Optimizing Substream Scheduling for Peer-to-Peer Live Streaming

K.-H. Kelvin Chan 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: {chankh, gchan}@cse.ust.hk

Ali C. Begen
Video & Content Platforms
Research & Advanced Development
Cisco Systems, Inc.
San Jose, CA 95134 USA
Email: abegen@cisco.com

Abstract—**In peer-to-peer (P2P) live streaming using unstructured mesh, packet scheduling is an important factor on overall playback delay. The hybrid pull-push approach has been recently proposed to reduce delay compared to classical pulling method. In this approach, video are divided into substreams and packets are pushed with low delay. There has been little work addressing the scheduling problem on substream assignment. In this paper, we study the scheduling problem on assigning substreams to minimize packet delay. Given heterogeneous contents, delays and bandwidths of parents, we formulate the Substream Assignment (SA) problem to assign substreams to parents with minimum delay. The SA problem can be optimally solved in polynomial time by transforming it into a Max-Weighted Bipartite Matching problem. Simulation results show that our distributed algorithm achieves substantially lower delay as compared with traditional pull and current hybrid pull-push approaches based on greedy algorithm.**

I. Introduction

In peer-to-peer (P2P) live streaming, peers collaboratively organize themselves into an overlay and share their upload capacities to serve others. In order to provide robustness against peer churns and to meet the streaming bandwidth requirement, a mesh is usually used in the overlay, where each peer connects to some other peers as its *parents* [1]. By retrieving packets from its parents, a *child* can aggregate and assemble a full stream to achieve stream continuity.[1] With multiple parents, a child needs to determine which parents should deliver which packets, given their heterogeneous bandwidths and available contents. This is called *scheduling*, which incurs some delay due to control messaging and packet buffering. In an overlay, such scheduling delay can be substantial if the scheduling is not designed properly; reducing such delay is therefore critical.

Traditionally, mesh-pull is often used on mesh overlay due to its simplicity, robustness and high bandwidth utilization (such as [2], [3]). Mesh-pull is based on the exchange of buffermap, through which a child can know exactly the packets that each parent has and can explicitly *pull* packets from each of them. However, these benefits come with the cost of long delay, due to buffermap advertisement and the pulling at the child. Because scheduling delay propagates along the overlay path, mesh-pull often results in high delay especially in a large network.

Recently, a hybrid pull-push approach has been proposed to incorporate the benefits of high bandwidth utilization of pulling and low delay of pushing [4], [5], [6]. In this approach, most packets are pushed to peers by dividing packets into substreams. The missing packets are then recovered by pulling. Naturally, the substream assignment is vital in contributing the packet delay. The substream assignments in [4], [5], [6] are done greedily as a plug-in to the traditional pull system, which is simple to implement. However, they do not achieve delay optimality. A scheduling is proposed in [7] that aims to provide differentiated video quality with the aid of layered coding.

In this paper, we minimize scheduling delay for P2P live streaming by optimizing the substream push assignment. Given a mesh topology and bandwidth from its parents, the child computes a substream schedule for its parents. As long as the network conditions do not change considerably, the schedule does not need to be adjusted. The video is divided into substreams of equal bandwidth, each of which is *optimally* pushed by a parent. Given the available contents, delays and uplink bandwidths of parents, we formulate the Substream Assignment (SA) problem to achieve the lowest delay for assigning substreams. The SA problem can also be optimally solved.

There has also been much work on mesh construction [8], [9]. Our work is orthogonal to them and focuses on the *scheduling* given a mesh; our scheduling can apply on their work to achieve better performance.

The major contributions of this study are as follows:

1) *The formulation and optimized solution of the Substream Assignment (SA) problem :* By considering the delay of packets from a parent to the child, we formulate the SA problem as an optimization problem to minimize the overall delay of the packets. We show that by transforming into a Max-Weighted Bipartite Matching

This work was supported, in part, by the General Research Fund from the Research Grant Council of the Hong Kong Special Administrative Region, China (611209), and the Cisco University Research Program Fund, a corporate advised fund of Silicon Valley Community Foundation (SVCF08/09.EG01&GMGS08/09.EG05).

[1]For clarity in exposition, we will focus on a child with its multiple parents in this paper.

problem, the SA problem can be solved exactly and efficiently in polynomial time.

