VIRTUAL MIXER: REAL-TIME AUDIO MIXING ACROSS CLIENTS AND THE CLOUD FOR MULTIPARTY CONFERENCING

*Jun Liao, Chun Yuan, Wenwu Zhu, Philip A. Chou**
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing, China
*Microsoft Research, Redmond, USA

ABSTRACT

Traditional multiparty audio or video conferencing uses a single node, sometimes called a multipoint control unit, or MCU, to mix audio data for the conference. We introduce a novel mixer, called a Virtual Mixer, which performs mixing in a distributed way over the network. The Virtual Mixer topology is optimized over Steiner trees using a metric of either average pairwise delay (APD) or maximum pairwise delay (MPD). Since the topology is adapted to the particular set of clients and servers available in the cloud, optimization speed is important. In order to solve this NP-hard Steiner tree optimization, we propose heuristic algorithms for finding the Pairwise-delay-optimal Tree (PT) for both APD and MPD, which are orders of magnitude faster than exhaustive search, yet find trees with delays that are minimal or within a few percent of minimal. We show through experiments both on a corporate intranet and on up to 12 PlanetLab nodes that Virtual Mixing can reduce both the APD and the MPD between clients by upwards of 50%, compared with the existing MCU-based and P2P-based mixing approaches.

Keywords—Audio Mixing, Video Mixing, Multiparty Conferencing, Telepresence

1. INTRODUCTION

The rapid development of the Internet has enabled more and more forms of interactive multimedia applications, among them audio and video teleconferencing across multiple sites. A fundamental feature of such multiparty conferencing is the need for each endpoint in a conference to present to its user a mixture of the audio from the other endpoints in the conference.

In pure peer-to-peer (P2P) systems, the endpoints communicate directly with each other, and therefore mixing is done at the endpoints. In an N-way conference, each of the N endpoints encodes and sends its audio stream to the $N-1$ other endpoints, receives and decodes $N-1$ audio streams from the other endpoints, and mixes the decoded audio streams for rendering. As N grows without any particular bound, the burden on every endpoint, in terms of decoder computation as well as outbound and inbound bandwidth, increases linearly for quadratic use of resources in the overall system.

Mixing services are therefore frequently used in multiparty conferencing. A mixer is a service that receives and decodes streams from a collection of endpoints, creates a unique mixture for each endpoint by leaving the endpoint's own stream out of the mixture, and encodes and sends the corresponding mixture back to the endpoint. This relieves the endpoint of the computational and bandwidth burden of doing the mixing itself. Each endpoint needs to encode and send only its own stream, and to receive and decode only the mixed stream. Though we have described this here for audio, it is also possible to have video mixers that, for example, combine multiple H.264 video streams into a single H.264 video stream, for decoding by a single H.264 decoder.

A mixer is typically hosted on a server in a dedicated content delivery infrastructure. A server that hosts a mixer is known as a Multipoint Control Unit, or MCU, in telecommunications terminology. Classically, in the telephone network, all of the endpoints (or "terminals") participating in a multipoint conference are handled by a single MCU. This single-MCU structure is also commonly employed by multipoint conferencing systems operating over the Internet [2]. Although there is only a single MCU per conference, it is not uncommon for a system to offer multiple geographically distributed MCUs from which to choose.

A mixer can also be hosted on one of the better endowed endpoints in a conference, as in Skype [4]. In this case, the mixer mixes the data from its own endpoint into each outgoing stream. Placing the mixers at end hosts is convenient in P2P systems. However, in many P2P conferences, there may be no particularly well-endowed peer to perform the mixing. A mixing scheme that rotates the mixing responsibility evenly through the peers at a packet level, known as MutualCast, is proposed in [3].

In this paper, we show that having not a single mixer, but a network of mixers, which acts as a single Virtual Mixer as shown in Figure 1, can significantly reduce the average delay or the maximum delay between endpoints in a conference. The reduction in delay is particularly striking when there are multiple clusters of endpoints engaged in a conference. This is sometimes known as the "branch office" scenario, since participants spread across two or more branch offices will frequently be clustered into geographically separated groups. But the scenario also occurs, for example, in multiparty conferences over the Internet spanning multiple geographic regions. In this paper, we demonstrate both in a branch office example over a corporate intranet and in a cross-region example over the Internet using PlanetLab that upwards of 50% reduction in both average and maximum delay can be achieved.

