Computer Networks 74 (2014) 53–63

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Overlay live video streaming with heterogeneous bitrate requirements

Dongni Ren [*], Wang Kit Wong, S.-H. Gary Chan

Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Kowloon, Hong Kong, China

ARTICLE INFO

Article history:
Received 24 February 2014
Received in revised form 30 July 2014
Accepted 30 August 2014
Available online 16 September 2014

Keywords:
Live streaming
Heterogeneous bitrate
Parent-disjoint trees

ABSTRACT

We study a streaming cloud formed by distributed proxies providing live video service to diverse users (e.g., smart TVs, PCs, tablets, mobile phones, etc.). The proxies form a push-based overlay network, with each proxy serving a certain video bitrate for users to join. To form a proxy overlay serving heterogeneous bitrates, we consider that the video is encoded into multiple MDC (Multiple-Description Coding) streams with the serving bitrate of proxy i being k_i description streams. In order to effectively mitigate stream disruption due to node churns, proxy i also joins an additional r_i redundant MDC streams ($r_i \geqslant 0$) in such a way that all the $(k_i + r_i)$ streams are supplied by *distinct* parents. For live streaming, the critical issue is how to construct the *parent-disjoint* trees minimizing the assembly delay of the proxies.

We present a realistic delay model capturing important system parameters and delay components, formulate the optimization problem and show that it is NP-hard. We propose a centralized algorithm which is useful for a centrally-managed network and serves as a benchmark for comparison (PADTrees-Centralized). For large network, we propose a simple and distributed algorithm which continuously reduces delay through overlay adaptation (PADTrees-Distributed). Through extensive simulation on real Internet topologies, we show that high stream continuity can be achieved with push-based trees in the presence of node churns. Our algorithms are simple and effective, achieving low loss and low delay.

1. Introduction

We have witnessed in recent years the proliferation and penetration of Internet-ready multimedia-capable smart devices such as tablets, notebooks, PCs, smart TVs, set-top boxes, etc. These devices have different screen resolution and processing capabilities. In order to serve these devices, a streaming application has to meet heterogeneous bandwidth requirements ranging from, for example, 500 kbps for tablets to several Mbps for smart TV.

There has been increasing interest in providing live streaming services (such as Internet TV) offering heterogeneous bitrate to these smart devices. To achieve scalability and low delay, we consider a streaming cloud formed by distributed proxies placed close to user pools.[1] The proxies are light-weight content servers deployed by the content provider in the public Internet across multiple ISPs. They form an overlay, and each serves streams of a certain bitrate. Users on different devices join the proxies in their proximity of corresponding bitrate requirements to be served directly.

* Corresponding author.
E-mail addresses: tonyren@cse.ust.hk (D. Ren), wwongaa@cse.ust.hk (W.K. Wong), gchan@cse.ust.hk (S.-H.G. Chan).

[1] In this paper, we use “proxy” and “node” interchangeably.

http://dx.doi.org/10.1016/j.comnet.2014.08.019
1389-1286/© 2014 Elsevier B.V. All rights reserved.

In this work we focus on the cloud formed by the proxy overlay, and address its design and optimization issues. In order to efficiently support different bitrates of the proxies, the live video is encoded with Multiple-Description Coding (MDC) into K streams of similar bandwidth [1–7]. All the description streams are first generated at the source, and then pushed down to proxies via multiple delivery trees. Each description is distributed by a unique delivery tree. Note that these trees may not span all the proxies in the network. Proxy i receives some subset of the description streams k_i ($k_i < K, k_i \in Z^+$) according to its bitrate requirement. After receiving k_i description streams, the proxy re-assembles the descriptions into a full video, and then serves it to its users.

We consider the realistic case that the overlay network may be dynamic, i.e., the nodes may churn at any time (i.e., be introduced, removed or fail). Whenever there is a churn, the video at the downstream nodes will be disrupted. In order to offer high continuity, node i receives an *additional* r_i description streams as redundancy ($r_i \geqslant 0$). In this way, the requirement can be met as long as node i receives k_i streams or more from the streams delivered. In other words, in case of packet loss due to node churn, the redundancy streams are used to meet the rate requirement. It is clear that the total number of MDC streams encoded is the highest proxy requirement, i.e., $K = \max_i(k_i + r_i)$. Due to the use of MDC and redundancy, video quality would only be gracefully degraded if fewer than k_i streams are received at node i. The cost due to node churn is an increase in bandwidth because of the extra r_i streams a node receives.

To further protect stream continuity against unexpected node churns, all the $(k_i + r_i)$ description streams of proxy i are supplied by *distinct* parents. The challenge is hence how to construct the K *parent-disjoint* description trees in order to minimize source-to-end delay due to stream assembly while meeting the heterogeneous rate requirements of the proxies. We tackle this problem by presenting a realistic node model on delay, formulating the optimization problem, analyzing its complexity, and designing effective optimization algorithms (centralized and distributed).

Push-based overlay live streaming has been shown to achieve substantially lower delay than pull-based approach [8,9]. However, there has not been work on the design and *optimization* of a *push-based* overlay with proxies of *heterogeneous* rates. Our approach is shown to achieve low delay with high continuity. Previous approaches are often based on a random pull-based mesh, where a node continuously searches for neighbors (using gossip) and pulls content from them. This rather uncoordinated *ad-hoc* connectivity clearly is not bandwidth-efficient and may not even meet heterogeneous requirements. Furthermore, as the major objective of pull-based approach is to aggregate a full video, it seldom optimizes source-to-end delay, leading to unsatisfactory delay and resource (bandwidth) utilization. On the other hand, much of the previous tree-based overlay work has not sufficiently considered MDC with redundant streams to achieve stream continuity. We propose and optimize a push-based overlay structure composed of multiple trees and redundancy streams to mitigate node churns. Our overlay meets heterogeneous bitrate requirements of proxies with high stream continuity.

Fig. 1 shows an example of our streaming overlay with three MDC streams, i.e., $K = 3$. Nodes A to F serve videos with different rate requirements: $k_i = 2$ for nodes A to D, and $k_i = 1$ for nodes E and F. They all receive one more description stream as redundancy. Nodes A, B and C are directly connected to the streaming server where they receive all their streams. Because the streaming server is stable, they do not need any redundant stream. On the other hand, because node D is not served by the streaming source, it connects to *distinct* parents in order to achieve fault-tolerance in streaming, i.e., D receives from three parents three descriptions with one as redundant stream. Nodes E and F both receive two descriptions from distinct parents while they require only one for viewing.

Because a proxy can decode and serve the video only after it assembles all the required streams, its delay from the source is the *slowest path* out of all the $(k_i + r_i)$ trees (i.e., the maximum-delay path). Such delay increases quickly if the trees are not constructed properly. The challenge is how to construct the *parent-disjoint* trees to achieve minimum delay. We propose algorithms, termed PADTrees (**Pa**rent-**D**isjoint Trees), to construct highly efficient trees achieving low delay and high continuity while meeting heterogeneous bitrate requirements.