2) *Simulation studies and comparisons:* With the optimal solution to the SA problem, we present a simple distributed protocol, which pushes packets in substreams and recovers lost packets by pulling. Using simulations, we study our scheme and compare its performance with the current pull and hybrid pull-push approaches. Our results show that our scheme indeed achieves a substantially lower delay.

The rest of the paper is organized as follows. We first present in Section II the system design of our scheme. Then, we present the formulation and solution of the SA problem in Section III. In Section IV, we present illustrative simulation results. We conclude the paper in Section V.

II. System Design

We start our discussion by presenting the overview in Section II-A. Then, we present the system protocol in Section II-B.

A. Overview

We consider that control messages are sent through reliable channel (using TCP) while data packets may be subject to losses (due to late or lost packets). Let R bits/s be the streaming rate of the video stream. The stream is composed of packets of constant size L bits that are uniquely identified by sequence numbers. The packets are interleaved into N substreams of bitrate R/N bits/s, i.e., substream j contains all packets whose sequence numbers and j are congruent modulo N.

We consider a simple multi-thread design of the system, where each child is served by a new thread of the parent. Each thread is allocated a certain bandwidth (which can be implemented simply by certain bit transmission quota per some time interval). As compared with the round-robin scheduling, such multi-thread system has the strength that a parent-child connection of low end-to-end bandwidth would not starve out other children in the service queue (the so-called head-of-line blocking). Using this design, without loss of generality we hence can focus on a certain *child* and its *parents*. The child has a set of parents denoted by $\mathcal{P}$. For parent i in $\mathcal{P}$, it allocates a certain uplink bandwidth to the child for substreams. For each parent $i \in \mathcal{P}$, the maximum number of substreams that can be pushed to the child is denoted by m_i ($m_i \in \mathbb{N}$). Denote $M = \sum_{i \in \mathcal{P}} m_i$. Clearly, $M \geq N$ is required, so that parents can support the whole stream of the video. Each parent in $\mathcal{P}$ also notifies the latest packet of each substream in its buffer, denoted by the vector $[l_{i,1}, l_{i,2}, ..., l_{i,N}]$, where $l_{i,j}$ is the latest packet sequence number of substream j at parent $i \in \mathcal{P}$. τ_i denotes the time when the most updated vector $[l_{i,j}]$ has been received at the child. Some important notations are summarized in Table I.

TABLE I: Table of Nomenclature.

Notation	Definition
$\mathcal{P}$	Set of all parents
N	Number of substreams
R	Streaming rate (bits/s)
L	Packet size (bit)
$l_{i,j}$	The latest packet sequence number at parent i for substream j
m_i	Maximum number of substreams that parent $i \in \mathcal{P}$ allocated to the child
M	$M = \sum_{i \in \mathcal{P}} m_i$
τ_i	The time when the most updated vector $[l_{i,j}]$ has been received at the child from parent i
$C(i,j)$	Delay cost of assigning parent i to substream j

B. Protocol

When a child joins the system, it connects to its parents and asks for their buffer information. Upon the vectors $[l_{i,j}]$ of all parents in $\mathcal{P}$ are received, the child computes a substream push schedule according to the SA algorithm (presented in Section III). The child re-computes a new schedule when it experiences changes in network conditions, such as the departure of a parent, or a change in the uplink bandwidth (e.g., by 10%), etc. Then, the child issues push requests to corresponding parents for the substream(s). Parents push substream(s) accordingly and the same schedule will be used repeatedly until a new schedule is computed or the child is dead.

Similar to the hybrid pull-push approaches in [4] and [5], the lost packets from substream push are pulled. A packet is identified as lost when a packet hole in the substream is detected by the child after a waiting timeout (e.g., 1 second in simulations). The child then pulls the lost packet from its parents in $\mathcal{P}$. The child re-pulls a packet for several times if the pulled packet is dropped by the parents or in transmission.

III. Substream Assignment (SA) Problem

We first formulate the SA problem as an delay optimization problem in Section III-A, and then present a polynomial-time solution in Section III-B.

A. Problem Formulation

Given the latest packet vector $[l_{i,j}]$ from parent i (received at some time τ_i), and the maximum number of substreams allocated to the child m_i, the SA problem is to find a substream assignment achieving the minimum total source-to-child delay. Note that not all parents in $\mathcal{P}$ have to be assigned.

