, the audio mixing phase can greatly reduce the network traffic and the stream processing load by reducing the number of concurrent streams each peer has to handle and distribute across networks. On the other hand, the height of the mixing tree is often much smaller than that of the distribution tree since the active speakers often constitute a small subset of all participants. The mixing tree delay is thus relatively small compared to the distribution tree delay that needs to cover all participants. Moreover, different from the multicast approach that has to use different multicast trees rooted at active speakers, peerTalk always uses the optimal multicast tree that has the smallest distribution delay. As a result, peerTalk can be more efficient than the multicast approach, especially for highly active, large-scale sessions with many active speakers and participants.

III. P2P VoIP Service Provisioning

We now present a fully distributed algorithm for dynamically constructing and adapting the audio mixing trees in P2P environments. The basic idea of our approach is to adaptively distribute dynamic audio stream mixing workload among different peer hosts while continuously optimizing the service quality of different MVoIP sessions.

A. Service Provisioning Protocol

We now present the VoIP service session provisioning protocol in the peerTalk system, illustrated by Figure 3. At a session beginning, all participants of the session run an election protocol to select the best peer as the

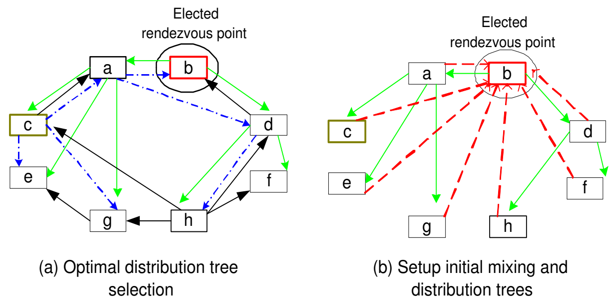


Fig. 3. MVoIP service session setup protocol.

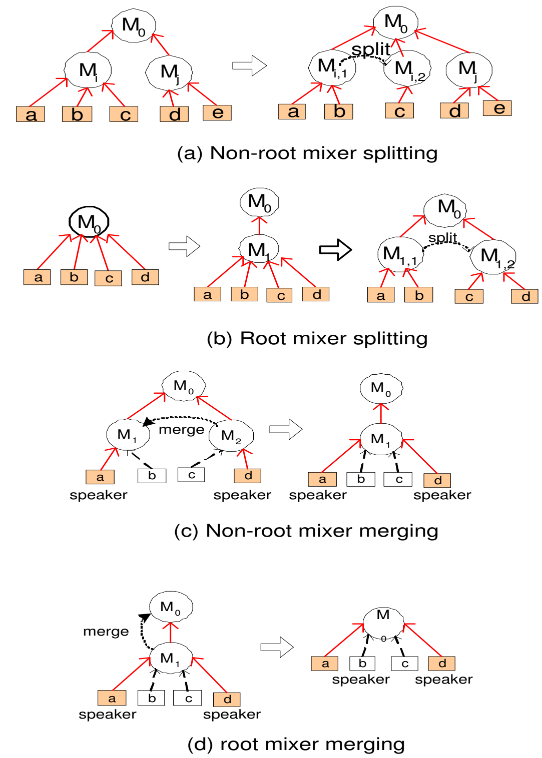


Fig. 4. Mixer splitting and merging operations.

rendezvous point that serves as the root of both mixing tree and distribution tree. Different from the multicast approach where each active speaker uses a different tree to disseminate the audio stream to all participants, peerTalk only uses one distribution tree to send the mixed audio stream to all participants. This provides an optimization opportunity for the system to employ the best multicast tree for the distribution phase. Thus, we want to place the rendezvous point on the peer host that is the source of the best multicast tree. In the current peerTalk system, the best multicast tree is the one that has the minimum average delay between the source and all other participates[1]. When two multicast trees have similar distribution delays, we choose the one that has larger mixing capacity. Specifically, all peers concurrently run the DVMRP algorithm to construct multicast trees rooted at themselves. Each peer measures the average delay of its own multicast tree and then propagates the delay information plus its mixing capacity to all other members via the overlay mesh. All peers then select the same best peer as the rendezvous point. For example, in Figure 3 (a), all eight participants initiate the multicast tree construction algorithm and then select the peer b as the rendezvous point.

Initially, the mixing tree only includes the root mixer instantiated on the rendezvous point, illustrated by Figure 3 (b). All participants are connected to the root mixer as its children. During runtime, the system adaptively grows or shrinks the mixing tree based on the dynamic mixing workload changes using a fully distributed algorithm. First, the root mixer monitors the number of active speakers among all participants. If the number of active speakers is larger than the number that the root mixer can handle, it spawns new child mixers on other peer hosts to offload the audio mixing workload. The basic idea of mixing tree adaptation is that each mixer can either split itself if it is overloaded or merge with its sibling mixers if it is under-loaded. The mixer is also dynamically migrated among different peer hosts to achieve improved service quality. We now describe the distributed algorithms for mixer splitting, mixer merging, and mixer migration, respectively.

[1]We can use different criteria for selecting the root mixer. We use the distribution tree delay as the primary selection criteria because the distribution delay often accounts for a major part in the end-to-end voice packet delay. We can also use different composite metrics based on the network conditions and audio mixing requirements.

B. Mixer Splitting

Each mixer M_i in the mixing tree monitors the number of audio streams concurrently arrived at its input ports. Since peers can perform silence suppression, a leaf node on the mixing tree generates an audio stream only if the local participant produces any sound. An internal node on the mixing tree generates an output audio stream if any of its input ports receives an input stream. Suppose the mixer M_i has n input ports denoted by $I_1, I_2, ..., I_n$. We use time-series $A_k, 1 \leq k \leq n$ to describe the data arrival pattern at the input port I_k. The time-series A_k consist of a sequence of time-stamped number denoted by $a_k \in A_k$. At time t, we set $a_k = 1$ if there are data arriving at the input port I_k, or $a_k = 0$ if no data arrives. Hence, the total number of audio streams concurrently arrived at the mixer M_i at time t, denoted by $\Omega_i(t)$, can be calculated as $\Omega_i(t) = \sum_{k=1}^{n} a_k$. To achieve stability, we use moving average value of total audio stream number at time t, denoted by $N_{i,t}$. $N_{i,t}$ can be computed by the exponential smoothing algorithm as follows,

$$N_{i,t} = \alpha \cdot N_{i,t-1} + (1-\alpha) \cdot \Omega_i(t), 0 < \alpha < 1 \quad (1)$$

For conciseness, we omit the t in $N_{i,t}$ and use N_i to represent the moving average value of total audio stream number at current time t.