The contributions of our study are:

- *Delay model, problem formulation and its complexity analysis:* Given K MDC streams, we present a rather realistic and comprehensive delay model for a node capturing all the major network and delay components such as scheduling delay (due to fanout of a node), edge bandwidth, end-to-end bandwidth, propagation delay, etc. With the model, we formulate the delay optimization problem which is to design MDC trees that minimize the diameter (i.e., worst-case delay) of the overlay

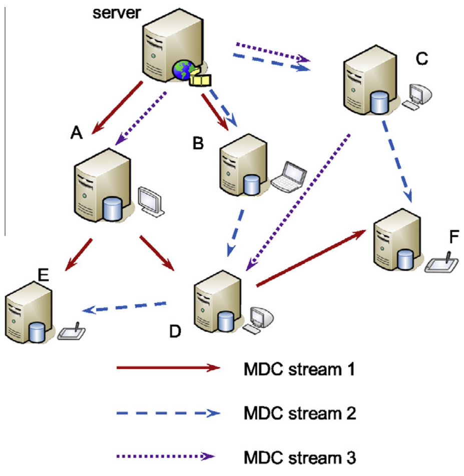

Fig. 1. An example of live overlay streaming with heterogeneous requirements, showing the constituent underlying delivery trees.

network given heterogeneous proxy bitrate requirements. We show that the problem is NP-hard.
- *PADTrees-Centralized:* Because the problem is NP-Hard, we propose a heuristic to construct parent-disjoint delivery trees to achieve low delay. The heuristic, termed PADTrees-centralized, can be computed by a controller for a centrally-managed network. It also inspires the design of the distributed algorithm, and serves as a benchmark for the comparison of various algorithms to show the benefit of central planning.
- *PADTrees-Distributed:* For a large distributed network, we propose PADTrees-Distributed, a distributed algorithm for each node to search for $(k_i + r_i)$ distinct parents to achieve low delay. Our algorithm is simple, efficient (i.e., low overhead), and maintains high stream continuity in the presence of node churns. It adapts to network conditions with nodes adjusting their positions in the trees to reduce its delay. Such adaptation is proved to converge to some steady value.

The organization of the paper is as follows. After discussing related work in Section 2, we formulate the optimization problem and its complexity analysis in Sections 3. The centralized and distributed versions of PADTrees are discussed in Sections 4 and 5, respectively. Illustrative simulation results are presented in Section 6. We conclude in Section 7.

2. Related work

Single-tree overlay structure was initially proposed to distribute streams, where all nodes are arranged into a tree rooted at the source [10–13]. Although these approaches are simple and achieve low delay, the streaming rate cannot be guaranteed as it is limited by the lowest bottleneck bandwidth in the tree. Furthermore, the failure of a node leads to stream disruption of all its descendants. Therefore, single-tree approach cannot be applied in our setting with node churns.

To address the weaknesses of single-tree structure, using forest of multiple trees (or a push-based mesh) was proposed [14–16]. Multiple interior node-disjoint trees are constructed to mitigate single parent failure. However, they have not addressed the issues of node churns and heterogeneous bandwidth requirements [17]. The works in [18,19] study the scalable video services for different classes of devices, but have not studied the *optimization* of the forest structure. We formulate the construction of multiple trees as an optimization problem, and our proposed algorithms lead to a highly optimized overlay. The multiple delivery trees accommodate heterogeneous bandwidth requirements while achieving low delay and high continuity in spite of node churns.

Another body of work uses pull-based approach for data exchange [20,21]. In this approach, nodes continuously search for and connect to their closest neighbors using gossip to pull data. The work in [22] proposes an age-based membership protocol for participating nodes, which is adaptive to node churns. The work in [23] studies the complementary nature of node joining and leaving to handle churn problem in a pull-based network. Despite its simplicity, this approach often leads to high delay and poor resource utilization (due to its random connectivity). It also has high control overhead (due to message and bitmap exchange). We show that push-based overlay can achieve stream continuity with much lower source-to-end delay.

Multiple description coding (MDC) has been widely used in media streaming to address bandwidth heterogeneity. The video source encodes data into multiple descriptions. At the receiver, the streaming quality depends on the number of descriptions received [1–4,24]. The work in [8,25] discusses using MDC in overlay video streaming to meet heterogeneity requirement. However they have not studied how to minimize delay in the presence of node churns. Our previous work in [26] discusses how to construct multiple *spanning trees* for *homogeneous* users with FEC. And there are other studies the error recovery and system resilience for overlay video streaming [27]. Our work advances from it by considering *heterogeneous user requirements*. Such requirement leads to a different problem formulation and algorithmic approaches to construct multiple *parent-disjoint* trees to meet heterogeneity and stream continuity requirements. We further present a centralized heuristic as a benchmark for comparison and to inspire the design of distributed algorithm. There are works on the overlay construction for users with heterogeneous bitrate requirement as well as upload capacity [28,29]. However they only consider stable nodes and overlays. In this paper we study how to achieve stream continuity in a dynamic node environment.

3. Problem formulation and complexity analysis

Consider an overlay network modeled as a directed graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices representing the nodes in the overlay (including the server) with $n = |\mathcal{V}|$, and $\mathcal{E} = \mathcal{V} \times \mathcal{V}$ is the set of overlay edges between nodes. Let $s \in \mathcal{V}$ represents the source. The edge cost of $\langle i, j \rangle$, denoted as d_{ij}, represents the unicast delay from node i to node j, which equals to the sum of the propagation delay d^p_{ij} and the scheduling delay d^s_{ij} from node i to node j, i.e., $d_{ij} = d^p_{ij} + d^s_{ij}$. Some of the important symbols used in the paper are given in Table 1.

The source video is split and encoded into K MDC streams of similar bandwidth. Each node has its own streaming rate requirement k_i, which is the number of description streams it requires. To provide error tolerance against churns, let r_i be the additional number of description streams that node i receives ($r_i \geqslant 0$). Clearly, $K = \max_{i \in \mathcal{V}}(k_i + r_i)$. To mitigate the adverse effect of node churn, all the K streams are distributed using distinct delivery trees in such a way that each proxy is served by $k_i + r_i$ *distinct* parents. Note that proxies directly connected to the source receive all the k_i streams from the source (as the source is assumed to be reliable) and no redundant stream is needed. A delivery tree is rooted at s and contains all the nodes that is receiving a specific stream. Denote the full set of K MDC trees as $\mathcal{K}$. Further let $\mathcal{P}_i$ be the set of all $(k_i + r_i)$ parents of node i.

