On Reducing Mesh Delay for Peer-to-Peer Live Streaming

Dongni Ren *Y.-T. Hillman Li* *S.-H. Gary Chan*
Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong
Email: {tonyren, hillmanl, gchan} @cse.ust.hk

***Abstract*—Peer-to-peer (P2P) technology has emerged as a promising scalable solution for live streaming to large group. In this paper, we address the design of overlay which achieves low source-to-peer delay, is robust to user churn, accommodates of asymmetric and diverse uplink bandwidth, and continuously improves based on existing user pool. A natural choice is the use of mesh, where each peer is served by multiple parents. Since the peer delay in a mesh depends on its longest path through its parents, we study how to optimize such delay while meeting a certain streaming rate requirement.**

We first formulate the minimum delay mesh problem and show that it is NP-hard. Then we propose a centralized heuristic based on complete knowledge which serves as our benchmark and optimal solution for all the other schemes under comparison. Our heuristic makes use of the concept of *power* in network given by the ratio of throughput and delay. By maximizing the network power, our heuristic achieves very low delay. We then propose a simple distributed algorithm where peers select their parents based on the power concept. The algorithm makes continuous improvement on delay until some minimum delay is reached. Simulation results show that our distributed protocol performs close to the centralized one, and substantially outperforms traditional and state-of-the-art approaches.

I. Introduction

In order to provide live streaming services (such as IPTV) to a group of users in the absence of IP multicast support, traditionally client-server model is used where servers are used to serve individual participants directly [1]. This model is not scalable to large group. Peer-to-peer (P2P) live streaming has been recently proposed to overcome the scalability problem. In P2P streaming, a server only needs to stream to some users, who in turn share their stream received with their neighbors. Because the stream is distributed using users' uplink bandwidth, the bandwidth requirement at the server can be drastically reduced. Such P2P system has shown to be effective in serving quite a large group[2], [3].

In P2P live streaming, peers join an overlay in a distributed manner. In this research, our goals are to design an overlay which achieves the following:

- *Low delay:* Live streaming applications are very sensitive to delay. Therefore, an overlay offers low source-to-peer delay is desirable. We would like to design such overlay which minimizes the maximum delay of the peers.
- *Robust to user churn:* Peer traffic in the network can be highly dynamic. A peer may join, leave or fail at anytime. The overlay structure has to accommodate this network dynamic and be robust to user churn.
- *Accommodation of asymmetric bandwidths:* Peers in the network may have diverse uplink bandwidth depending on their access network (such as ADSL, broadband Ethernet, Wireless LAN, cable, etc.). The overlay should meet streaming rate requirement for each peer despite of this bandwidth heterogeneity or asymmetry. For fairness and incentive purposes, those peers who contribute more to serve others (i.e., those with higher uplink bandwidth) should enjoy relatively lower delay [4].
- *Distributed, simple and adaptive:* The protocol should be distributed, and its performance should be scalable to large number of users. The protocol should be adaptive in the sense that it continuously improves the overlay based on the existing heterogeneous user characteristics. It should also be simple so that it can be implementable.

It has been proposed to use a tree structure to distribute streams among peers, where all peers are arranged into a tree rooted at the source [5]. The media is streamed down from the source to every peer along the tree edges in a push-based manner. Though the tree approach is simple and achieves low delay, the failure of a node can seriously affect the streaming quality of all its descendants due to tree re-construction. Furthermore, the streaming rate cannot be guaranteed as it is limited by the least uplink bandwidth of a node in the tree. Therefore, tree cannot accommodate well network dynamics and asymmetric bandwidth.

To address the above problems, streaming mesh [6] has been proposed. In mesh, each peer maintains a list of neighbors that it exchanges information with. A peer obtains its stream by aggregating the flows from many parents using either pull-based or push-based methods. Figure 1 shows an overlay example of push-based streaming using mesh. S is the streaming source and A, B, C, D, E are five peers in the streaming session. The number by the arrows is the overlay delay (in units). S streams media to peers A and B, who in turn stream to peers C, D, and E. Due to insufficient uplink bandwidth between A and D, C streams part of the stream to D. This is the same for E, which has two parents. Clearly, because

This work was supported, in part, by Direct Allocation Grant at the HKUST (DAG05/06.EG10), Hong Kong Research Grant Council (611107), and the Hong Kong Innovation and Technology Commission (GHP/045/05).

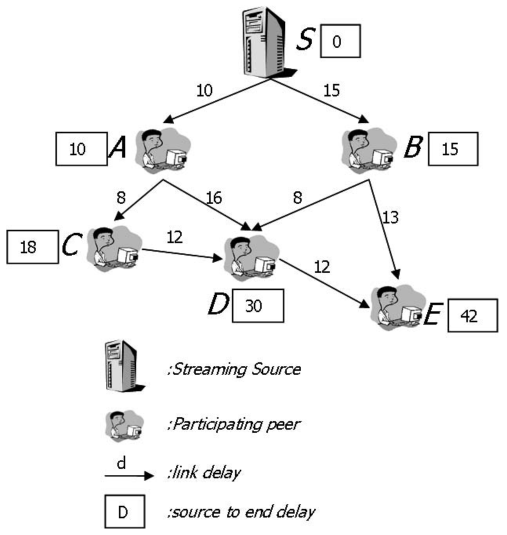


Fig. 1. Streaming with mesh with node delay.

of multiple parents, mesh is more robust than tree to user churn. The asymmetric bandwidth problem is also overcomed by aggregating the bandwidth of multiple parents to guarantee a certain streaming rate.