To achieve the maximum reduction in delay, the topology of the network of mixers in the Virtual Mixer must be optimized for each multiparty conferencing session. We define a class of topologies based on Steiner trees, which include all the endpoints (or clients) in the multiparty conferencing session. The Steiner trees may also include any number of optional helper MCUs (or servers) made available for mixing. Any node in the interior of the Steiner tree, whether it is a server or a client node, becomes a mixer. Hence our approach is a hybrid of the infrastructure-based and P2P-based mixing approaches in previous works. We optimize over Steiner trees using as a metric either average pairwise delay (APD) or maximum pairwise delay (MPD) between clients. Since finding the minimum weighted Steiner tree is an NP-hard problem, we propose a heuristic algorithm called the Pairwise-delay-optimal Tree (PT) algorithm. We compare PT's APD and MPD to the minimum APD and MPD as determined by exhaustive search in networks containing up to 12 nodes. For these networks, the trees found by PT have delays that are minimal or within a few percent of minimal.

Because we are able to search quickly over trees containing any number of optional servers, it is possible for a multiparty conferencing system to offer a large number of helper MCUs in many possible locations around the world. Helper MCUs may be deployed a priori in global data centers, in content distribution networks, on university or corporate networks, in buildings or even in rooms that support multiparty conferencing. When combined with the client nodes in a particular session, these servers form a highly effective content delivery network. However, servers are only included if they are part of the minimum delay topology. Thus, our Virtual Mixer suits situations where users are clustered in a local region as well as where users are in multiple clusters across regions. Our algorithm optimizes the distribution topology according to the locations of users in each session. From the clients' perspective, as Figure 1 shows, there is only one "virtual" mixer available to them, and they can easily access the data streams given to them from the virtual mixer with the least cross-region delay.

Our work is related to the MixNStream work of Yuen and Chan [1], in which an overlay network of servers is constructed and continually updated to support video distribution with a small worst-case delay. In other works on mixing, [5] presents a mixing algorithm that minimizes the difference between generation times of the media packets in the absence of globally synchronized clocks and in the presence of jitter in transmission delays, [6] investigates different types of mixing algorithms and proposes objective methods for evaluating their performance, and [7] proposes a simpler MCU using algorithms for echo cancelation.

2. VIRTUAL MIXER

In this section we first introduce the graph model of the network and the tree model of the Virtual Mixer. Then we define the APD and MPD delay metrics. Finally, we show how the metrics are optimized over the trees, and how trees are dynamically chosen.

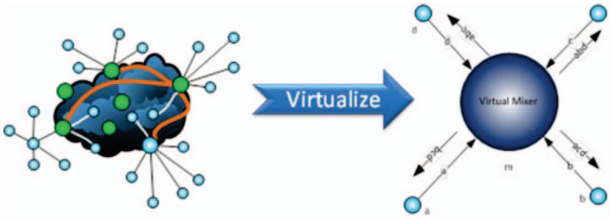

Figure 1: Virtualize mixing resources into one virtual mixer to conference parties.

2.1 Graph Model

We define an undirected graph $G =< V, E >$ to model the network of clients and all possible servers in the Virtual Mixer. Let each node in V be either a client node c labeled $1, \dots, C \in V_C$ or a server node s labeled $C+1, \dots, C+S \in V_S$. We have $V_C \cup V_S = V$. We assume the graph is fully connected, since each node can reach any other node through a direct network connection without going through a third node.

Not all server nodes need to be part of the Virtual Mixer. For a graph G, we define subgraphs G_k, $k \in K = \{1, 2, \dots, 2^S\}$, where each subgraph G_k includes all C clients plus the kth unique subset of the S servers.

2.2 Mixing Model

For each subgraph G_k, a spanning tree $T =< V_T, E_T >$ on G_k is a tree connecting all vertices of G_k. V_T has two types of nodes: clients in a set V_{T_C} and servers in a set V_{T_S}, so $V_T = V_{T_C} \cup V_{T_S}$.