Since peer hosts are often resource constrained, they can only process a limited number of audio streams while keeping up with the input stream rate without dropping data. Let us consider the mixer M_i located on the peer host v_i that can process at most C_i streams. The mixer M_i triggers the splitting process if the number of arriving audio streams exceeds its processing limit, i.e., $N_i > C_i$. If the overloaded mixer M_i is not the root mixer, it splits itself into two mixers $M_{i,1}$ and $M_{i,2}$, illustrated by Figure 4 (a). One of them $M_{i,1}$ remains on the host v_i and is assigned a subset of the children of M_i whose aggregate workload is $\lceil \frac{C_i}{2} \rceil$. The rest of the children are assigned to the new mixer $M_{i,2}$. The peer host v_i then selects its most lightly-loaded neighbor v_j to host $M_{i,2}$. If the workload of $M_{i,2}$ still exceeds the processing limit of v_j, the mixer $M_{i,2}$ continues to split itself until the workload of each new mixer is within the processing limit of its hosting peer. Note that the above process may trigger the parent of M_i to split since the number of its children is increased.

If the overloaded mixer M_i is the root mixer, i.e., $M_i = M_0$, the peer host v_i first creates a new mixer M_1 and transfers all the children of M_0 to M_1, illustrated by Figure 4 (b). The new mixer M_1 then becomes the only child of M_0 and is migrated to one of the neighbors of v_i that has the largest available stream processing capacity. By doing so, the height of the mixing tree is thus increased by one. Let us assume M_1 is placed on the peer host v_j. If the workload of M_1 still exceeds the capacity of v_j, M_1 performs the same splitting as the previous case since M_1 is not the root mixer. All spawned new mixers become the children of the root mixer M_0. To minimize the average workload for all input streams, we distribute the children of M_i to each new spawned mixers $M_{i,1}..., M_{i,k}$ based on the data arrival time series $A_1, ..., A_n$. We calculate the correlation coefficient between every two data arrival time series A_i and A_j, which indicates the possibility of concurrent data arrivals on the input ports I_i and I_j. We then allocate least correlated input streams to the same mixer to minimize the average aggregate workload at each mixer.

C. Mixer Merging

We now present the mixer merging algorithm illustrated by Figure 4. The mixer merging process can effectively shrink the mixing tree to avoid excessive audio mixing overhead (delay, packet loss) by minimizing the number of mixers traversed by the audio streams. Similar to the mixer splitting process, each mixer M_i monitors the number of audio streams concurrently arrived at its input ports. If the total workload N_i is significantly less than the mixer's processing capacity C_i (e.g., $N_i < \lfloor \frac{C_i}{2} \rfloor$), the mixer seeks to merge with its succeeding sibling M_j in the mixing tree. If the aggregate workload of M_i and M_j is within the processing limit of a single mixer, i.e., $N_i + N_j \leq max(C_i, C_j)$, we merge the two mixers into one mixer. If $C_i \leq C_j$, we delete M_i and connect the children of M_i to M_j. Otherwise, we delete M_j and connect the children of M_j to M_i. Note that the above process may trigger the parent of M_i and M_j to perform mixer merging since the input stream number of the parent mixer decreases. If a mixer M_i becomes the only child of its parent mixer M_p, we can merge M_i with M_p to reduce the height of the mixing tree. The situation occurs when the children of M_p merge with each other into one mixer. Figure 5 shows the psudo-code of the mixer merging algorithm. To avoid system thrashing between mixer splitting and mixer merging, peerTalk requires that mixer merging cannot be triggered within certain time threshold if the mixer is just partitioned from the other mixer.

D. Mixer Migration

The peerTalk system performs dynamic mixer migration to continuously optimize the audio mixing process. We can migrate a mixer M_i from a peer host v_i to one of the neighbors of v_i if the neighbor peer is better in terms of (a) larger stream processing capacity because of more abundant CPU, memory and network bandwidth resources; (b) better network connection (i.e., less delay or packet loss) from the children of M_i to M_i, and then from M_i to the parent of M_i; and (c) higher availability [9]. Each of these criteria can lead to different peer host comparison results. Thus, the peerTalk system allows the upper-level application to prioritize these different criteria for customized decision-making. For illustration, let us assume that criteria (a), (b), and (c) has decreasing priorities.

Each mixer M_i on the peer host v_i periodically probes the neighbor hosts of v_i in the overlay mesh to decide whether migration should be triggered. Let us assume v_i has k neighbors $v_1, ..., v_k$. The mixer M_i sends the addresses of its parent M_p and children $M_1, ..., M_n$ to all of its neighbor hosts $v_j, 1 \leq j \leq k$. The mixer M_i then asks each neighbor to return a set of information including (1) current stream processing capacity, (2) average delay/packet loss from $M_1, ..., M_n$ to v_j and from v_j to M_p, and (3) failure probability of v_j. The mixer M_i first selects qualified neighbor hosts whose processing capacity can satisfy the current workload of

Procedure: $Merge(M_i, M_j)$
Pre-conditions: M_p: parent of M_i and M_j
begin
if $(N_i < [\frac{C_i}{2}]) \wedge (N_i + N_j \leq max(C_i, C_j))$
  if $C_i \leq C_j$
  then merge M_i into M_j
  else merge M_j into M_i
  if M_p has only one child M_k
    if M_p is not the root mixer
      if M_p can handle all workload
      then merge M_k into M_p
      else merge M_p into M_k
    if M_p is the root mixer $\wedge$ $(N_k + N_p \leq C_p)$
      then merge M_k into M_p
end

Fig. 5. Mixer merging algorithm.

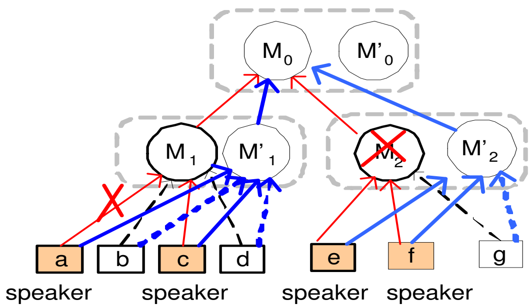


Fig. 6. Failure recovery in mixing tree.

M_i. If qualified neighbor hosts exist, M_i further selects the best neighbor host that has (1) minimum worst-case delay/packet loss, and (2) lowest failure/departure probability. If the best neighbor host is *significantly* better than the current host v_i, the mixer M_i is migrated to the selected neighbor host[2].