However, the robustness and bandwidth guarantee of mesh comes with the cost of delay. Refer to Figure 1 again where we show how overlay delay is accumulated by the square boxes. Nodes A and B have a delay of 10 and 15, respectively, given by their overlay paths from S. Because of the delay of A, C suffers a delay of 18. Because D receives its stream from two parents A and C, its delay is the maximum of the two, i.e., $\max(10+16, 18+12) = 30$ (units). E gets its stream from B and D, and hence its delay is obtained similarly as $\max(15+13, 30+12) = 42$. As we can see, the delay accumulates quite quickly as the number of peers increases.

In streaming using mesh, the total delay compse of two componets: 1)*Mesh delay*, due to the longest path from the node to the source out of all its parents. This is the number indicated in the square boxes in Figure 1; and 2) *Packet scheduling delay*, due to packet transmission time and scheduling policy of a peer with its parents of heterogeneous bandwidth. Designing a mesh *jointly* optimizing these delay componets is very complex. Therefore much work has been focus on only reducing packet scheduling delay. In this work we focus on minimizing mesh delay of a node, as in a large network, it is the major delay componet in the overall source-to-node delay if not treated properly. This is because when the user population grows large, streams have to go through many hops and the mesh delay accumulates fast. The motivation is to begin with a mesh with low delay. With the use of any existing good scheduling algorithm on the mesh, we achieve an overall low delay for live streaming.

To the best of our knowledge, this represents the first body of work addressing the optimization of mesh delay for P2P streaming. We address the problem from the following three directions:

1) *Problem formulation and a centralized heuristic*
We first formulate the minimum-delay mesh problem, which is to form a mesh which minimizes the maximum delay of the peers in the network while meeting a certain streaming rate requirement. We show the problem is NP-hard, and hence propose a centralized heuristic based on complete knowledge. This serves as a benchmark for our later comparison study.
Given the end-to-end delay of the peers, our centralized heuristic makes use of the traditional *power* concept in network design given by the ratio of throughput with delay. The centralized heuristic achieves low delay by first ranking the nodes according to their uplink bandwidth followed by "maximizing" the power in the mesh.
2) *A distributed protocol for low-delay mesh*
Given the good performance of our centralized heuristic, we propose a novel distributed protocol to build a low-delay mesh with random user joins and leaves. A new arrival first selects some good set of parents based on the power concept (as in the centralized heuristic). The mesh continuously improves itself with the peers trying to locate and connect to better parents to further reduce their delay. This adaptation mechanism move the nodes to appropriate positions in the mesh.
3) *Performance study on the algorithms*
We conduct simulation study on our centralized and distributed algorithms, and compare them with traditional and state-of-the-art approaches (closest-parents and Outreach). Our results show that our algorithms achieves substantially lower delay by more than 50%, with lower server workload and better fairness.

The rest of this paper is organized as follows. We first review related works in Section II. Then we present the formulation of the minimum delay mesh problem and the centralized heuristic in Section III. In Section IV, we discuss the distributed protocol to build a low-delay mesh. Illustrative simulation results and comparison are presented in Section V. We conclude in Section VI.

II. Related Work

In this section, we briefly review previous work. Because a peer in a mesh is served by many parents, a packet scheduling mechanism is needed to schedule when parents should send which packet. There has been much study on packet scheduling algorithms to reduce packet reassembly delay or to improve throughput (see, for examples, [7], [8], [9], [10] and references therein). Our work is orthogonal to them, and the mesh built may apply any of the scheduling algorithms to achieve low-delay streaming.

There has been much work on how to construct trees for overlay streaming. Centralized algorithms such as CoopNet, ALMI build a tree rooted at the streaming source [11], [12].

Narada first constructs a overlay mesh, and then spanning trees with multiple sources are generated on top of the mesh for data delievery[13]. NICE and ZIGZAG are distributed protocols which arrange the participating peers into clusters and layers in a distributed manner to minimize delay and workload [14], [15]. However the structures cannot be easily maintained. In all these works, the streaming rate and robustness to network dynamic cannot be easily guaranteed. We study mesh here as it addresses the streaming and robustness issues in peer-to-peer streaming.

There has been work on using mesh for P2P streaming. However, most of them do not optimize the mesh by randomly connecting the participating peers to neighbors for data exchange. Based on gossip, the peers gradually connect to some closest parents for data exchange (the *Closest Parents* approach in our simulation). Such approach leads to high delay from the source to end hosts. Chainsaw is built based on request-response data dissemination and gossip protocol [16]. Peers request fresh data from neighbors in a BitTorrent-like manner. Coolstsreaming is another work motivated by the gossip concept, where peers pull data from multiple partners using a scheduling algorithm to reduce packet redundancy [17]. There are also other streaming protocols that adopt the mesh structure, such as Bullet and GridMedia [18], [19]. As compared to the above, our protocol achieves much lower delay by optimizing the mesh structure through parent selection and adaptations.

Asymmetric bandwidth problem has been studied in Outreach [20]. Outreach targets to minimize the source workload by making full use of the peers' uplink capacity. Every peer in the network estimates a upstream and downstream bandwidth difference of the whole network based on its knowledge. Every new peer will connect to the peers with largest bandwdith difference. Though Outreach is on the right track, the performance is not far from the random schemes. Because the newcomer randomly asks one of the source children to perform the estimation, chances are we will miss the good estimation. More importantly, Outreach has not proposed any continuous adaptation to improve the streaming mesh; nor does it place powerful peers in strategic locations, e.g. at positions where they enjoy low delay.