Table 1
Important symbols used in the paper.

Symbol	Meaning
n	Number of proxies in the network (including the server)
K	Number of MDC streams ($= \max_i(k_i + r_i)$)
k_i	The required number of MDC streams of node i
r_i	The additional number redundant streams for node i
s	The source node
$\mathcal{V}$	The set of nodes in the overlay
$\mathcal{E}$	The set of edges between nodes
$\mathcal{G}_l$	The set of nodes in (partial) delivery tree $l, 1 \leqslant l \leqslant k$
$\mathcal{K}$	The set of all K MDC trees
$\mathcal{P}_i$	The set of $k_i + r_i$ parents of node i
$\mathcal{C}_i$	The set of children of node i
b	Bit rate of a substream, the bandwidth unit (kb/s)
C	Video segment size (bits)
u_i	Bandwidth of node i (unit)
c_i	Number of streams that node i is pushing
w_{ij}	End-to-end throughput of edge $\langle i,j\rangle$ (units)
d_{ij}	Delay from node i to j (seconds)
d^s_{ij}	Scheduling delay from node i to j (seconds)
d^p_{ij}	Propagation delay from node i to j (seconds)
D^l_i	Delay of node i in tree $l, 1 \leqslant l \leqslant K$ (seconds)

We refer b as the basic unit of network bandwidth, which is the bandwidth reserved by the parent for a single end-to-end connection (i.e., parents stream to each of their children at rate b).

For every node i in V, it has an uplink bandwidth of u_i units ($u_i \in Z^+$), which represents the maximum total number of children it can serve in all spanning trees. We consider all nodes have enough downlink bandwidth to receive the video streams. The end-to-end throughput of the edge $\langle i,j\rangle$ is denoted as $w_{ij} \in Z^+$, which is the maximum number of substreams that can simultaneously accommodate in edge $\langle i,j\rangle$. For any node in V, if it gets an aggregate of k_i out of the $k_i + r_i$ streams from its parents, we call the node *fully served*. In other words, if node i receives k_i streams from all K spanning trees, it is *fully served* and can play back the video with continuity. Note that s has an uplink bandwidth of u_s units and has no parent.

The worst-case scheduling delay from node j to node i, denoted as d^s_{ji}, is given by

$$d^s_{ji} = \sum_{k \in \mathcal{C}_j} \frac{L}{\min(w_{jk}, u_j)b/t_{jk}}, \tag{1}$$

where L (kbits) is the segment size used in streaming, and t_{jk} is the number of concurrent streams on edge $\langle j,k\rangle$. $\min(w_{jk}, u_j)b$ is the maximum rate (in kbps) that node j can stream to node k. Therefore $L/(\min(w_{jk}, u_j)b/t_{jk})$ represents the time needed to transmit one segment in every substreams that goes from j to k. This delay is the total amount of time that a node needs to wait until a segment is fully delivered by its parent. The transmission delay between the parent and the node is included in d^s_{ji} as well.

Our problem is to minimize the worst-case delay of the network. Consider a packet transmitted from s at time 0 via a delivery tree l, where $l \in \mathcal{K}$. Node i gets its packet from its parent in tree l. Denote the delay of the packet to node i through its parent j as D^l_i, which is obviously given by

$$D^l_i = D^l_j + d_{ji}. \tag{2}$$

Denote D_i as the maximum delay (including recovery stream) for the packet to arrive at node i. By definition,

$$D_s = 0. \tag{3}$$

The maximum delay of node i, i.e. the time taken for a node to aggregate the entire stream before playback, is determined by the slowest path, i.e.,

$$D_i = \max_{l \in \mathcal{K}} D^l_i. \tag{4}$$

Minimum Delay Robust Trees Problem (MDRT): The MDRT problem is to construct a streaming overlay with dynamic nodes with K delivery trees rooted at the source and each node i having $k_i + r_i$ distinct parents (k_i source parents and r_i redundant parents) so that the worst-case delay of the nodes is minimized, i.e.,

$$\min \max_{i \in \mathcal{V}} D_i. \tag{5}$$

Claim. MDRT problem is NP-Hard.
Proof. Travelling salesman problem (TSP) is reducible to MDRT in polynomial time. An input to TSP is a weighted, undirected complete graph $G(\mathcal{V}, \mathcal{E})$ and a vertex $s \in \mathcal{V}$. The TSP is to find a tour of minimum cost through all vertices exactly once (Hamiltonian cycle) such that s is both the starting point and ending point.

First of all the MDRT problem is in NP. The maximum delay of a mesh can be calculated in polynomial time. Therefore given a graph $G(\mathcal{V}, \mathcal{E})$, and the min–max delay of the problem, we can verify whether the mesh is the optimal solution. Now we prove that TSP can be reduced into MDRT problem. The polynomial time transformation is as follows. Let $G'(\mathcal{V}', \mathcal{E}')$ be the graph of a TSP instance. We transform $G'(\mathcal{V}', \mathcal{E}')$ into $G(\mathcal{V}, \mathcal{E})$ by adding a vertex S_{end} and 0 cost edges from all vertices to S_{end}. In this way, the vertices in $\mathcal{V}$ represent nodes and the weight on edges is the delay between two adjacent nodes. We let S be the source, and consider the special case that the uplink bandwidth of each node is b, and S_{end} has zero uplink bandwidth. Consider also the streaming rate for all nodes to be b and they do not seek redundant parents, i.e., $k_i = 1, r_i = 0, \forall i \in \mathcal{V}$. In this way the resulting overlay topology must be a chain starting at S and ending at S_{end}. $\max_{i \in \mathcal{V}} D_i$ equals to the delay of S_{end} which is the sum of all delays preceding it. Hence it is obvious that $\max_{i \in \mathcal{V}} D_i$ in G is minimum if and only if the cost of a tour in G' is minimum. Therefore, TSP is polynomial reducible to MDRT. The complexity of a similar problem is proved to be NP-Hard in [15]. □

4. PADTrees-centralized: a centralized heuristic and its run-time complexity

Because MDRT problem is NP-hard, in this section we propose and present a simple and effective centralized heuristic to construct a overlay with minimum delay given k_i source parents, r_i redundant parents for each node $i \in \mathcal{V}$, and the server capacity u_s, the maximum number of nodes that the server can support. The heuristic is suitable for a small centrally-managed network, and can also serve as the benchmark for our study of distributed algorithm.

The algorithm constructs the K delivery trees iteratively by putting nodes one at a time into a partially constructed tree. Let the set of nodes in K partially constructed trees be labeled as $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_k$. Our algorithm is initialized as follows. The source s is first added to all the K trees. We associate with each of the those nodes a priority score, which determines which ones to connect to the source. Obviously those high-bandwidth nodes close to the server should be connected to the server with higher priority. To balance between the bandwidth and distance, the score of node i is calculated as u_i/d_{si} (this has been shown to be effective from [15]). Nodes with the highest scores become the children of s and hence are added to all the trees to get the streams. Note that in order to give each MDC stream similar availability in the overlay, the streaming server randomly distribute the K streams to the nodes.