To achieve smooth mixer migration, the system first creates a new mixer M_i' on the selected neighbor host and connects M_i' to the parent of M_i and the children of M_i. In the meantime, the system still uses M_i to serve the current MVoIP session. When M_i' finishes the setup, the children of M_i is notified to send audio streams to M_i'. The old mixer M_i is then deleted. Since the mixer M_i' may be instantiated on a more powerful peer host, the mixer migration can trigger the mixer merging process. Hence, the mixer migration can not only improve the performance of the current mixing tree but also help to consolidate the mixing tree so as to reduce intermediate mixers during the stream mixing process.

IV. Failure Resilience Management

We now present a set of light-weight schemes to improve the system's resilience to peer failures and churns. By failure resilience, we mean that the system should be able to quickly recover an MVoIP session from end-system or network failures with minimum service interruption. Compared to dedicated servers, peer hosts are more prone to failures. Hence, failure resilience management becomes particularly important in P2P environments[3].

A. Mixer Replication

We design a proactive replication-based failure recovery mechanism to tolerate fail-stop failures of networks and peer hosts, illustrated by Figure 6. Different from a reactive approach that dynamically finds a replacement for the primary upon failure, our replication-based approach is proactive by maintaining a number of backups in advance. For example, in Figure 6, each of the three mixers M_0, M_1 and M_2 maintains one backup mixer for itself. During the VoIP session, no audio data are sent to the backup mixer. However, the primary mixer needs to periodically probe its backup mixers to monitor their liveness and resource availability. The motivation of the proactive approach is two-fold. First, P2P environments provide plentiful redundant resources for hosting backup replicas. Second, the proactive approach can avoid constructing a new mixing tree on-the-fly if backup mixers are still usable. Thus, we can achieve *fast* failure recovery for time-sensitive VoIP services. Each mixer in the mixing tree, called the primary, maintains a number of backup replicas on different peer hosts.

Let us assume a primary mixer wants to maintain k backup mixers. As we mentioned before, each mixer periodically probes its neighbor hosts to decide whether one of them is better for hosting the mixer. At the same time, the primary mixer can identify k qualified peer hosts to host replicas. If less than k qualified peer hosts are found, the primary mixer probes the neighbors of its neighbors until k replicas are instantiated. During runtime, the primary mixer periodically probes its replicas to check their liveness and update the states of all replicas. If one of replicas becomes unavailable, the primary mixer tries to find another qualified peer host in its nearby neighborhood to host the replica. When the primary mixer is migrated to a new peer host, the replicas are also migrated to the neighbors of the new peer host to assure that backups are still close to the primary for localized replica maintenance.

The number of replicas represents the trade-off between failure resilience and replication overhead. If the primary maintains k replicas up all the time, the primary can survive $k-1$ concurrent replica failures. Note that the

[2]For stability, mixer migration is triggered only if the performance of the neighbor host is better than the current host by a certain threshold value.

[3]We can leverage previous resilient overlay multicast solutions (e.g., [8], [34]) to achieve failure resilience in the distribution phase. Thus, our research focuses on the mixing phase of MVoIP service delivery.

roles of different mixers are non-uniform to the failure-resilience of the mixing tree. The higher level mixers in the mixing tree are more important than the lower level mixers because they are responsible for aggregating the output streams of those lower level mixers. Thus, we propose a differentiated mixer replication scheme to maintain more replicas for higher level mixers in the mixing tree. The motivation of differentiated replication is to maximize the overall failure resilience of the MVoIP service under limited replication overhead.

B. Failure Detection

The failure of the mixing tree can be caused by either network failures between peer hosts or end-system failures. We do not distinguish graceful failures (quitting with notification) from fail-stop failures (crashes/quiet leaving) although the graceful failures can be handled more efficiently. For example, we can request the quitting peer to continue working until the system finishes switching to one of its replicas.

When replicas stop receiving the heartbeat messages from the primary, they assume that the primary fails[4]. Replicas then execute an election algorithm to reach a consensus on which replica should take over based on a pre-defined election criteria (e.g., smallest peer identifier). The elected replica then contacts the parent and the children of the failed primary mixer. The parent and the children of the failed mixer then drop the connections to the failed primary mixer and connect to the new primary mixer[5]. For example, in Figure 6, when the primary mixer M_2 fails, the replica M_2' takes over the audio mixing process for the participants e, f, g and connects to the parent mixer M_0.

C. Churn Management

In contrast to conventional client-server systems, P2P systems exhibit a high rate of continuous node arrivals and departures, which is called churn. The peerTalk system reacts to churn according to different roles of peers in the MVoIP service: (1) *participant* that produces and receives audio streams; (2) *overlay router* that provides application-level forwarding in the overlay mesh; (3) *mixer* that provides audio mixing service, (4) *distributor* that distributes audio streams to multiple receivers; and (5) *backup* that hosts mixer replicas.

Peer joins. When a peer wants to join an existing MVoIP session, it is first incorporated into the P2P overlay mesh by an out-of-band bootstrap mechanism [15]. The peer selects a few peer hosts provided by the bootstrap service as neighbors and also requests a few other peers to add itself as a neighbor. After the peer successfully joins the overlay mesh, it becomes an overlay router that can forward packets for its neighbors. The peer then broadcasts a message to other peers via the overlay mesh requesting to join the MVoIP session. The peer can acquire the session ID from the bootstrap service. If any peer that is already in the session receives the requesting message, it replies the message with the address of the mixer M_i to which it is connected. The peer then connects to the mixer M_i according to the first reply it receives and ignores other later replies. Thus, the peer is successfully added into the mixing tree by becoming a child of M_i. The overlay multicast algorithm can connect the new peer into the distribution tree. While the peer stays in the system, the peer can be selected to play the role of mixer, distributor or backup.

Peer departures. When a peer v_i leaves the system without pre-notice (i.e., crash/disconnection), the system first needs to repair the overlay mesh and updates membership lists on other live peers. The neighbors of v_i can detect the departure of v_i after they stop receiving the heartbeat messages from v_i for an extended period. The system then updates the mesh by deleting v_i from the neighbor lists of all other live peers. The mesh can become partitioned because of the departure of v_i. The system can repair the partitioned mesh by adding more overlay links at partitioned peers [15]. If v_i also hosts a primary mixer M_i, the departure of v_i triggers dynamic failure recovery to repair the mixing tree with a replica of M_i. If v_i only acts as a backup for a primary mixer M_i, the departure v_i causes M_i to create a new backup replica.