III. Problem Formulation and Centralized Heuristics

We present the formulation of the minimum delay mesh problem in section III-A. Given the complete knowledge of the overlay network, we present a centralized heuristic to solve the problem in section III-B.

A. Problem Formulation

We model the overlay network as a complete directed graph $G = (V, E)$, where V is the set of vertices representing the participating peers and $E = V \times V$ is the set of overlay edges. For any edge $\langle i, j\rangle$ in G, the cost of the edge is the underlay unicast path delay from node i to node j in the physical network. We let d_{ij} represent the delay of $\langle i, j\rangle$. Let T_i be the set of parents of node i, we denote D_i as the delay of node i along the overlay paths, which is the maximum of the delays of node i's parents plus the connection delay between i and its parents, i.e.,

$$D_i = \max_{j:j\in T_i}(D_j + d_{ji}). \tag{1}$$

To simplify notations, we further define $D_i(j) = D_j + d_{ji}$, which is the delay of node i through its parent j. Therefore D_i can be written as $D_i = \max_{j:j\in T_i} D_i(j)$.

There is a single source node S in V that streams data to all nodes at a rate of s units/second, which we call the streaming rate. For every node i in V, it has an uplink bandwidth of U_i unit/second. Note that U_i may be differnt for diffrent peers and may be lower than s. We define $s_i(j)$ as the rate that parent j streams to node i. For any node i in G, if it gets an aggregate incoming stream of s unit/second from its parents, i.e., $\sum_{\forall j\in T_i} s_i(j) = s$, we call node i *fully served*. A fully served node can play back the video smoothly and is able to stream to its children nodes. We define R_i to be the residual uplink bandwidth of node i. It means the remaining uplink bandwidth after it serves all of its children in the mesh. We consider network core has enough capacity, and hence the bottleneck lies on the network edge.

Minimum Delay Mesh Problem (MDM problem): The MDM problem is to find a mesh which minimizes the maximum of the peer delay, i.e., $\min\max_{i\in V} D_i$ subject to the streaming requirement, i.e., $\sum_{j\in T_i} s_i(j) = s$.

We assume a streaming mesh exists, which requires the total uplink bandwidth is larger than the total downstream bandwidth, i.e.,

$$\sum_{i=0}^{|V|} U_i \geq (|V| - 1) \times s. \tag{2}$$

The MDM problem is NP-Hard.

Proof: First we show the problem is in NP. By running breadth-first search on the mesh, we can go through all the nodes, checking the streaming rate constraint and calcuating their delay in polynomial time. We then verify whether the mesh has the mimimum delay by comparing its maximum delay with the given delay constraint. Next we show the problem is NP-hard. Traveling Salesman Problem(TSP) can be reduced to MDM problem [21][22]. Given a complete graph $G = (V, E)$, a TSP formulation is to find a path which starts at the source S, visits all the nodes in V exactly once and the total cost D is minimized. Let's have a MDM formulation on the graph $G' = (V', E')$, which is identical to G. For all the node in G' including the source, we let their uplink bandwdiths be the same as the streaming rate s, i.e., $U_i = s, \forall i \in V'$. In another word, every node will have just enough bandwidth to serve one other node and in this case every node will have exactly one parent and one child in the streaming mesh. The resulting mesh becomes a chain and the maximum delay of the mesh, i.e. $\max_{i\in V} D_i$, will be the delay of the last peer in the chain. Therefore the maximum delay of the mesh in G'

will be the same as the total cost in G. The TSP in G will have a path with minimum cost of D if and only if the same path in G' is the minimum delay mesh and the min-max delay of MDM is D. We just showed TSP can be reduced to our problem. Therefore the MDM problem is also NP-hard.

B. A Centralized Heuristic

Given the NP-hard nature of our problem, in this section we propose a centralized heuristic based on complete knowledge (i.e. knowledge of the user pool at the beginning and pairwise distance between them). The algorithm serves as a benchmark for the evaluation of our proposed distributed protocol.

Clearly, to achieve low delay, a good heuristic should construct a "shallow" streaming mesh where peers are close to the source with low hop count. Therefore we should try to put the nodes with high uplink bandwidth close to the source to increase the fanout of the mesh towards the uplink. Since nodes of heterogenous uplink bandwidths are randomly distributed in the network, we make use of the concept of *"power"* to achieve a balance between the delay and uplink bandwidth. Traditionally in networking, *power* is defined as the throuhput divided by delay. In this paper the concept of *"power"* is a little bit different. We Let $P_i(j)$ be the *power* between a peer i and its parent j, defined as the rate that node j is serving node i divided by the delay of i via parent j, i.e.,