After the initialization steps, we push one node into one delivery tree in each iteration. Define δ_i^l as the delay of node i in tree l at the current stage, i.e.,

$$\delta_i^l = \delta_j^l + d_{ji}, \tag{6}$$

where j is the parent of i in tree l and, by definition, $\delta_s = 0$. Let $\mathcal{C}_i^l$ be the set of candidate parents of node i in tree $l, 1 \leqslant l \leqslant K$, and $\mathcal{P}_i$ be the set of current parents of node i. Clearly,

$$\mathcal{C}_i^l = \mathcal{G}_l \setminus \mathcal{P}_i. \tag{7}$$

Let c_i be the number of streams that node i is streaming to its children at the current stage of the algorithm. The residue bandwidth of i then equals $u_i - c_i$. Nodes with higher residue bandwidth and close to the partial trees should have higher priority in entering the trees. Define the priority score of node i connecting to j in l as $p_{ij}^l, 1 \leqslant l \leqslant K$, which is given by

$$p_{ij}^l = \frac{u_i - c_i}{\delta_j^l + d_{ji}}. \tag{8}$$

The max priority score of node i in delivery tree l is then defined as the p_i^l, i.e.,

$$p_i^l = \max_{j \in \mathcal{C}_i^l} p_i^l j. \tag{9}$$

The algorithm is shown in Algorithm 1. We calculate the priority scores for all nodes in all partial trees, and selects the node $\hat{i}$ with maximum score to push into the overlay with the corresponding parent and tree, i.e.,

$$\hat{i} = \arg \max_{i \in \mathcal{V} \setminus \mathcal{G}_l, 1 \leqslant l \leqslant K} p_i^l. \tag{10}$$

The scheduling delay is updated accordingly. This process ends when all participating nodes obtain the required number of streams, i.e.,

$$|\mathcal{P}_i| = k_i + r_i, \quad \forall i \in \mathcal{V}. \tag{11}$$

The complexity of the algorithm is as follows. It takes $O(n)$ time for a node to calculate its priority score in one tree. There are $n - 1$ nodes and K trees in total, so it takes us $O(Kn^2)$ time to calculate the all scores in one iteration. Since there are $O(Kn)$ iterations in total, the construction time for K delivery trees is $O(K^2 n^3)$.

Algorithm 1. Centralized Algorithm

```
Require: Input $K, \mathcal{V}$.
Ensure: Output $K$ delivery trees $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_K$ all
   rooted at the source $s$.
 1: $L$ = sorted list of $i \in \mathcal{V} \setminus \{s\}$ in descending order
    of $u_i/d_{si}$
 2: $x = 0$
 3: while $u_s > c_s$ do
 4:   while $L(x)$ is not fully served do
 5:     Randomly select a tree $l, L(x) \notin l$
 6:     Push $L(x)$ into $\mathcal{G}_l$
 7:   end while
 8: end while
 9: while Not all nodes are fully served do
10:   $x$ = currently the best node to enter
11:   $y$ = the potential parent for $x$
12:   for $l = 1$ to $k$ do
13:     for all $i$ not yet fully served do
14:       for all $j \in \mathcal{G}_l$ not yet fully served do
15:         Calculate $p_i^l j$
16:       end for
17:       if $p_i^l > p_x^l$ then
18:         $x = i$
19:         $y = j$
20:       end if
21:     end for
22:   end for
23:   Push $x$ into $\mathcal{G}_l$ with parent $y$
24: end while
```

5. PADTrees-distributed: a distributed algorithm to construct low-delay streaming overlay

The centralized heuristic works well to minimize delay for a centrally-managed network. However in a distributed proxy streaming network, we do not have global knowledge. Furthermore, nodes may arrive and leave at any time. The distributed algorithm should be adaptive to node churns. In this section, we present a simple and fully distributed algorithm which is scalable to large group to reduce mesh delay. There are four operations of the algorithm.

5.1. Node join

A new arrival, say node i, has to have $k_i + r_i$ parents to assemble a playable video in a dynamic network environment. To achieve that, it contacts a rendezvous point (RP) which returns a number of nodes in the overlay as the pool of candidate parents. It may enlarge the pool by requesting neighbors from these nodes.

Node i then checks the delay d_{ji} with each candidate j. In addition to network distance, node i also asks j for source-to-end delay D_j^l in each spanning tree l and the amount of its available bandwidth (which is simply u_j minus the number of children that j has). In other words, the nodes in the overlay compute their delays according to Eq. (4).

Given a delivery tree l, node i selects the node with the minimum delay among the candidates (i.e., $\min_j(D_j^l + d_{ji})$) and connects to it to retrieve the stream. The selected parent is removed from the pool and node i repeats the process for the remaining delivery trees. In this way, it joins all $k_i + r_i$ delivery trees to fulfill the streaming rate requirement.

5.2. Node departure and failure

Each proxy in the network periodically sends its "heartbeat" to its parents and children. Upon detecting that a parent of tree l has left, the child node contacts RP to retrieve a new list of proxies, and tries to rejoin tree l by selecting a new parent from the proxy list. During the rejoin process, In the mean time if the lost parent serves one of the k_i source streams, the affected nodes use the redundant streams to recovery the lost data.

5.3. Adaptation

In a distributed environment, there is no specific *a priori* joining and leaving order of nodes. As a result, a node may need to adapt to the dynamic network condition to move to a better position in the mesh to reduce delay. This is especially important when a high-bandwidth node joins the network at a later time. Node movement to better position is the design objective of tree adaptation.