Each spanning tree T defines a Virtual Mixer as follows. If a node in T is an interior node (i.e., not a leaf), it acts as an MCU, as follows. When the interior node is a server with a set of neighbors N, it sends to each neighbor $n \in N$ a mixture of the streams from all other neighbors $N - \{n\}$. When the interior node is a client, it sends to each neighbor $n \in N$ a mixture of the stream originating from itself and from all other neighbors $N - \{n\}$. When a client is a leaf, it simply sends its own stream to its neighbor. When a server is a leaf, it is discarded as it performs no function. No client or server forwards more than one copy of the same data to different nodes.

2.3 Delay Metrics

Every spanning tree T, as a Virtual Mixer, can be evaluated by a set of metrics. For any pair of client nodes $u, v \in V_{T_C}$, let the pairwise delay $D_{pw}(uv)$ be the sum of the delays along the unique path in the spanning tree from u to v.

We define the average pairwise delay (APD) of a spanning tree T to be the average of the pairwise delays between clients u to v,

$$APD_T = \frac{1}{n} \sum_{\substack{u,v \in V_{T_C} \\ u \neq v}} D_{pw}(uv), \qquad (1)$$

where $n = C(C-1)$. Similarly, we define the maximum pairwise delay (MPD) of a tree T to be the maximum of all pairwise delays between clients u to v,

Table 1. PT algorithm to find the minimum APD/MPD

PT Algorithm

Input: A non-empty fully connected weighted graph $G_{PT} = < V_{PT}, E_{PT} >$, a server set S

Initialize:

$V_{new} = \{x\}$, where x is an arbitrary node (starting point) from V_{PT}, $E_{new} = \{\}$

Repeat until $V_{new} = V_{PT}$

Choose an edge (u, v) with minimal APD/MPD increase such that u is in V_{new} and v is not (if there are multiple edges with the same APD/MPD increase, any of them may be selected)

Add v to V_{new}, and (u, v) to E_{new}

For all v in V_{new}

If degree of v is one and v is in server set S, remove v from V_{new} and remove the edge connecting the leaf from E_{new}

$$MPD_T = \max_{\substack{u,v \in V_{T_C} \\ u \neq v}} D_{pw}(uv). \quad (2)$$

Among all the spanning trees T in G_k, let T_k be a spanning tree that has the minimum APD (or MPD). Then, among all trees T_k in graph G, let T_{min} be a tree that has the minimum APD (or MPD). We call T_{min} a pairwise delay minimum Steiner tree in graph G, with respect to APD (or MPD).

2.4 Pairwise-delay-optimal Tree

Note that T_k is *not* in general a minimum weight spanning tree in G_k, where the weight of a tree is defined as the sum of the delays (or other quantities) on the edges of the tree. Thus finding T_k cannot be done with any of the efficient minimum spanning tree (MST) algorithms, which rely on the modularity of the tree weight. Unfortunately the APD and MPD metrics, which are more relevant than tree weight to end users, do not admit efficient minimization algorithms. Here we propose a simple heuristic algorithm to find the tree with minimum APD or minimum MPD, called the Pairwise-delay-optimal Tree (PT) algorithm. The PT algorithm actually is an algorithm suite that has two versions, PT-APD and PT-MPD, depending on the metric (APD or MPD) adopted, but their procedures are similar. Here we explain how the PT algorithm finds the minimum APD or MPD spanning tree. The output of the PT algorithm is a spanning tree, i.e. PT. The PT algorithm works as follows. To initialize, it builds an empty set T with zero edges. For a graph G with n nodes, a PT has $n-1$ edges. So the algorithm iteratively adds a new edge to the set, by selecting the edge that yields a partial tree T with the minimum APD or MPD. That is, it selects the edge that causes the least APD/MPD increase and adds it to the set T.

The algorithm is shown in Table 1. Note that if a Server becomes a leaf, indicating that the server is far away from any client node in the graph, we remove the server from the tree. We run the PT algorithm on all 2^S possible configurations of the S servers to find T_{min}, the pairwise delay minimum Steiner tree with respect to APD (or MPD). This is tractable since the number of servers S available to perform mixing for an audio conference is typically small.