$$P_i(j) = \frac{\min(R(j), s)}{D_i(j)}. \tag{3}$$

Our algorithm runs as follows. First we rank all the nodes according to their uplink capacities divided by their delay to the source, i.e., $\frac{U_i}{d_{si}}$. After that we push them into the mesh in descending order. It is because we want to put the nodes with large bandwidth and small delay as close to the streaming source as possible so that they can have a low source to end delay. In this way their children can in turn achieve a low source to end delay. When a node i is pushed into the mesh, we calculate the power $P_i(j)$ for all the nodes already in the mesh and connect node i to node j with the largest $P_i(j)$ value. If node i is not fully served by node j, we will connect to node i to one more parent with the second largest $P_i(j)$ value. We kept connecting node i to parents in the mesh according to $P_i(j)$ until it is fully served, which means the total uplink bandwidths it consume from all of its parents equal to the streaming rate s. And then we begin to push the next node into the mesh. The algorithm ends when all the nodes are pushed into the mesh.

```
Input:
    $G = (V, E)$
    Source node: $S$
    edge delay: $d_{ij}$ for $i, j \in V$
    streaming rate: $s$
Output:
    M with the minimum source to end delay
Algorithm:

1: for each $i \in V$
2:     $s_i$ = 0
3:     $R_i$ = $U_i$
4: push $S$ into $M$
5: sort all $i$ in $V$ according to $U_i$ in descending order
6: for each $i \in V$
7:     for each $j \in M$
8:         Calculate $P_i(j)$
9:     sort j according to $P_i(j)$ in descending order
10:    $j = 0$
11:    while $s_i < s$
12:        if $R_j \geq s - s_i$
13:            $R_j = R_j - (s - s_i)$
14:            $s_i = s$
15:            push $i$ into $M$
16:        else
17:            $s_i = s_i - R_j$
18:            $R_j = 0$
19:        $j = j + 1$
20: return $M$
```

Fig. 2. Pseudocode for the centralized algorithm.

IV. Power-Based Distributed Algorithm

Given the complete knowledge of the network topology and user pool, the centralized algorithm works well to minimize delays. However in practice, we do not have such global information. A joining peer does not know all the other current users. Besides we cannot afford to have central planners to keep tracks of thousands of peers with dynamic internet condition. Our centralized heuristic serves as an benchmark for comparison. We also propose a distributed protocol, which is scalable and follows the same guiding principles as the centralized algorithm.

In this section, we will first introduce our parent selection algorithm for new joining peers. By selecting the right parents to uplink to them, newcomers are able to have a low source to end delay and good streaming quality. We will then discuss about the mesh adaptation mechanism, which is used to further adapt the existing mesh to achieve a more optimal mesh than the current one. At the end we will explain the operations for a node leave request.

A. Node Join and Parents Selection

In order to receive the streaming content, a newly arrived peer has to look for parents with good quality. Ideally the parents should timely deliver the contents so that the delay perceived by the child is as small as possible. Yet the parents should have sufficient residual uplink capacity. Small latency does not guarantee sufficient bandwidth. To this end, we make use of the power defined previously.

Upon the arrival of a new peer i, it contacts a *Rendezvous Point* which caches a list of recently arrived peers. The Rendezovus Point returns a few of them to the newcomer. These peers are the potential parents. Peer i checks its delay with respect to each of potential parents j. Next, peer i

requests the residual bandwidth of j and evaluates its power by Equation(3). As mentioned before, power is in fact the amount of data sent from the peer in unit time. By choosing the parents with large power, it prevents the newcomer from connecting to biased parents, such as parents with large bandwidth but high delay or parents with low delay but insufficient bandwidth.

After computing the powers, the newcomer selects parents in greedy manner. More precisely, peer i first connects to the parent with largest power. If this parent does not have enough residual bandwidth to fully serve it, peer i then connects to second most powerful parent. This process repeats until it is fully served.

If the peers returned by the *Rendezvous Point* cannot fully serve the newcomer, the newcomer request the neighbor of those peers. Then the above process is applied again to these newly retrieved nodes.

B. Adaptation

In the distributed environment, there is no specific joining order of peers. The powerful peers may come late and as a result are far from the source. As recent studies have suggested, most participants in live streaming peer-to-peer systems have low and even zero uplink bandwidth[23]. With high probability, there are some low-bandwidth peers occupying the areas in between the source and the powerful ones. The intuition behind our adaptation algorithm is to figure out the proper positions where we should promote the high-bandwidth peers to.

The adaptation consists of three steps:

1) Request Step: A child initiates the process by inspecting its residual bandwidth. If this is greater than the streaming rate, the child sends its parents the REQUEST message which contains its uplink bandwidth and a time-to-live field (TTL).

2) Grant Step: There are two things for a parent to process REQUEST messages which have come from different successors. First, if the TTL field is greater than zero, the parent decrements it and forward the REQUEST message to its own parents. Next, the parent checks whether its uplink bandwidth is greater than that of the sender. If this is the case, the parent makes an response to the sender with an GRANT message which contains its distance from the source.

3) Accept Step: The successor may receive a number of GRANT messages. Among these the ancestor with shortest distance from the source is picked. More specifically, the successor replaces all its existing parents with that ancestor's parents. After that, the ancestor disconnects from the existing parents and then takes the successor as its parent. In this process, the successor should also check whether the new parents have higher power than the existing ones before making the change.

We would like to discuss more about the distance mentioned in the Grant step. The distance of a peer from the source can be its delay time, that is the time taken by packets to arrive at the peer after the source has streamed out. Alternatively, we can measure the distance as the number of intermediate peers involved in the streaming path. The invariant is that the ancestors must have a smaller distance than the successors so that in the Accept step we are moving the successor closer to the source. As long as this invariant is maintained, any other measure can serve the purpose of distance. But some peers have multiple parents, data will be streamed from different paths. In this case, the distance of the peers is the longest path.

The key parameter in the adaptation protocol is the TTL value which means how far away the REQUEST messages should travel from the peers. On one hand, if the REQUEST messages can travel farther away, more ancestors will send out GRANT messages thus the successors can potentially obtain higher positions; on the other, too many REQUEST messages result in flooding the streaming network, wasting network resources for data transmission. So it may seem rational to set the TTL field to small values, in the extreme case, it will be 1. But as we will show in simulation results, this encourages unnecessary adaptation changes as each time the successors only move up by one step at most. In the real world environment, frequent tearing down old connections and establishing new ones will affect the mesh stability and so it may not be able to smoothly stream out the movie. In the simulation section followed, we will try with different TTL to figure out a reasonable value.