Fig. 2 shows the flow diagram of the adaptation process of a node in the network. The adaptation consists of three steps:

- *Request:* A child i periodically inspects its residual bandwidth. If this is greater than the streaming rate, i sends its parents a REQUEST for adaptation. The REQUEST message contains its available bandwidth and a time-to-live field (TTL) indicating the scope of the REQUEST is to be flooded. When a node receives a REQUEST message, if the TTL field is greater than zero, it decrements TTL and forwards the REQUEST message to its neighbors including parents and children (except the one from which the message comes from).
- *Grant:* Upon receiving a REQUEST message, the candidate checks whether its uplink bandwidth is less than the residual bandwidth of the requester i, the request originator. If this is the case, the candidate sends the requester i a GRANT response which contains its delays (which may be in hops) in each delivery tree. The GRANT message indicates that the adaptation between the requester and the candidate is permitted. Note that our adaptation mechanism can also use the quality fluctuation as an indicator for node adaptation. In this work we focus on the network re-arrangement of the adaptation mechanism, and uses available bandwidth as indicator for simplicity.
- *Accept:* Child i then chooses the slowest tree l from all $k_i + r_i$ spanning trees to adapt. At this stage, child i may have received a number of GRANT messages from different ancestors. Among the ancestors (upstream nodes) who have sent GRANT messages to it, child i accepts the ancestor j that satisfies the following conditions: (i) $D_j^l < D_h^l + d_{hi} < D_i^l$, where h is the parent of j in tree l (i.e., i can improve its delay by changing parent to h); and (ii) available uplink bandwidth of node j is less than i. Obviously, h-i-j does not form any parent–child relationship in the original tree. After such j is found, i replaces its existing parent in the tree with h. In addition, j connects to i to get the substream l. This is possible because the conditions guarantee both i and j still retains the property of distinct parents. Moreover, the second condition

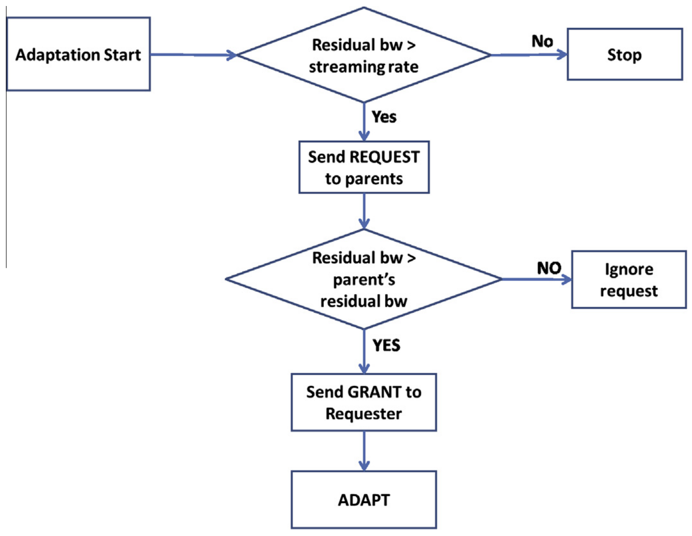

Fig. 2. A flow diagram of a node adaptation.

makes sure that we are moving high-bandwidth nodes to a position above the low bandwidth ones. In summary, child i takes up the position of the ancestor j in tree l and in turn provides the sub-stream l to j. It is clear that after the process, the delay of the slowest tree is reduced.

6. Simulation environment and illustrative results

We carry out simulations to evaluate the performance of the proposed algorithms. For comparison purpose, we also simulate one state-of-the-art scheme *TOMTree* [30], as well as two traditional scheme, namely, *closest parent* and *random parent*. *TOMTree* is a multi-tree based topology that minimize the average height of sub-stream trees and the average propagation latency in each tree. In the *closest parent* scheme, nodes look for the closest parents for streaming. Since this scheme captures locality among nodes, it (or its variant) is widely adopted for overlay streaming with satisfactory performance [31,11]. While the *random parent* scheme randomly chooses parents for newly-arrived nodes. This scheme does not capture the locality of the nodes, thus streaming mesh constructed in this way is often of high delay.

6.1. Simulation setup and metrics

In the simulation, nodes arrive at the overlay according to a Poisson process at rate λ (request/s) and then remains in the overlay for an exponential length of time (seconds). During their life-time, proxies follow the proposed algorithms to search for parents. The node holding time is according to an exponential distribution with rate μ (requests/s). We have used other distribution and the results are qualitatively the same. There are 480 nodes on average in our baseline setting. To be fair in comparison, Each node in all schemes looks for the same number of parents to achieve the same level of streaming quality. The bandwidth requirements are 2, 4, 6, 8, with the same number of nodes. We use a video encoded by H.264 with a resolution of 1280×960, and the bandwidth requirements are 320×240, 640×480, 960×720, 1280×960 respectively.

Our simulation is carried out on a real Internet topology provided by CAIDA [32], which was collected on June 12th, 2011 and contains 1747 routers and 3732 links. The round trip times (RTTs) between inter-connected routers are also given in the topology. We use Distance-vector routing to compute the latencies between any two routers in the network. Nodes are randomly attached to the routers. Unless otherwise stated, we have used the baseline parameters as shown in Table 2. Note that u_i is normally distributed with mean 4 Mbps and standard deviation 1 Mbps (We only take the positive values). The search time for a new parent is uniformly distributed continuous random variable from 1 s to 15 s. The performance metrics of interest are:

- *Delay:* The primary concern of our protocol is the source-to-end delay of the proxies. It is measured by summing all the link delays on the overlay path from the source to the proxy. Because there are multiple paths to reach a node, the maximum delay of all the paths is taken as the measure. Delay is measured according to Eqs. (2)–(4). We are interested in the average, maximum and distribution of the source-to-end delay in the overlay.
- *Loss rate:* When ancestors of a node leave the network, the proxy may experience data loss. In our scheme, each node i obtains r_i more streams for redundancy in the streaming overlay besides its k_i source streams. As long as k_i out of these $k_i + r_i$ streams are received by the node, it is able to recovered full video and will not be affected by the node leave. If a node i receives less than k_i stream in total, its video quality will be partially affected. Its residual video quality equals to number of MDC streams received after recovery divided by k_i; and the loss rate equals to one minus residual video quality. The overall loss rate of a node is equal to its average loss rate over time. We define the worst-case loss rate of the network as the largest overall loss rate among all nodes.

Table 2
Baseline parameters.

Parameter	Value
u_i	4 Mbps (average)
b	500 kps
K	10
k_i	2–8
λ	1 req/s
$1/\mu$	480 s
Parent search time	0–15 s
C	100 kbits
Target loss rate	6%

6.2. Illustrative simulation results

We choose r_i for each proxy according to its streaming rate requirement k_i and the target loss rate. Fig. 3 shows the loss rate of nodes with different requirements given

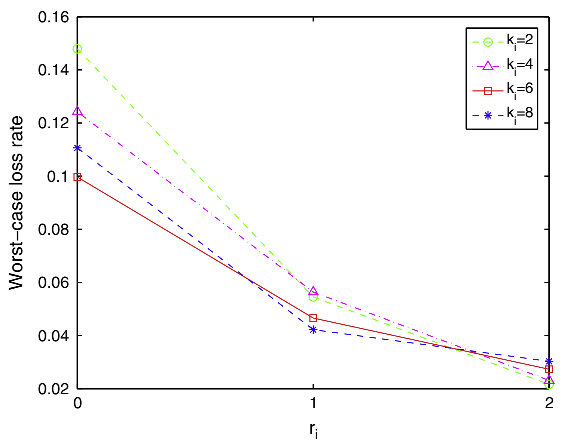

Fig. 3. Worst-case loss rate versus *t*.

different number of redundant streams in PADTrees-Distributed. The loss rate significantly drops with the increase of the number of redundant streams. Therefore given a target loss rate, we can decide how many redundant streams each proxy needs to obtain in order to meet this continuity requirement. In our study we set the target loss rate as 6% maximum. Therefore all nodes only need to find one more extra parent for redundant.