2.5 Dynamic Group Forming

When client nodes join and leave, the tree is updated towards a smaller APD/MPD.

3. EXPERIMENTS

3.1 Experimental Setup

To evaluate the effectiveness of our solutions, we performed two sets of experiments. The first set is performed over a corporate intranet in two cities across continents, and the second is over the Internet on PlanetLab. In both setups, we compare our PT-APD and PT-MPD algorithms with the single mixer approach, the MutualCast approach, and the Steiner tree with minimum APD or minimum MPD obtained by exhaustive search. Since exhaustive search becomes intractable as the problem scales up, we first present how we conduct exhaust search and pruning to reduce the search cost as much as possible.

3.1.1 Exhaustive search and tree pruning

First we calculate the number of all possible spanning trees on a graph. Cayley's formula states that the number of spanning trees in a complete graph with n labeled vertices is n^{n-2}. Indeed, there is a 1-1 mapping between the set of spanning trees and sequences over n letters of length $n-2$, known as Prüfer sequences (or Prüfer codes). Each Prüfer sequence can be converted into a spanning tree. So a brute force approach is to enumerate all the Prüfer sequences, calculate the APD/MPD of the corresponding spanning tree, and find the one that has the minimum APD/MPD.

The complexity of enumerating all Prüfer sequences is clearly exponential. Therefore finding an optimal solution through straightforward enumeration becomes intractable as n increases. We adopt a branch-and-bound approach in traversing the solution space.

The exhaustive search of all possible solutions is done in a depth-first-search way. For a given Prüfer sequence, a corresponding spanning tree is generated node by node. In each step, when a node is added to the tree, we evaluate its APD or MPD. If it exceeds a pruning bound, we stop the generating process and go to the next Prüfer sequence. The bound is dynamically updated towards optimal during the enumeration.

3.2 Experimental Results

3.2.1 Corporate intranet experiment

In this experiment, we have six machines: three in Asia and three in North America. We compare the single mixer approach, the MutualCast approach, the Virtual Mixer (PT-MPD and PT-APD) we proposed, and exhaustive search. Results are listed in Table 2. All delays in the table in this paper are measured as Round Trip Delay. Figure 2 shows the topology of the Virtual Mixer solution.

In Table 2, each row represents the topology generated by different approaches. The last two rows are optimal topologies obtained by enumerating all possible configurations of spanning trees and finding the tree with

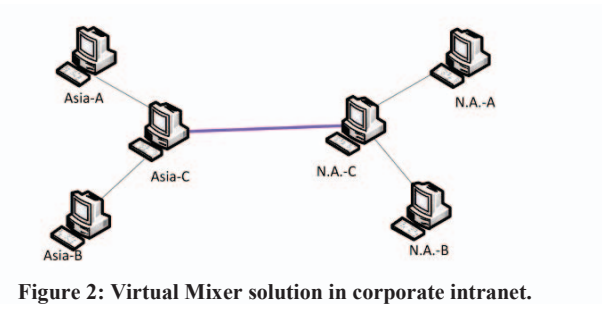

Figure 2: Virtual Mixer solution in corporate intranet.

Table 2. Comparing mixing approaches in corporate intranet.

Metric (millisecond)	Maximum Pairwise Delay (MPD)	Average Pairwise Delay (APD)	Running Time
Single Mixing	444.021	222.043	N/A
MutualCast	443.896	166.510	N/A
Virtual Mixer (PT-MPD)	**222.109**	133.272	24.23
Virtual Mixer (PT-APD)	222.109	**133.272**	21.07
Minimum MPD Tree	**222.109**	133.272	1877.63
Minimum APD Tree	222.109	**133.272**	1889.48

the minimum APD or MPD. In each column is the performance of the various approaches under the two metrics, APD and MPD. We also log the running time of PT algorithm and exhaustive search in the last column.

From Table 2 we can see that, in the Virtual Mixer scheme, both the APD and MPD have been reduced almost by a factor of two compared with the other schemes. Since the six-node case has a small enumeration space, it does not take much time to exhaust all the possibilities. In this case, the MPD and APD metric achieve their optima with the same tree, but we will show in the next experiment that they are not always so coherent.