V. Experimental Evaluation

We now present an experimental evaluation of the peerTalk system. We ran large-scale experiments on a network simulation environment and prototype experiments on the PlanetLab Internet testbed [27]. Our results demonstrate that (1) peerTalk that employs decoupled distributed processing (DDP) can achieve better MVoIP service quality than coupled distributed processing (CDP) and overlay multicast, two existing state-of-the-art schemes; (2) peerTalk can simultaneously achieve both low resource contention and short network delay while CDP has long network delay and overlay multicast tends to incur high bandwidth congestion; and (3) peerTalk can achieve failure resilience under P2P system churn by just maintaining a few backup mixers.

[4]The heartbeat messages are small messages sent with high frequency to ensure timely failure detection.

[5]The session transition may cause VoIP service glitch. To further reduce the failure impact, we can incorporate the failure prediction mechanism into the system to initiate the session transition protocol before the primary fails, which however is beyond the scope of this paper.

A. Evaluation Methodology

We have implemented a prototype of the peerTalk system and tested it on both simulation environments and the Planetlab Internet testbed [27]. The simulator performs packet-level, discrete-event network simulation. The simulator uses the degree-based Internet topology generator Inet-3.0 [41], [39] to generate a 5120 node power-law graph to represent the IP physical network. The delay of each physical link is distributed in the range of [8,12] ms similar to [15], which is proportional to the Euclidean distance between two end points. The bandwidth of each edge network link is distributed in the range of [256k, 10M]bps according to the capacity of current residential access networks (e.g., ADSL, cable networks). We also emulate asymmetric residential access networks (e.g., ADSL, cable networks) where the upload bandwidth is smaller than the download bandwidth. The inbound or outbound bandwidth of a core network node is proportional to the number of its inbound or outbound physical links. We have conducted experiments on different physical networks where link bandwidth follows either uniform or Zipf distribution.

To emulate mixer processing delays and peer relaying delays, each overlay node is configured with a certain mixing or relaying capacity denoting the amount of data the overlay node can mix or relay per second. We assign varied capacity values to different hosts to emulate heterogeneous environments. We then randomly select a number of stub nodes as application end-points (i.e., peer hosts). Each peer host is randomly connected to [5, 10] other peers as neighbors to emulate a scalable overlay mesh with low node degrees. The overlay topology is connected using the short-long algorithm presented in [31]. The simulator emulates packet routing at both IP-layer and overlay-layer using the Dijkstra shortest path algorithm based on the delay metric.

To demonstrate the efficiency of peerTalk, we compare our approach with CDP [28], [22], [10] and overlay multicast [15]. The CDP algorithm first selects the best multicast tree among all peers similar to the peerTalk system. But the CDP algorithm uses the same tree for both stream mixing and stream distribution. The overlay multicast uses the DVMRP algorithm [16] to construct multicast trees on top of the overlay mesh.

Previous study indicates that delay and loss are the key factors that decide the user's perception about the voice quality [23]. Hence, we use the following metrics to evaluate the service quality of an MVoIP service session: (1) *link stress* over all utilized physical links where the link stress of one physical link is defined as $\frac{RequiredBandwidth}{AvaliableBandwidth}$. Higher link stress implies larger network queueing delay and loss probability; (2) *node stress* over all utilized peer hosts where the node stress of one peer host is define as total amount of audio data the peer host needs to process over its processing capacity. Larger node stress implies larger stream processing delay and loss probability at peer hosts; (3) *propagation delay* of an MVoIP session is defined as the mean propagation delay from all active speakers to all listeners where each propagation delay denotes the network propagation delay over the network path for each voice packet travelling from one speaker to one listener; and (4) *service delay* of an MVoIP session is defined as the mean service delay from all active speakers to all listeners where each service delay includes network propagation delays, peer mixing delays, and peer distribution delays[6].

We use a range of different workloads to evaluate the performance of the peerTalk system. The voice encodings follow the G.711 standard [23] with 64Kbps codec bitrate, 80 bytes codec sample size, 10 ms codec sample interval. Each packet includes 40 bytes for IP/UDP/RTP headers and 160 bytes for voice payload. The stream rate is 50 packets per second. Thus, the total bandwidth per connection is 80Kbps. We use two different models to emulate the speaking activities: (1) *explicit ON/OFF model* that directly adjusts the number of active speakers to reflect speaking activity changes. The activity of each active speaker alternates between ON periods and OFF periods. During the ON period, a stream of voice packets is generated while no data is generated during the OFF period. The durations of the ON period and the OFF period are generated from two exponential distributions based on previous experimental study [23]; and (2) *real VoIP conversation data* that use real telephony conversations from switchboard data [17], which consist of 500 pairs of conversations for a total of 1000 voice streams. Each conversation session lasts 300 seconds. The original voice data have been converted to VoIP packets, and consisted of multiple pairs of users conversing on diverse topics. Unless otherwise specified, each simulation run lasts 300 seconds and has a certain warm-up period for the system to reach its stable performance.

B. Simulation Results

In the first set of experiments, we evaluate the performance of the peerTalk system under different session sizes, illustrated by Figures 7 - 10. The overlay network consists of 800 peers. We instantiate three MVoIP sessions concurrently, where the session size ranges from [50, 500] peers. The workload is generated using the explicit ON/OFF model that randomly selects 10% session

[6]The simulator emulates the propagation delay on physical links but does not emulate queueing delay, packet losses, or cross traffic for achieving large-scale simulations.

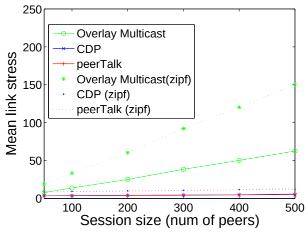


Fig. 7. Link stress under different session sizes.

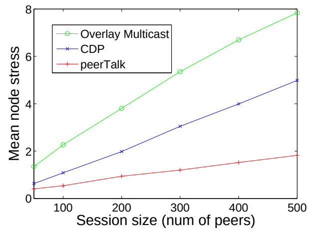


Fig. 8. Node stress under different session sizes.

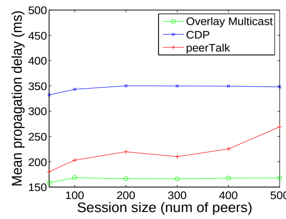


Fig. 9. Propagation delay under different session sizes.