Let $C(i,j)$ be a function denoting the (delay) cost of packets if parent i is assigned to substream j, where a larger $C(i,j)$ means higher source-to-child delay. Note that SA problem does not assume any form of $C(i,j)$.

The goal of the SA problem is to find an assignment of substreams to parents in $\mathcal{P}$, i.e., find $A : \{1, 2, \cdots, N\} \rightarrow \mathcal{P}$,

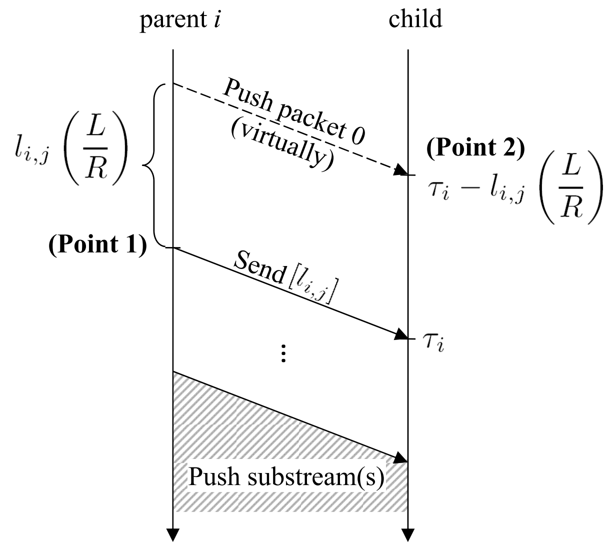
parent i
child
$l_{i,j}\left(\frac{L}{R}\right)$
Push packet 0 (virtually)
(Point 2)
$\tau_i - l_{i,j}\left(\frac{L}{R}\right)$
(Point 1)
Send $[l_{i,j}]$
τ_i
Push substream(s)

Fig. 1: The timeline for virtual arrival between parent i and the child.

such that substream j is assigned to parent $A(j) \in \mathcal{P}$ for $j \in \{1, 2, \cdots, N\}$, so as to

$$\text{Minimize} \sum_{j=1}^{N} C(A(j), j), \qquad (1)$$

subject to the bandwidth constraint

$$|A^{-1}(i)| \leq m_i, \forall i \in \mathcal{P}, \qquad (2)$$

where $A^{-1}(i)$ is the set of assigned substream(s) to parent i in A, and $|A^{-1}(i)|$ is the size of the set.

One may consider a delay cost as follows. We illustrate in Fig. 1 the timeline to send substreams from parent i to the child, where Point 1 is the time the parent i sends packet sequence numbers of $[l_{i,j}]$ to the child. Since the packet sequence numbers are received at time τ_i, $\tau_i - l_{i,j}\ (L/R)$ (Point 2) represents the arrival time of packet 0 if it *were* pushed to the child. This can be interpreted as the "virtual" starting time for substream j at the child from parent i, which can also be viewed as a reference for the "timeliness" of the substream. Clearly, the earlier this virtual arrival time is, the lower the delay is. Therefore, we may consider the delay cost, $C(i, j)$, as

$$C(i, j) = \tau_i - l_{i,j}\left(\frac{L}{R}\right). \qquad (3)$$

B. An Exact Solution Based on Max-Weighted Bipartite Matching

The SA problem can be solved exactly in polynomial time, as it can be transformed into an equivalent *Max-Weighted Bipartite Matching (MWBM)* problem. This is shown as follows. The objective of SA problem in (1) is to minimize $\sum_{j=1}^{N} C(A(j), j)$, which is equivalent to maximize $-\sum_{j=1}^{N} C(A(j), j)$. We consider a complete bipartite graph

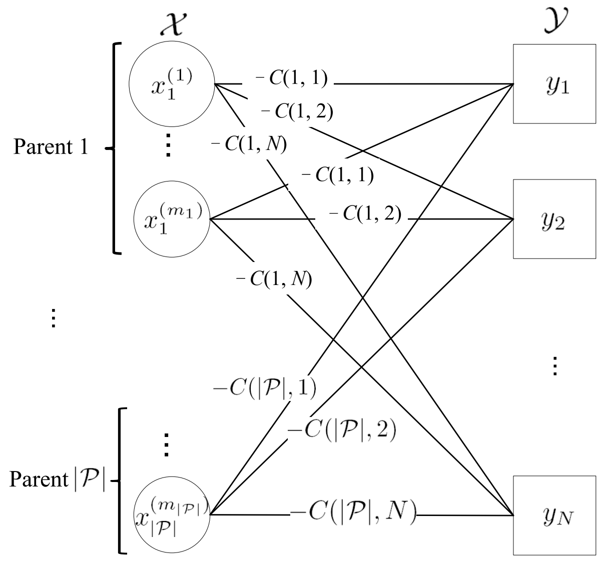

Fig. 2: A bipartite graph for the SA solution.