C. Node Leave

When a peer is about to leave, it will initiate a leaving message which is to send to its parents, noticing them to release the uplink bandwidth that they have used to uplink to the leaving peer. Afterwards another message will be sent to all of its children the peer currently streams to. The children peers are asked to look for new parents in the mesh in order to maintain the streaming rate. There are two levels of nodes affected upon a single node leave, its parent nodes and children nodes.

V. Illustrative Simulation Results

We carried out simulation to evaluate the performance of our schemes with two others, namely closest parent and Outreach. In the closest parent scheme, newly arrived peers choose parents which are closest to them. This scheme is slightly better than picking parents randomly as it can capture locality of the peers. The nice thing is that it is simple and this is why quite a number of streaming systems adopt it. The details of Outreach can be found in [20]. Also, the centralized scheme was included to serve as the ideal case for comparison purpose.

A. Simulation Setup and Metrics

In the simulation, we use Brite [24] to generate 10 different two levels top-down hierarchical topologies. Each topology consists of 8 autonomous systems each of which has 625 routers. This gives us a total of 5000 routers and about 20000 links in each topology. Brite also provides us latency with each link and we can interpret it as millisecond.

Peers are attached to the routers randomly and their access links bandwidth distribution follows the data suggested by

Uplink Bandwidth (unit)	Number of Peers (percentage)
0	30%
1 − 10	58%
20 − 90	5%
100	7%

TABLE I
UPLINK BANDWIDTH DISTRIBUTION OF PEERS.

[23]. One thing we want to emphasize here is that peers' bandwidths are neither constant nor uniformly distributed as one may have assumed. Instead, rich diversity [25],[26] has often been found in the Internet and this is true for end-hosts bandwidths too. In our case, as shown in Table I, the access links bandwidths distribution follows an extensive bandwidth measurement from a large scale real-world streaming event in [23]. (For the sake of clearness, we measure the bandwidth in streaming unit.) The streaming rate in simulations is 10 units and the number of peers used is 500 if not specified.

We also ran the simulations which bandwidths are more abundant and uniformly distritubted among peers. The results of those simulations follow the same trend as that presented here.

We define the following evaluation metrics that we would use in the analysis:

- *Delay:* The prime concern of our protocol is the source to end delay perceived by peers. It is the time taken for data to travel from the streaming server to the peers We measure both the average source to end delay among all peers and the maximum source to end delay of the mesh.
- *Hop Count:* We refer hop count to the number of intermediate peers involved on the overlay path from the source to a peer. Intuitively, hop count gives us an idea of the depth of the overlay. Small hop count does not necessary guarantee small delay, though, we can infer from the depth how the algorithms position peers. Putting high bandwidths peer near the source allows branching to occur earlier, thus giving a flatter toplology.
- *Source Workload:* Theoretically speaking, peer-to-peer system is scalable because the reliance on the streaming server is mostly eliminated. Nonetheless, the source is needed if parents with sufficient uplink capacity cannot be located, especially in the case of asymmetric bandwidth environment. A certain degree of source involvment can help keep the streaming mesh robust and efficient. The question is how much the source need to afford. Source Workload is defined as the amount of bandwidth that the source uses to upload data to the peers that directly connect to it. It measures the resource consumption level of the streaming server in the amount of streaming units used at its outgoing link.

B. Simulation Results

Before comparing with other schemes, we would like to argue for the merit of adaptation. Some may ponder that in our adaptation protocol the ancestors in fact gives way to the successors and they may be worse off as a result. It is true that this may happen to a few peers but in general most of them can gain in performance. Figure. 3 shows the average delay of peers against their bandwidth in a typical run of the simulation. Not only has the average delay been reduced by using adaptation, but the variance has also narrowed down. This means that by putting high bandwidth peers closer to the source, most peers, if not all, can receive the data streams sooner. In the following results, our scheme has always performed the adaptation with TTL value equal to 3.

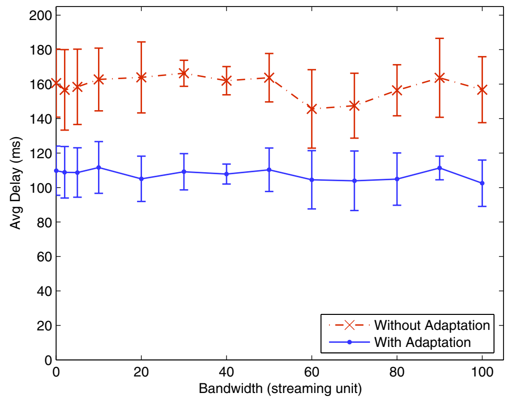
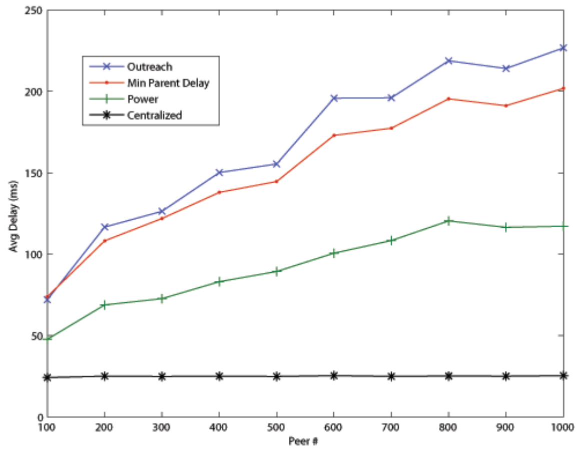


Fig. 3. Delay of peers with different bandwidths, TTL=3.