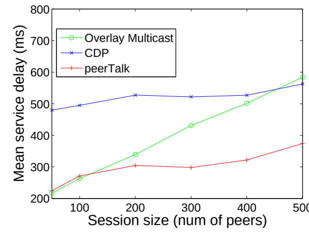


Fig. 10. Service delay under different session sizes.

members as active speakers[7]. Figure 7 shows the average link stress on all the physical links used by the three running MVoIP sessions under different algorithms. We conducted experiments using both uniform and Zipf network bandwidth distributions. We observe that although peerTalk typically employs smaller mixing trees than CDP, peerTalk can achieve similar link stress as CDP by employing explicit load balancing. Both approaches can achieve much lower link stress than the multicast algorithm, especially under large session sizes. The link stress reduction is even more prominent for the network with Zipf bandwidth distribution. The reason is that both peerTalk and CDP employ a multi-stream audio mixing phase that can greatly reduce the number of concurrent audio streams distributed across networks. This result indicates that both peerTalk and CDP incur much lower network congestion than the multicast approach, which implies lower network queueing delay and packet loss rate. Similarly, both peerTalk and CDP impose much lower node stress than the multicast approach, shown by Figure 8. Compared to the multicast scheme, both peerTalk and CDP have an extra mixing phase. We need to evaluate whether the mixing phase causes significant increase to network propagation delay during the audio stream delivery. Figure 9 shows the average network propagation delays achieved by different algorithms. The average network propagation delay is calculated among all the audio packets that are transmitted from all speakers to all listeners. We observe that peerTalk has much lower propagation delay than the CDP algorithm by using separate trees for mixing and distribution phases. Figure 10 shows the average service delay achieved by different algorithms as we increase the session size. The service delay includes network propagation delay, peer mixing delay, and peer distribution delay. The results show that peerTalk consistently achieves lower service delay than CDP and multicast approaches. Note that the real service delay of the multicast approach will be higher if we add the network queueing delay, which can be induced from the link stress results. The results show the advantage of decoupled processing model and adaptive stream mixing scheme employed by the peerTalk system.

Our second set of experiments compare the performance of different algorithms under different number of active speakers, shown by Figures 11 - 14. The number of active speakers is controlled by a speaker ratio that denotes the percentage of session members as active speakers. Every 10 seconds, we randomly select a number of session members as active speakers. Similar to the first set of experiments, we use a 800-node overlay network and concurrently run three MVoIP sessions. Each session includes 100 peers with [5%,30%] randomly selected active speakers. Figure 11 shows that both peerTalk and CDP have much lower link stress than multicast by employing audio mixing, especially under high speaker ratios. From Figure 12, we observe that peerTalk can achieve lower node stress than CDP because of its inherent load balancing capability. Figure 13 shows that peerTalk has much lower network propagation delay than CDP and adaptively expands the mixing tree as speaker ratio increases. Finally, Figure 14 shows the total service delay achieved by different algorithms. We observe that peerTalk can consistently achieve lower service delay than CDP and multicast approaches. Under low speaker ratio, peerTalk can employ a small mixing tree to avoid excessive mixing delay; under high speaker ratio, peerTalk can adaptively expand the mixing tree to handle high stream workloads.

Our third set of experiments studies how different algorithms scale as we gradually increase the number of concurrent sessions running on top of the overlay system, illustrated by Figures 15 - 18. In this set of experiments, we use a 800-node overlay network. Each session includes 50 randomly selected peers with 10% randomly selected peers as active speakers. Similar to previous two experiments, both peerTalk and CDP incur lower link stress and node stress than the multicast approach. Further, peerTalk achieves lower node stress than CDP by performing explicit load balancing using mixer migration. Overall, peerTalk consistently achieves lower service delay than CDP and multicast. We also observe that the service delay of the multicast approach

[7]The advantage of peerTalk is even more prominent under heavier stream workload with a larger number of active speakers, which is shown by the second set of experimental results.

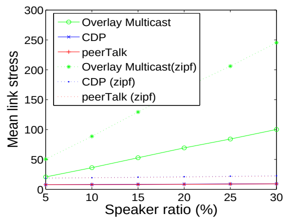


Fig. 11. Link stress under different speaker ratios.

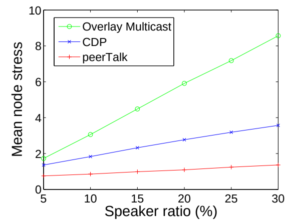


Fig. 12. Node stress under different speaker ratios.

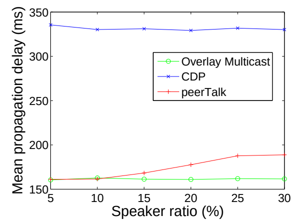


Fig. 13. Propagation delay under different speaker ratios.

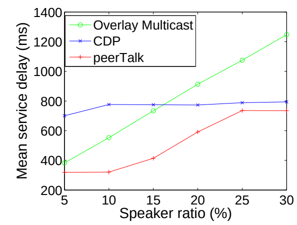


Fig. 14. Service delay under different speaker ratios.

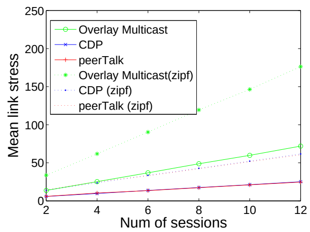


Fig. 15. Link stress under different session number.

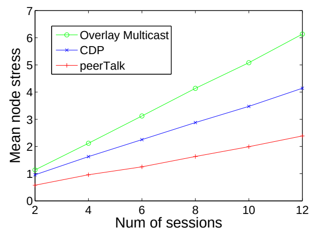


Fig. 16. Node stress under different session number.

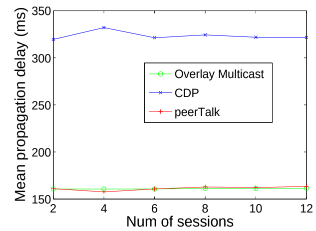


Fig. 17. Propagation delay under different session number.

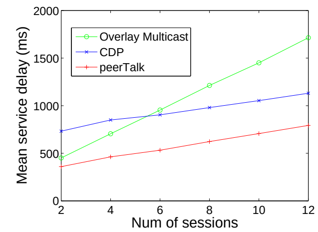


Fig. 18. Service delay under different session number.

increases much faster than peerTalk and CDP as more sessions are created on top of the overlay system. This results show that audio mixing is necessary in order to achieve scalable MVoIP services over P2P overlay networks.