3.2.2 PlanetLab experiment

In this experiment, we have twelve nodes spread across Hong Kong, Taiwan, United States and Canada. We run the same test as in Section 3.1.1, with the results shown in Table 3. Figure 3 shows the topology of the Virtual Mixer solution determined by PT-MPD.

In this case, there is no single tree that minimizes both APD and MPD. However, the minimum APD tree has an MPD only 4% larger than that of the minimum MPD tree. The former is the tree found by PT-APD, in almost four orders of magnitude less time.

4. CONCLUSIONS

In this paper we propose a novel Virtual Mixer that can achieve substantial latency reduction among parties in a conference. In order to achieve the maximum reduction in latency, the proposed Virtual Mixer is optimized over Steiner trees using a metric of either average pairwise delay (APD) or maximum pairwise delay (MPD). In lieu of an NP-hard Steiner tree optimization, we propose the greedy Pairwise-delay-optimal Tree (PT) algorithm, which uses

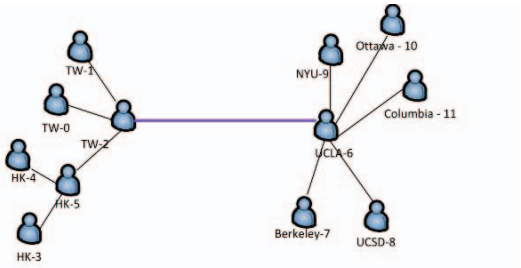

Figure 3: Virtual Mixer solution on PlanetLab.

Table 3. Comparing mixing approaches on PlanetLab.

Metric (millisecond)	Maximum Pairwise Delay (MPD)	Average Pairwise Delay (APD)	Running Time
Single Mixing	471.525	207.406	N/A
MutualCast	523.766	240.033	N/A
Virtual Mixer (PT-MPD)	**262.461**	138.089	797.75
Virtual Mixer (PT-APD)	273.268	**133.387**	783.60
Minimum MPD Tree	**262.461**	135.747	4972683.09
Minimum APD Tree	273.268	**131.046**	4894240.12

either MPD or APD as the objective, respectively called PT-MPD and PT-APD. We compare the performance of PT-MPD and PT-APD to the minimum MPD and minimum APD as determined by exhaustive search both in a corporate intranet and in a Planet Lab network with up to 12 nodes. The experimental results show that the Virtual Mixer using the trees generated by our proposed algorithms can achieve delays that are minimal or within a few percent of minimal. This is about half of the delay of the traditional approaches.

5. REFERENCES

[1] Yuen, P. and Chan, G. MixNStream: multi-source video distribution with stream mixers.In *Proceedings of the 2010 ACM workshop on Advanced video streaming techniques for peer-to-peer networks and social networking*, pages 77-82, ACM, 2010.

[2] TANDBERG MCU and IP. *http://www.tandberg.com/collateral/white_papers/whitepaper_TANDBERG_MCU_and_IP.pdf.*

[3] Li, J. MutualCast: A Serverless Peer-to-Peer Multiparty Real-Time Audio Conferencing System., In *Multimedia and Expo, IEEE International Conference on*, pages 602-605 2005.

[4] Ahson, S. and Ilyas, M. VoIP handbook: applications, technologies, reliability, and security, pages 162-163 CRC, 2008.

[5] Rangan, P. V., Vin, H. M. and Ramanathan, S. Communication architectures and algorithms for media mixing in multimedia conferences. *IEEE/ACM Transactions on Networking (TON)*, 1, 1 1993), pages 20-30.

[6] Chandra, S. P., Senthil, K. M. and Bala, M. P. P. Audio mixer for multi-party conferencing in VoIP., In *Internet Multimedia Services Architecture and Applications (IMSAA)*, pages 1-6. 2009.

[7] Junlin, L., Li-wei, H. and Florencio, D. Multi-Party Audio Conferencing Based on a Simpler MCU and Client-Side ECHO Cancellation. In *Multimedia and Expo, 2007 IEEE International Conference on*, pages 84-87. 2007.