We show in Fig. 4 the loss rate distribution of PADTrees-Distributed. We observe that with one redundant stream, most of the proxies achieve very high stream continuity (More than 90% of the proxies have a loss rate less than 2%). Only a few nodes suffer from larger losses.

We compare PADTrees with other schemes by plotting the average and maximum delay versus proxy arrival rate λ in Figs. 5 and 6. In general, the delay increases with the arrival rate. This is because the system population increases with arrival rate, which leads to longer overlay diameter and hence delay. Clearly, the PADTrees-Centralized performs the best due to its complete knowledge. There is some performance loss in PADTrees-Distributed since it is a distributed protocol. Nevertheless, PADTrees-Distributed performs significantly better than TOMTrees, as well as the traditional Closest parents scheme and random parents scheme. Given the best performance of PADTrees-Centralized, having global information is beneficial. The PADTrees-Centralized can position the nodes effectively. This achieves much better delay, which stays low even when the number of nodes increases. The Closest parent scheme, despite its forming close-neighbor groups among proxies, performs the worst. There are two reasons for this. First, it has not considered source-to-end delay. Each node only greedily selects parents with shortest RTT to it. On the other hand, PADTrees achieves low delay by putting nodes closer to the source. Second, Closet neighbor does not consider how many children a node is serving, thus often leads to a parent node overwhelmed and with high scheduling delay. Random parents scheme does not have good delay performance either because it has not considered the locality of the nodes.

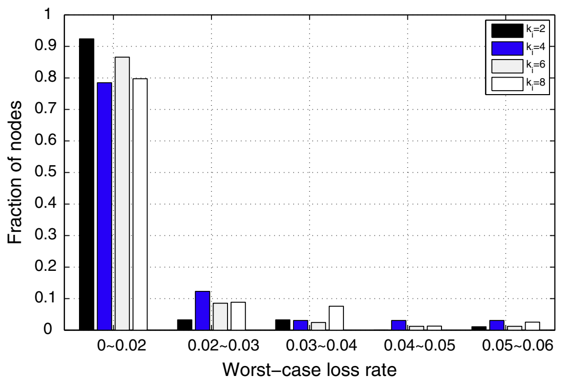

Fig. 4. Loss rate distribution.

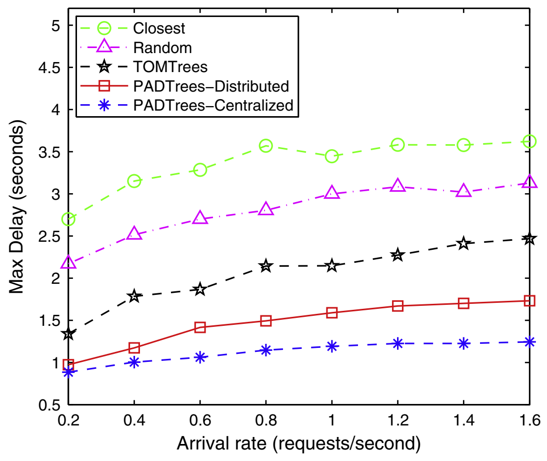

Fig. 6. Maximum delay versus arrival rate.

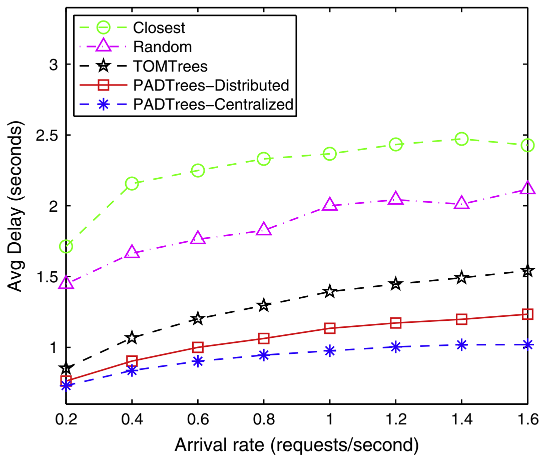

Fig. 5. Average delay versus arrival rate.

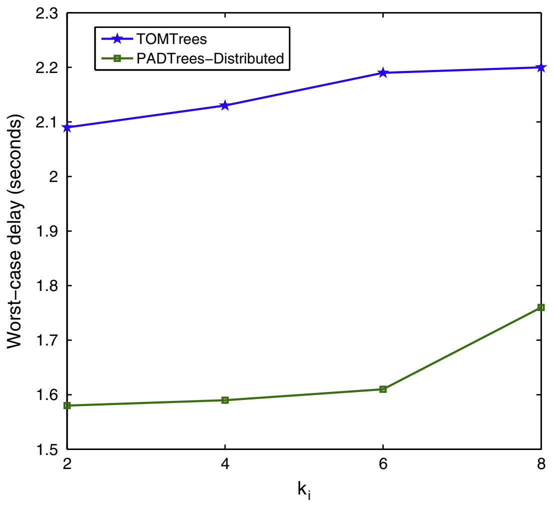

Fig. 7. Delay of PADTrees versus k_i.

Fig. 7 shows the performance of PADTrees on proxies with different requirements. We observe that PADTrees generally outperforms the state-of-the-art scheme TOM-Trees. The maximum delay of the proxies increases with $k_i + r_i$, which means proxies with larger streaming rate requirement suffer from longer delay from the source. Recall from Eqs. (2) and (4), the delay of a node is determined by the slowest path of all streams. Nodes with high requirement obtain more description streams, and hence they have larger delay than the nodes with low streaming rate.

In Fig. 8 we show the comparison of delay distribution between PADTrees and other schemes. Clearly, PADTrees outperforms other schemes with more low-delay nodes. In general, PADTrees-Centralized performs the best since it has complete knowledge of the network. The performance of PADTrees-Distributed is also very well when most proxies achieve low delay (0.9–1.5 s). The worst-case delay is not much larger than the other nodes. It demonstrates that PADTrees is able to arrange the overlay in a way that most of the nodes in the overlay share rather similar delays, and the worst-case delay is optimized.