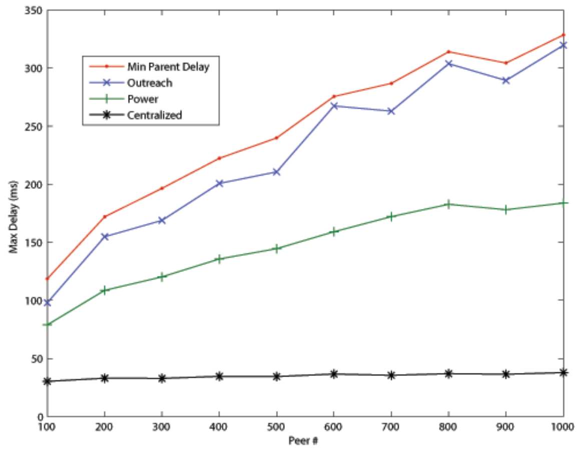
It should be noticed that adaptation does not guarantee bandwidth to vary inversely with delay, though, the peers with high bandwidth are moved upwards aggressively. There are two reasons for that. First, the ancestors of high-bandwidth peer may possess high bandwidth too, so adaptation does not happen between them. Moreover, those ancestors may have some other children (i.e. siblings to the successor) with lower bandwidth but the adaptation message never get to them. These peers, therefore, stay in the upper level of the mesh and enjoy relatively low delay.

The average and maximum delay of the four schemes we study are shown in Figure. 4. When there are not so many peers, the difference in performance of the schemes is nearly negligible. Nevertheless, we are interested in the performance with a large number of concurrent users. As the simulation shows, Power scheme outperforms the other two as the number of peers grows. The growth is also slower too. Having a small delay with low growth rate means the Power scheme is scalable to huge amount of peers. This illustrates the effectiveness of the Power scheme combined with adaptation in achieving low delay streaming mesh.

Figure. 5 shows the average and maximum hop count of the schemes under study. If we do not look at the bandwidth of the parents, it is likely we end up with a large mesh, as in the case of closest parents scheme. The Power scheme gives a more compact mesh than Outreach. This can be attributed to the nature of Outreach which picks bandwidth estimation randomly. High bandwidth peers in Power scheme are aggresively promoted upwards and thus more branchings occur near the source.

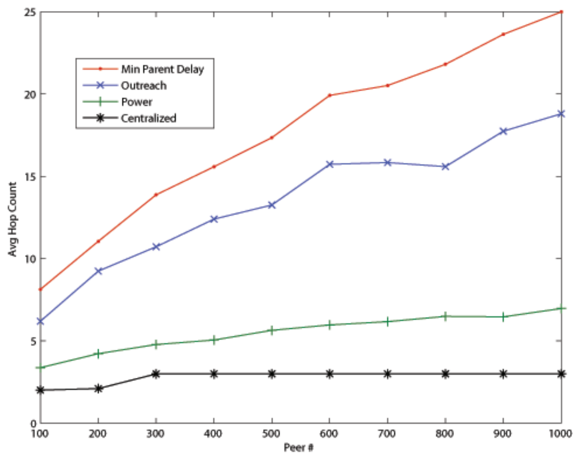


(a) Average

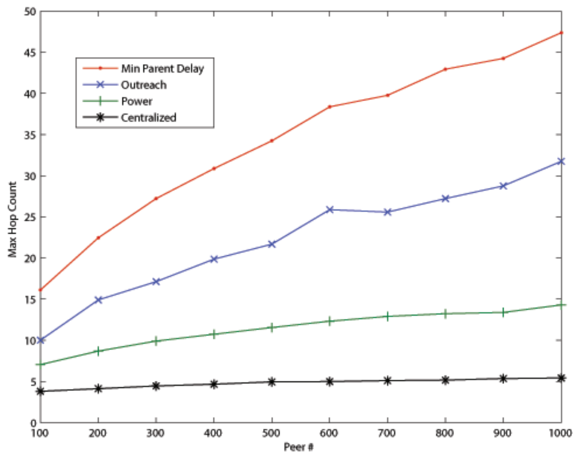


(b) Max

Fig. 4. Delay from Source to Peers.



(a) Average



(b) Max

Fig. 5. Hop Count.

Figure. 6 suggests the amount of bandwidth consumed at the source. We find that Ourtreach actively places peers under source and thus its reliance on the source is relatively larger. In Power and closest parent scheme, the source has roughly contributed the same amount of resources in order to keep the streaming mesh performing. In fact both of these schemes will try to utilize the peers bandwidth as much as possible. Sometimes after a considerable number of trial of searching good parents, still a newcomer cannot find out satisfactory parents. The best thing it can do is to connect to the streaming source. Although the source workload of these schemes are larger than the centralized scheme, in the lack of global knowledge these values are acceptable.

We define *Number of Adaptaion Change* as the number of existing connections that are broken in the adaptaion phase before the mesh reaches a static state.