We have also compared the performance of different algorithms using real VoIP conversation data. We use a 400-node overlay network and instantiate three MVoIP sessions concurrently on top of the overlay network. Each session consists of 20 peers. The speaking activity of each peer pair is the playback of one conversation trace selected from the 500 pairs of conversation trace files. Figure 19 and Figure 20 shows the link stress and total service delay of different algorithms under real workloads. The results show similar trend as the results under synthetic workloads. Both peerTalk and CDP can significantly reduce the link stress using audio mixing compared to the multicast approach. Overall, peerTalk consistently achieves lower service delay than the other two approaches.

We now evaluate the proactive failure recovery schemes of the peerTalk system under P2P network churn where a number of peers dynamically leave or join the system, illustrated by Figure 21 and Figure 22. The algorithm "backup-k" means that we maintain k backup mixers for each primary mixer. We use a 1000-node overlay network and instantiate three MVoIP sessions concurrently on top of the overlay network. Each session consists of 100 randomly selected peers with 10% speaking ratio. The system randomly selects a number of departure nodes every five seconds according to a specified churn rate. During each 300-second simulation run, we start from a low-churning system with $\delta = 10\%$ churn rate (i.e., 10% of total system peers randomly leave the system[8]), then increase the churn rate to 20% at time 100, and further increase the churn rate to 30% of all nodes at time 200. The system reconstructs the distribution tree using the DVMRP algorithm and repairs overlay mesh partition by randomly adding neighbors to the peers with few neighbors left. In Figure 21, the Y-axis shows the accumulated number of failures that cannot be recovered by the maintained backup mixers. In Figure 22, the Y-axis shows the failure frequency that denotes the number of failures that cannot be recovered by the peerTalk backup scheme every second. The "backup-0" algorithm represents the reactive failure recovery approach that takes no prevention action (i.e., no backup mixers/distributors). The fault tolerance improvement (i.e., failure number reduction) from "backup-0" to "backup-1" and from "backup-1" to "backup-2" is much larger than that from "backup-2" to "backup-3" and from "backup-3" to "backup-4". We observe that by maintaining four backup mixers, the system can recover most failures even under high system churn (i.e., up to 30% random failing peers).

C. *PlanetLab Results*

To evaluate the feasibility and performance of our approach under real Internet environment, we have deployed and evaluated the peerTalk system on the Planetlab wide-area network testbed [27]. The peerTalk software at each PlanetLab host includes five major modules:

[8]Some nodes will be dynamically added back to the system to keep the number of live nodes in the system at a constant level of $(1-\delta)\cdot N$.

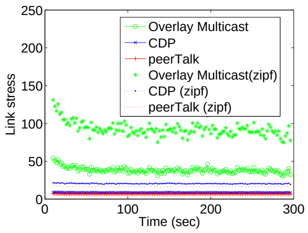


Fig. 19. Link stress under real VoIP workloads.

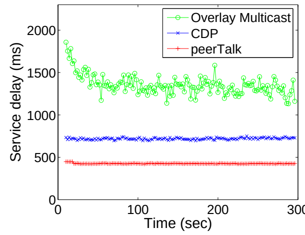


Fig. 20. Service delay under real VoIP workloads.

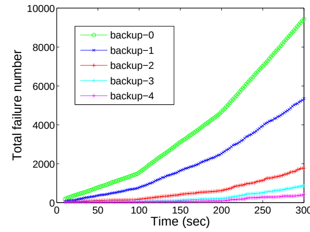


Fig. 21. Total failure number under system churn.

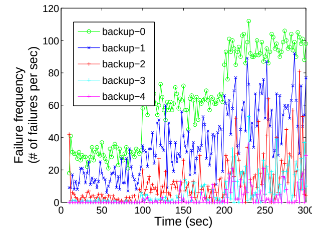


Fig. 22. Failure frequency under system churn.

(1) *mixer manager* executes the mixer splitting, mixer merging, and mixer migration algorithms; (2) *overlay topology manager* maintains the overlay mesh network; (3) *monitoring* module is responsible for monitoring the network/service states of neighbors (e.g., network delays); (4) *session manager* maintains the peer membership information about all VoIP sessions, which is built on top of the DHT system [33], [38], [30]; (5) *data transmission* module is responsible for sending, receiving, and forwarding audio data. We used the SCRIBE software [14] to realize P2P overlay multicast. To evaluate the feasibility of adaptive mixing, we have measured the average time of basic mixer adaptation operations (i.e., mixer splitting, mixer merging, mixer migration) in the real Internet setting. Our initial results indicate that peerTalk can finish the basic mixer adaptation operations between Planetlab hosts within a few milliseconds.

Different from the simulation that uses explicit model to generate workload, the prototype experiments used the ON/OFF workload model. We dynamically adjust the mean duration values of the ON period and the OFF period to emulate different speaking activities. The audio packets follow the standard G.711 codec requirements described in Section V-A. Our experiments used about 50 PlanetLab hosts that spread across US. We instantiate two peerTalk nodes on each PlanetLab host. At the beginning, each peer sends a probe message to all other peers via the SCRIBE multicast interface and measures average delay between itself and all other peers. All peers then exchange with each other the average delay from themselves to all other peers. All peers then select the best multicast tree that has the minimum average delay as the optimal distribution tree. The CDP uses the optimal tree for both mixing and distribution. The peerTalk uses the optimal tree for distribution and constructs the mixing tree using the adaptive stream mixing algorithm. The overlay multicast scheme uses SCRIBE to perform multi-stream distribution from all active speakers to all group members.

We fist test the three different algorithms under a light workload condition with few concurrent active speakers. Figure 23 shows the cumulative distribution of total delays between all pairs of communicating participants using the three different algorithms. Different from the simulation that only models network propagation delay, the packet delay measured on PlanetLab reflects all the processing and queueing delays at both peer hosts and Internet connections. We observe that peerTalk achieves the best performance (i.e., shortest service delays) while CDP has the worst performance. The reason is that under light workload condition, the advantage of audio mixing is not significant and the CDP suffers from large mixing delay. Besides packet delay, the quality of VoIP services is also affected by the inter-packet delay jitter [23]. The delay jitter describes the variations of inter-packet delays. Thus, we also measured the delay jitter result during the above experiment, which is illustrated by Figure 24. We observe that peerTalk can also achieve better delay-jitters than the other two schemes.

We then increase the system workload by increasing the number of concurrent active speakers. Figure 25 and Figure 26 show the cumulative distribution of total delays and delay-jitters achieved by different algorithms under a heavy workload. We observe that the effect of audio mixing becomes significant and peerTalk can achieve much lower delay than the other two alternatives. The multicast approach is completely overloaded by the number of concurrent streams, which have excessive total delays. The experimental results validate our hypothesis that adaptive audio mixing can greatly reduce network and stream processing delays by reducing the link stress and node stress. Such an improvement can offset the small extra mixing delay with a large margin in most cases compared to the multicast approach. Since peerTalk tends to perform more adaptations under heavy workload, peerTalk has slightly larger delay jitters than the other two schemes. However, such difference is marginal. Thus, we conclude that peerTalk can perform better than the two state-of-the-art approaches in real Internet environments.