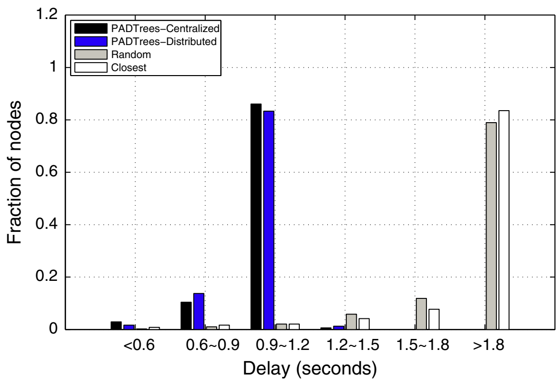

Fig. 8. Delay distribution for different schemes.

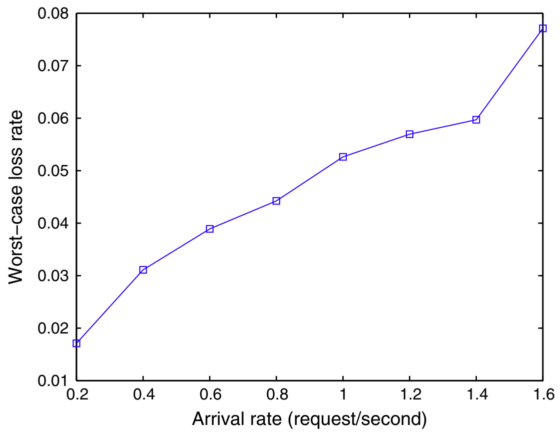

Fig. 9. Worst-case loss rate versus proxy arrival rate.

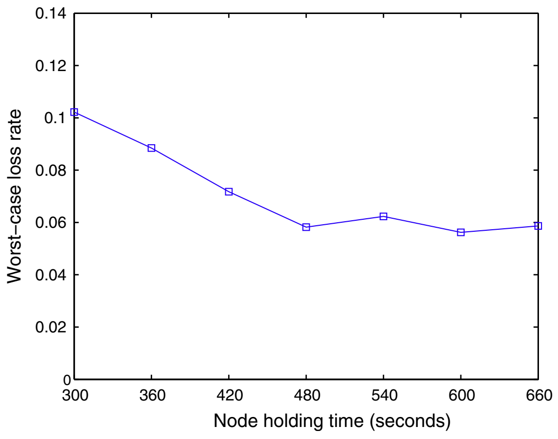

Fig. 10. Worst-case loss rate versus proxy holding time.

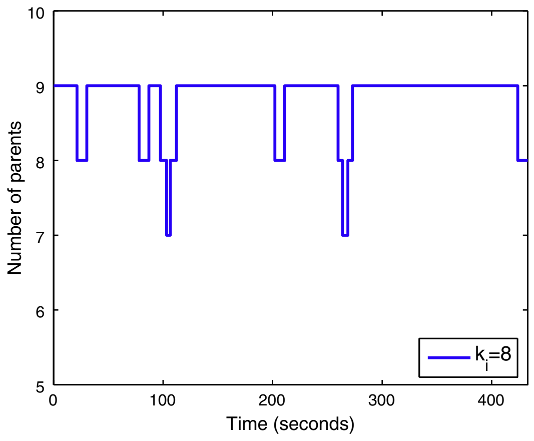

Fig. 11. Parent dynamic of a node.

Fig. 9 shows the worst-case loss rate versus the arrival rate of the proxies. When arrival rate increases, the participating nodes in the streaming network increases. The height of the delivery trees grows larger and the continuity of the nodes is more easily affected by the churns of their ancestors. Therefore the loss rate increases with the proxy arrival rate.

Fig. 10 shows the worst-case loss rate versus average holding time of the proxies. The loss rate decreases as the holding time increases. This is because when the holding time is larger, nodes stay connected to their parents longer and hence experience less interruption, and loss rate therefore improves. However when the holding time is greater than some value (550 s), there is no obvious improvement in loss rate. This is because the number of proxies in the streaming overlay increases with holding time and thus the tree depth also gets higher. Each proxy will have a larger hop count to the source and hence much easier to be affected by the churns of its ancestors.

Fig. 11 shows the dynamic of a proxy's number of parents during its lifetime. We trace the continuity of a representative node i with different quality requirement $k_i = 8$. Node i in Fig. 11 has $r_i = 1$. It is able to reassemble the video at required quality as long as it receives 8 description streams or more. Therefore node i only experiences the reduce of quality twice for very short time when its number of parents drops to 7.

7. Conclusion

In this work, we have studied low-delay and high-continuity live streaming overlay formed by proxies with heterogeneous bitrate requirements. The video stream is encoded into K description streams. Node i obtains k_i streams according to its own bitrate requirement and r_i redundant streams to meet its target loss rate. To offer error resilience, all the $(k_i + r_i)$ streams are pushed by *distinct* streaming parents in multiple trees. We have addressed the overlay design problem, which is to form a minimum-diameter network given k_i and r_i for all nodes. We show the problem NP-hard, and propose a simple centralized heuristics which can be implemented by a controller and used for benchmarking purpose. We also propose a scalable and distributed algorithm that adaptively constructs parent-disjoint trees to achieve low delay and high video continuity.

We have conducted extensive simulation on Internet topologies to study the performance of our algorithms. The results show that a centrally-managed network can construct a much better overlay network. The distributed version achieves much lower source-to-end delay as compared with the traditional closest parent and random schemes. It also achieves high stream continuity despite the dynamic behavior of the network. The results show that push-based trees are effective to meet heterogeneous bitrate requirements, even under network dynamics.

Acknowledgements

This work was supported, in part, by Hong Kong Research Grant Council (RGC) General Research Fund (610713) and The Hong Kong Innovation and Technology Fund (UIM/246).