We will evaluate the effect of different TTL values in the adaptation protocol. Figure. 7 shows the number of adaptation changes against the value of TTL. The more the adaptation happens, the more often connections are re-established. We can therefore think of the cost of adpatation proportional to

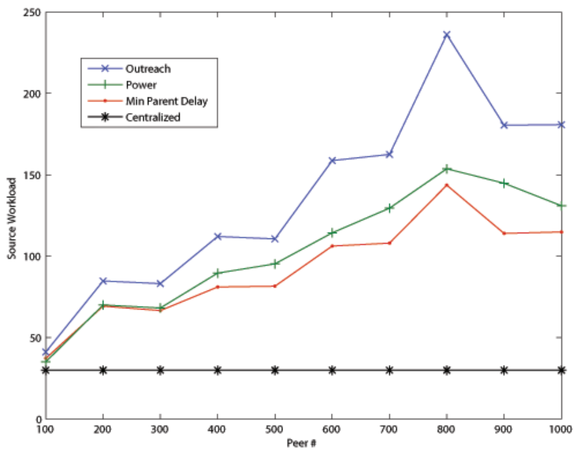


Fig. 6. Source Workload.

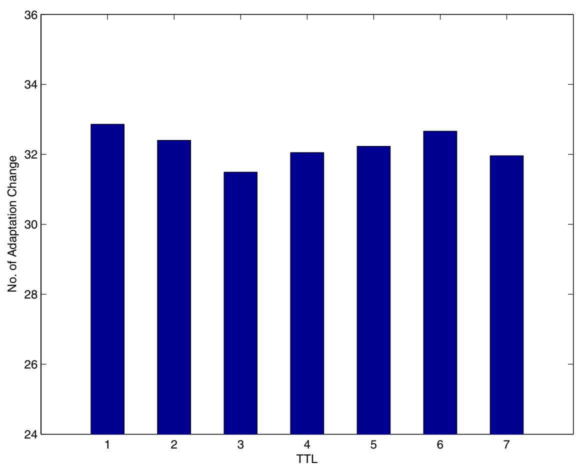


Fig. 7. Number of adaptation changes under different TTL.

the number of adaptation happened. As expected, small TTL values result in more adaptation. But increasing TTL does not reduce much adaptation changes. We found that actually most adaptation, even for large TTL values, only advance peers for a few levels. This indicates even a small change in peers position can give a considerable improvement over delay.

Beside comparing the number of adaptaion occurs, we would like to know how much the delay is reduced with different TTL. To this end, we define

- *Average Delay Reduction:* ratio of average delay reduced by adaption to average delay without adaptation;
- *Maximum Delay Reduction:* ratio of maximum delay reduced by adaption to maximum delay without adaptation;

If these values are positive, there is a reduction in delay time; otherwise the delay has in fact got longer.

Figure. 8 plots the average and maximum delay reduction. For clarity reason we only show the extreme cases of TTL=1 and TTL=7. We have tried with other TTL values and found that the graphs of other TTL values lie in between these two. Most of the time the adaptation can reduce average delay by as much as 30% and the maximum delay by 25%. This is important to delay sensitive streaming application because it allows shorter delivery time to transmit data from the source to end-hosts. In this way, the peers will be less likely to miss the playing deadline when it receives the data. Also, having large TTL value, only gives slightly better benefit. In other words, delay reduction is not proportional to the TTL. But increasing TTL will bring us the risk of flooding the overlay. Considering both the cost and benefits of different TTL values, we conclude that medium values, say 3, should be used.

VI. Conclusion and Future Work

In this paper, we study the problem of minimizing delay in live streaming mesh, formed by peers with asymmetric bandwidths. We have formulated the Minimum Delay Mesh Problem and derived a heuristics solution for it. Moreover, the heuristics is extended to a distributed protocol. Simulation results have shown our scheme achieving much lower delays than techniques commonly used. Despite our focus only on live streaming in this paper, we are confident that the protocol can play a critical role in other applications, e.g., video-on-demand. After all, it is always desirable to shorten time needed for data delivery.

Field experience [27] tells us that flash crowds are common in live streaming. Thousands of users appear suddenly when a popular program is about to broadcast. After the program has finished, vast amount of users leave simultaenoutly. In the future we would like to investigate how flash crowds impact our protocol and find out ways to deal with this problem.

References

[1] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "A measurement study of peer-to-peer file sharing systems," in *Proceedings of Multimedia Computing and Networking 2002 (MMCN '02)*, San Jose, CA, USA, January 2002.

[2] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "A measurement study of a large-scale p2p iptv system," *IEEE Transactions on Multimedia*, vol. 9, no. 8, 2007.

[3] T. Silverston and O. Fourmaux, "Measuring p2p iptv systems," in *In Proc. of 17th International workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV07)*, Jun. 2007.

[4] Y. Chu, S. Rao, and H. Zhang, "A case for end system multicast," in *Proceedings of ACM SIGMETRICS*, Santa Clara, CA, USA, Jun. 2000, pp. 1–12.

[5] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: High-bandwidth multicast in cooperative environments," in *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, The Sagamore, Bolton Landing (Lake George), New York, Oct. 2003, pp. 298–313.

[6] N. Magharei, R. Rejaie, and Y. Guo, "Mesh or multiple-tree, a comparative study of live p2p streaming approaches," in *IEEE INFOCOM*. Anchorage, Alaska, USA: IEEE, May 2007, pp. 1424–1432.

[7] E. Setton, J. Noh, and B. Girod, "Congestion-distortion optimized peer-to-peer video streaming," in *Image Processing, 2006 IEEE International Conference*, oct 2006, pp. 721–724.

[8] C. Y. Chan and J. Y. B. Lee, "A decentralized scheduler for distributed video streaming in a server-less video streaming system," in *4th IEEE/ACM International Symposium on Cluster Computing and the Grid*, Apr. 2004.

[9] H. Chi, Q. Zhang, J. Jia, and X. Shen, "Efficient search and scheduling in p2p-based media-on-demand streaming service," *IEEE Journal on Selected Areas in Communications*, vol. 25, pp. 119–130, Jan. 2007.