VI. Related Work

In this section, we compare peerTalk with related work that is classified into three major categories: (1) *voice-over-IP systems*; (2) *peer-to-peer systems*; and (3) *distributed multimedia systems*.

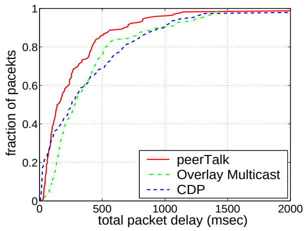


Fig. 23. Packet delays under a light workload.

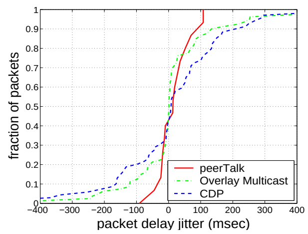


Fig. 24. Delay jitters under a light workload.

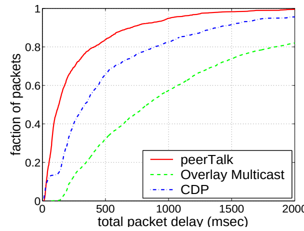


Fig. 25. Packet delays under a heavy workload.

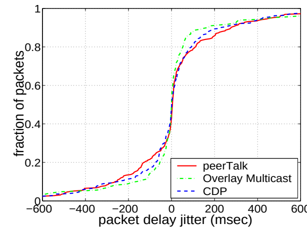


Fig. 26. Delay jitters under a heavy workload.

Voice-over-IP systems. Recently, VoIP systems have received a lot of research attention. Much of previous work has been devoted to evaluating and improving the quality of two-party VoIP services (e.g., [23], [40]). Ren et. al. [32] proposed an Autonomous-System-aware peer relay protocol to improve two-party VoIP quality. People have also studied the MVoIP (MVoIP) services that present more challenges. For example, Rangan et. al. proposed a hierarchical Media Mixing architectures for supporting large-scale audio conferencing [29]. Lennox and Schulzrinne developed a reliable MVoIP system using a full mesh topology [22]. Radenkosvic and Green-Halgh proposed a Distributed Partial Mixing approach to supporting MVoIP service with TCP fairness[28]. Different from previous work, the peerTalk system focuses on providing MVoIP services in P2P environments, which provides a pure application-level solution with unique features of self-organization, adaptation to workload, and failure-resilience. In [18], we have presented the basic adaptive mixer splitting and merging algorithms. This paper presents the complete peerTalk framework including the new algorithms for rendezvous point election, mixer migration, and failure resilience management.

P2P systems. With the popularity of P2P file sharing systems, P2P systems have drawn much research attention. One salient advantage of P2P systems is that they can aggregate a tremendous amount of resources in a failure-resilient and cost-efficient fashion. Previous work has addressed the problems of scalable data lookup using distributed hash table (DHT) (e.g.,[33], [38], [30]) and incentive engineering (e.g., [25], [11]) for providing efficient P2P data sharing. Inspired by P2P file sharing systems, researchers have proposed many other P2P applications such as P2P content delivery (e.g., [21], [13], [12]), P2P file systems (e.g., [3]), and P2P storage systems (e.g., [4]). While peerTalk can benefit from many previous P2P research results, our research is orthogonal to previous work. Our work more focuses on exploring the specific properties and requirements of MVoIP services. To the best of our knowledge, our work is the first study on using P2P stream processing for MVoIP services, which we believe could be a new killer application for the P2P technology.

Distributed multimedia systems. Many multimedia processing needs to be performed in a distributed fashion. For example, Amir et al. [5] proposed the active service framework and applied it on a media transcoding gateway service. In [26], Ooi and Renesse proposed a framework to decompose a computation into sub-computations and assign them to multiple gateways. In [24], Nahrstedt et al. proposed an Hourglass-based system to deliver composite multimedia content to users in pervasive computing environments. The peerTalk system is similar to the above work in terms of distributing media processing among multiple hosts. However, instead of considering generic media processing, our work more focuses on P2P audio stream processing for providing MVoIP services. Thus, the new contribution of the peerTalk system is to organize and adapt the audio stream processing based on the unique features of MVoIP services.

VII. Conclusion

Traditionally, multi-party voice-over-IP (MVoIP) services use a collection of multicast trees or a centralized audio mixing server. In this paper, we argue that a P2P MVoIP system can achieve better scalability and cost-effectiveness by adaptively and efficiently distributing stream processing workload among different peers. To the best of our knowledge, this is the first work that studied the P2P system design for the MVoIP application, which we believe could be a new killer application for the P2P technology. Specifically, this paper makes the following contributions: (1) we propose a novel *decoupled stream processing* model that can better explore the asymmetric property of MVoIP services and optimize the stream mixing and distribution processes separately; (2) we provide localized mixer splitting/merging/migration algorithms to continuously optimize the quality of the MVoIP services according to speaking activity changes; and (3) we propose light-weight backup schemes to make peerTalk resilient to peer failures/departures by utilizing redundant resources in P2P environments. We have implemented a prototype of the peerTalk system that are evaluated in both real-world wide-area networks and simulated P2P networks. Our results show that peerTalk can combine the advantages of two state-of-the-art approaches (i.e., multicast, audio mixing) while overcoming their disadvantages for providing MVoIP services.

References

[1] The Second Life 3D on-line digital world. *http://secondlife.com/*.

[2] The Skype Internet telephony system. *http://www.skype.com/*.

[3] T. Gil A. Muthitacharoen, R. Morris and B. Chen. Ivy: A Read/Write Peer-to-peer File System. *Proc. of Fifth Symposium on Operating Systems Design and Implementation (OSDI), Boston*, December 2002.

[4] A. Adya and et. al. FARSITE: Federated, Available, and Reliable Storage for an Incompletely Trusted Environment. *Proc. of the Fifth Symposium on Operating Systems Design and Implementation (OSDI'02), Boston, MA*, December 2002.

[5] E. Amir, S. McCanne, and R. H. Katz. An Active Service Framework and Its Application to Real-Time Multimedia Transcoding. *Proc. of ACM SIGCOMM 1998, Vancouver, B. C., Canada*, October 1998.