References

[1] V.K. Goyal, Multiple description coding: compression meets the network, IEEE Signal Process. Mag. (2001) 74–93.
[2] V. Stankovic, R. Hamzaoui, Z.X. Xiong, Robust layered multiple description coding of scalable media data for multicast, IEEE Signal Process. Lett. 12 (2) (2005) 154–157.
[3] M. Ma, O.C. Au, S.-H.G. Chan, Multiple description coding for error-resilient video transmission, WSEAS Trans. Comput. 4 (10) (2005) 1426–1431.
[4] C.-W. Hsiao, W.-J. Tsai, Hybrid multiple description coding based on H.264, IEEE Trans. Circ. Syst. Video Technol. 20 (2010).
[5] S.-H.G. Chan, X. Zheng, Q. Zhang, W. Zhu, Y.-Q. Zhang, Video loss recovery with FEC and stream replication, IEEE Trans. Multimedia 8 (2) (2006) 370–381.
[6] G. Dan, V. Fodor, I. Chatzidrossos, On the performance of multiple-tree-based peer-to-peer live streaming, in: INFOCOM 2007: 26th IEEE International Conference on Computer Communications, May 2007, pp. 2556–2560.
[7] D. Jurca, P. Frossard, A. Jovanovic, Forward error correction for multipath media streaming, IEEE Trans. Circ. Syst. Video Technol. 19 (2009).
[8] N. Magharei, R. Rejaie, Y. Guo, Mesh or multiple-tree, A comparative study of live P2P streaming approaches, in: IEEE INFOCOM, IEEE, Anchorage, Alaska, USA, 2007, pp. 1424–1432.
[9] W. Jiang, S.-H.G. Chan, M. Chiang, J. Rexford, K.-F.S. Wong, C.-H.P. Yuen, Proxy-P2P streaming under the microscope: fine-grain measurement of a configurable platform, in: Proceedings of the 19th International Conference on Computer Communications and Networks (ICCCN) (Invited paper), 2–5 August 2010.
[10] H. Yin, X. Liu, Zhan, Livesky: enhancing CDN with P2P, ACM Trans. Multimedia Comput. Commun. Appl. 6 (2010) 16:1–16:19.
[11] T. Small, B. Li, B. Liang, Outreach: peer-to-peer topology construction towards minimized server bandwidth costs, IEEE J. Select. Areas Commun. 25 (2007) 35–45.
[12] X. Jin, K.-L. Cheng, S.-H.G. Chan, SIM: scalable island multicast for peer-to-peer media streaming, in: Proc. IEEE International Conference on Multimedia Expo (ICME), Toronto, Canada, July 2006, pp. 913–916.
[13] J. Chen, S.-H.G. Chan, V.O.K. Li, Multipath routing for video delivery over bandwidth-limited networks, IEEE J. Select. Areas Commun. 22 (10) (2004) 1920–1932. Special Issue on Design, Implementation and Analysis of Communication Protocols.
[14] P. Fraigniaud, H.-A. Phan, "Tree-farms" for tree-based multicast schemes in peer-to-peer overlay networks, in: 2010 IEEE International Conference on Communications (ICC), May 2010, pp. 1–5.
[15] D. Ren, Y.-T.H. Li, S.-H.G. Chan, On reducing mesh delay for peer-to-peer live streaming, in: IEEE INFOCOM, IEEE, Phoenix, Arizona, 2008.
[16] D. Ren, Y.T.H. Li, S.H.G. Chan, et al., Fast-mesh: a low-delay high-bandwidth mesh for peer-to-peer live streaming, IEEE Trans. Multimedia 11 (8) (2009).
[17] W. Zhang, Z. Li, Q. Zheng, Samp: supporting multi-source heterogeneity in mobile p2p iptv system, IEEE Trans. Consumer Electron. 59 (4) (2013) 772–778.
[18] J. Liu, Y. Zhang, J. Song, Energy saving multicast mechanism for scalable video service using opportunistic scheduling, IEEE Trans. Broadcast. PP (99) (2014). 1–1.
[19] D. Grois, O. Hadar, Recent trends in online multimedia education for heterogeneous end-user devices based on scalable video coding, in: 2013 IEEE Global Engineering Education Conference (EDUCON), March 2013, pp. 1141–1146.
[20] F. Pianese, D. Perino, J. Keller, E.W. Biersack, PULSE: an adaptive, incentive-based, unstructured P2P live streaming system, IEEE Trans. Multimedia 9 (8) (2007).
[21] N. Magharei, R. Rejaie, PRIME: peer-to-peer receiver-driven mesh-based streaming, IEEE/ACM Trans. Netw. 17 (2009) 1052–1065.
[22] H. Liu, X. Liu, W. Song, W. Wen, An age-based membership protocol against strong churn in unstructured P2P networks, in: Proceedings of the 2011 International Conference on Network Computing and Information Security, NCIS '11, vol. 02, 2011, pp. 195–200.
[23] X. Meng, X. Chen, Y. Ding, Using the complementary nature of node joining and leaving to handle churn problem in P2P networks, Comput. Electr. Eng. 39 (2) (2013) 326–337.
[24] Y. Xu, C. Zhu, W. Zeng, X.J. Li, Multiple description coded video streaming in peer-to-peer networks, Signal Process.: Image Commun. 27 (2012) 412–429.
[25] F. de Asis Lopez Fuentes, Adaptive mechanism for P2P video streaming using SVC and MDC, in: International Conference on Complex, Intelligent and Software Intensive Systems (CISIS), 2010, pp. 457–462.
[26] D. Ren, W. Wong, S.-H. Chan, Toward continuous push-based P2P live streaming, in: 2012 IEEE Global Communications Conference (GLOBECOM), 2012, pp. 1969–1974.
[27] C.-L. Chang, W.-M. Chen, C.-H. Hung, Reliable consideration of p2p-based vod system with interleaved video frame distribution, IEEE Syst. J. 8 (1) (2014) 304–312.
[28] M. Yang, Y. Yang, Optimal overlay construction on heterogeneous live peer-to-peer streaming systems, in: 2010 39th International Conference on Parallel Processing (ICPP), September 2010, pp. 690–698.
[29] L. Cui, Y. Jiang, J. Wu, Employing qoS driven neighbor selection for heterogeneous peer-to-peer streaming, in: 2011 IEEE International Conference on Communications (ICC), June 2011, pp. 1–6.
[30] C. Liang, Y. Liu, K. Ross, Topology optimization in multi-tree based P2P streaming system, in: 21st International Conference on Tools with Artificial Intelligence, 2009 (ICTAI '09), 2009, pp. 806–813.

[31] X. Liao, H. Jin, Y. Liu, L.M. Ni, D. Deng, Anysee: peer-to-peer live streaming, in: Proc. IEEE Infocom, 2006, pp. 2411–2420.

[32] CAIDA: cooperative association for internet data analysis, <http://www.caida.org>.

Dongni Ren is currently a Ph.D. candidate at the Department of Computer Science and Engineering in the Hong Kong University of Science and Technology (HKUST), supervised by Prof. Gary Chan. He also received his B.Eng. in Computer Science (Information Engineering) from HKSUTin 2007, and his M.Phil. in Computer Science from HKUST in 2009. His research interest includes live streaming technologies, multimedia networking, overlay and peer-to-peer networks.

S.-H. Gary Chan (S'89-M'98-SM'o3) is currently Full Professor and Undergraduate Programs Coordinator at the Department of Computer Science and Engineering and Director of Sino Software Research Institute, The Hong Kong University of Science and Technology (HKUST), Hong Kong. He received M.S.E and Ph.D. degrees in Electrical Engineering from Stanford University (Stanford, CA) in 1994 and 1999, respectively, with a minor in business administration. He obtained his B.S.E. degree (highest honor) in Electrical Engineering from Princeton University (Princeton, NJ) in 1993, with certificates in Applied and Computational Mathematics, Engineering Physics, and Engineering and Management Systems. His research interest includes multimedia networking, overlay streaming, wireless networks and mobile applications.