[10] E. Setton, J. Noh, and B. Girod, "Low latency video streaming over peer-to-peer networks," in *Proceedings of IEEE International Conference on Multimedia & Expo (ICME)*, Toronto, Ontario, Canada, 9-12Jul. 2006, pp. 569–572, invited paper.

[11] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," in *Proceedings of the 12th ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, Miami Beach, FL, USA, May 2002, pp. 177–186.

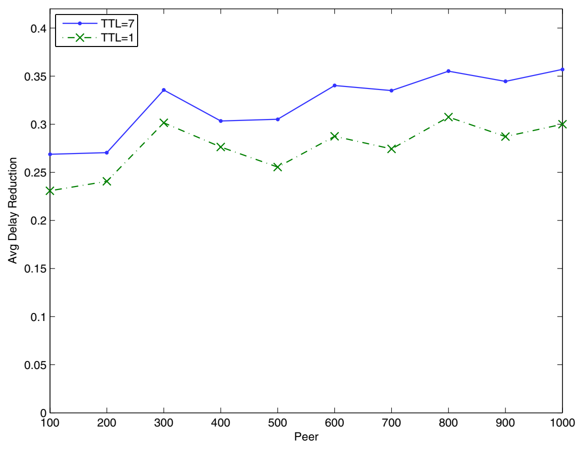
[12] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel, "ALMI: an application level multicast infrastructure," in *Proceedings of 3rd USENIX Symposium on Internet Technologies and Systems. (USITS'01)*, 2001, pp. 49–60.

[13] Y. H. Chu, S. G. Rao, S. Seshan, and H. Zhang, "A case for end system multicast," *IEEE JSAC*, vol. 20, no. 8, pp. 1456–1471, Oct. 2002.

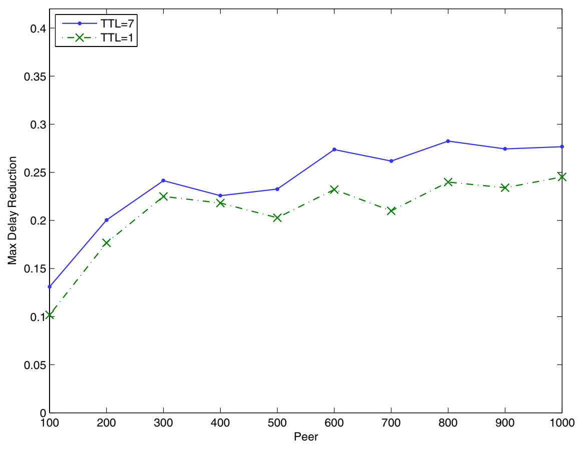
[14] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *Proc. ACM SIGCOMM'02*, Aug. 2002, pp. 205–217.

[15] D. A. Tran, K. Hua, and T. Do, "A peer-to-peer architecture for media streaming," *IEEE Journal on Selected Areas in Service Overlay Networks*, vol. 22, pp. 121–133, jan 2004.

[16] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. Mohr, "Chainsaw: Eliminating trees from overlay multicast," in *The Fourth International Workshop on Peer-to-Peer Systems*, feb 2005.



(a) Average



(b) Max

Fig. 8. Delay Reduction.

[17] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "CoolStreaming/DONet: A data-driven overlay network for live media streaming," in *Proceedings of IEEE INFOCOM*, Miami, FL, USA, Mar. 2005, pp. 2102–2111.

[18] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: High bandwidth data dissemination using an overlay mesh," in *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, The Sagamore, Bolton Landing (Lake George), New York, Oct. 2003, pp. 282–297.

[19] "GridMedia," http://www.gridmedia.com.cn/.

[20] T. Small, B. Li, and B. Liang, "Outreach: peer-to-peer topology construction towards minimized server bandwidth costs," *IEEE Journal on Selected Areas in Communications*, vol. 25, pp. 35–45, Jan. 2007.

[21] E. L. Lawler, J. K. Lenstra, A. H. G. R. Khan, and D. B. Shmoys, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. John Wiley & Sons, 1985.

[22] S. Y. Shi, J. S. Turner, and M. Waldvogel, "Dimensioning server access bandwidth and multicast routing in overlay networks," in *Proceedings of The 11th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 2001)*, Jun. 2001.

[23] K. Sripanidkulchai, A. Ganjam, B. Maggs, and H. Zhang, "The feasibility of supporting large-scale live streaming applications with dynamic application end-points," in *Proceedings of ACM SIGCOMM*, Portland, OR, USA, Aug. 2004, pp. 107–120.

[24] A. Medina, A. Lakhina, I. Matta, and J. Byers, "BRITE: Universal topology generation from a user's perspective," in *Proceedings of MASCOTS '01*, no. 2001-003, Jan. 2001.

[25] K. Sripanidkulchai, B. Maggs, and H. Zhang, "An analysis of live streaming workloads on the internet," in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, New York, NY, USA, 2004, pp. 41–54.

[26] E. Veloso, V. Almeida, W. M. Jr., A. Bestavros, and S. Jin, "A hierarchical characterization of a live streaming media workload," *IEEE/ACM Transactions on Networking*, vol. 14, pp. 133–146, Feb. 2006.

[27] X. Zhang, J. Liu, and B. Li, "On large-scale peer-to-peer live video distribution: Coolstreaming and its preliminary experimental results," in *IEEE Multimedia Signal Processing Workshop*, Shanghai, China, Oct. 2005, invited paper.