[6] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient Overlay Networks. *Proc. 18th ACM SOSP 2001, Banff, Canada*, October 2001.

[7] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable application layer multicast. *Proc. of ACM SIGCOMM, Pittsburgh, PA*, August 2002.

[8] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan. Resilient Multicast using Overlays. *Proc. of ACM SIGMETRICS, San Diego, CA*, June 2003.

[9] R. Bhagwan, K. Tati, Y. Cheng, S. Savage, and G. M. Voelker. TotalRecall: System Support for Automated Availability Management. *The ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, March 2004.

[10] A. Bharambe, V. Padmanabhan, and S. Seshan. Supporting Spectators in Online Multiplayer Games. *Proc. of HOTNETS-III, San Diego*, November 2004.

[11] A. Blanc, Y.-K. Liu, and A. Vahdat. Designing Incentives for Peer-to-Peer Routing. *Proc. of the IEEE Infocom Conference, Miami, FL*, March 2005.

[12] J. W. Byers, J. Considine, M. Mitzenmacher, and S. Rost. Informed Content Delivery Across Adaptive Overlay Networks. *IEEE/ACM Transactions on Networking, 12(5), pp. 767-780*, October 2004.

[13] M. Castro, P. Druschel, A-M Kermarrec, A. Nandi, A. Rowstron, and A. Singh. SplitStream: High-Bandwdith Multicast in Cooperative Environments. *Proc. of ACM SOSP 2003, New York*, October 2003.

[14] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron. SCRIBE: A large-scale and decentralized application-level multicast infrastructure. *Proc. of ACM SIGCOMM, San Diego, CA*, August 2001.

[15] Y.-H. Chu, S. G. Rao, S. Seshan, and H. Zhang. Enabling Conferencing Applications on the Internet using an Overlay Multicast Architecture. *Proc. of ACM SIGCOMM, San Diego, CA*, August 2001.

[16] S. Deering. Multicast routing in internetworks and extended lans. *Proc. of ACM SIGCOMM, Stanford, CA*, August 1988.

[17] D. Graff, K. Walker, and A. Canavan. Switchboard-2 phase ii. LDC 99S79 – http://www.ldc.upenn.edu/Catalog/. 1999.

[18] X. Gu, Z. Wen, P. S. Yu, and Z.-Y. Shae. Supporting Multi-Party Voice-Over-IP Services with Peer-to-Peer Stream Processing. *Proc. of ACM Multimedia, Singapore*, November 2005.

[19] Iperf. iperf bandwidth measurement tool. *http://dast.nlanr.net/Projects/Iperf/*.

[20] B. Knutsson, H. Lu, W. Xu, and B. Hopkins. Peer-to-Peer Support for Massively Multiplayer Games. *Proc. of IEEE INFOCOM 2004, Hong Kong*, March 2004.

[21] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat. Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh. *Proc. of ACM SOSP 2003, New York*, October 2003.

[22] J. Lennox and H. Schulzrinne. A Protocol for Reliable Decentralized Conferencing. *Proc. of ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'03), Monterey, CA*, June 2003.

[23] A. Markopoulou, F. Tobagi, and M. Karam. Assessment of VoIP Quality over Internet Backbones. *IEEE Transactions on Networking*, October 2003.

[24] K. Nahrstedt, B. Yu, J. Liang, and Y. Cui. Hourglass Multimedia Content and Service Composition Framework for Smart Room Environements. *Elsevier Journal on Pervasive and Mobile Computing*, 2005.

[25] T.-W. Ngan, D. Wallach, and P. Druschel. Enforcing Fair Sharing of Peer-to-Peer Resources. *Proc. of 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, February 2003.

[26] W. T. Ooi and R. V. Renesse. Distributing Media Transformation Over Multiple Media Gateways. *Proc. of the 9th ACM International Multimedia Conference, ottawa, CA*, September 2001.

[27] L. Peterson, T. Anderson, D. Culler, and T. Roscoet. A Blueprint for Introducing Disruptive Change in the Internet. *Proc. First ACM SIGCOMM HotNets Workshop*, October 2002.

[28] M. Radenkosvic and C. GreenHalgh. Multi-party Distributed Audio Service with TCP Fairness. *Proc. of ACM Multimedia 2002, Juan-les-Pins, France*, December 2002.

[29] P. V. Rangan, H. M. Vin, and S. Ramanathan. Communication Architectures and Algorithms for Media Mixing in Multimedia Conferences. *IEEE/ACM Transactions on Networking, Vol. 1, No. 1*, February 1993.

[30] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. *Proc. of the ACM SIGCOMM 2001, San Diego, CA*, 2001.

[31] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-Aware Overlay Construction and Server Selection. *Proc. of Infocom 2002,New York City, N.Y.*, June 2002.

[32] S. Ren, L. Guo, and X. Zhang. ASAP: an AS-Aware Peer-Relay Protocol for High Quality VoIP. *Proc. of IEEE ICDCS 2006, Lisbon, Portugal*, July 2006.

[33] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. *Proc. of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, November 2001.

[34] D. Rubenstein, S. Kasera, D. Towsley, and J. Kurose. Improving Reliable Multicast Using Active Parity Encoding Services (APES). *Journal of Computer Networks, Volume 44, Issue 1,*, January 2004.

[35] A. Singh and A. Acharya. Using Session Initiation Protocol to build Context-Aware VoIP Support for Multiplayer Networked Games. *Proc. of SIGCOMM 2004 Workshops, Portland, Oregon*, August 2004.

[36] A. C. Snoeren, K. Conley, and D. K. Gifford. Mesh Based Content Routing using XML. *Proc. 18th ACM SOSP 2001, Banff, Canada*, October 2001.

[37] K. Sripanidkulchai, A. Ganjam, B. Maggs, and H. Zhang. The Feasibility of Supporting Large-Scale Live Streaming Applications with Dynamic Application End-Points. *Proc. ACM SIGCOMM 2004*, August 2004.

[38] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. *Proc. of ACM SIGCOMM 2001, San Diego, California*, August 2001.

[39] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger. Network Topology Genrators: Degree-Based vs. Structural. *Proc. of ACM SIGCOMM 2002, Pittsburg*, Aug 2002.

[40] S. Tao and et. al. Improving VoIP Quality through Path Switching. *Proc. of IEEE Infocom 2005, Miami*, March 2005.

[41] J. Winick and S. Jamin. Inet3.0: Internet Topology Generator. *Tech Report UM-CSE-TR-456-02 (http://irl.eecs.umich.edu/jamin/)*, 2002.