$\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with bipartition $\mathcal{V} = \mathcal{X} \cup \mathcal{Y}$ as illustrated in Fig. 2. $\mathcal{X}$ is the set of nodes $x_i^{(k)}$ for each parent $i \in \mathcal{P}$, where $k = 1, \cdots, m_i$. With $M = \sum_{i \in \mathcal{P}} m_i$, we have $|\mathcal{X}| = M$. For each $x_i^{(k)} \in \mathcal{X}$, it represents a possible assignment slot of parent i. $\mathcal{Y}$ is the set of nodes y_j, for $j = 1, 2, ..., N$. Node $y_j \in \mathcal{Y}$ represents substream j, therefore $|\mathcal{Y}| = N$. The weight of edge $(x_i^{(k)}, y_j)$ is set to be $-C(i, j)$.

The optimal solution of the SA problem can be transformed directly from the MWBM solution on $\mathcal{G}$. For every matching $(x_i^{(k)}, y_j)$ in the MWBM solution, we assign substream j to parent i in the SA problem. The MWBM problem can be solved efficiently by the Hungarian Algorithm [10]. Therefore, the time complexity of SA problem follows with the MWBM problem, which is $O(N^3 + M^3)$.

IV. Illustrative Simulation Results

We have conducted a simulation study on our scheme (referred to *SA - opt*). We present the simulation environment and metrics in Section IV-A, and illustrative results in Section IV-B.

A. Simulation Environment and Metrics

We compare our scheme with the following approaches:

- *Pull:* The pull scheme adopts a rarest-first algorithm (used in, for example, CoolStreaming [2]). If a packet is lost, the same packet is re-pulled again (at most twice). The bitmap exchange period of 500 ms is used as suggested in [5].
- *Hybrid:* A hybrid pull-push scheme reduces the delay by pushing substreams. Similar to the one used in [5], the substream assignment algorithm is done in a greedy fashion. If a packet is lost, the same packet is pulled by the children for once.

We have implemented an event-driven simulator in C++. The results are taken at steady state and averaged over all peers. We generate Internet-like two-level top-down topology using BRITE with default parameters [11]. BRITE also provides the underlying link latencies in milliseconds. Peers are randomly attached to different underlying routers. Unless otherwise stated, we use the following baseline parameters: $R = 512$ Kbps, $L = 8$ Kbits, $N = 8$ and the number of peers is 500. For simplicity, we model the transmission loss rate between two peers as uniformly distributed from 2% to 10% (so the average transmission loss rate is 6%). The loss rate between a pair of peers is maintained throughout an instance of simulation. The upload bandwidth of the source is 4 Mbps, while the upload bandwidth of a normal peer is uniformly distributed between 512 Kbps and 1 Mbps. As our main focus is on low delay, we do not enforce tight bandwidths to allow higher scheduling flexibility. For all schemes, the waiting timeout is 1 second.

Since our focus is on packet scheduling rather than mesh construction, we use a simple mesh construction algorithm. Every peer sequentially joins the system and is randomly assigned to a default number (10 in our case) of online peers with a non-zero residual uplink bandwidth as parents. Every parent randomly allocates some uplink bandwidth for substream pushing for each of its children. The number of substreams supported, m_i, follows a uniform distribution on $[0, \lfloor R/2 \rfloor]$.

The following performance metrics are used in our simulations:

- *Residual Loss Rate:* It is the percentage of packets that cannot be successfully received at a peer.
- *Packet Delay:* It refers to the source-to-peer delay of each packet for a peer. This shows the general delay performance of packets. As the video playback is delayed by the highest packet delay, we call the maximum packet delay of all packets as *playback delay* of a peer. We are also interested in its distribution and average.
- *Bandwidth Dilation:* It is defined as the percentage of extra upload bandwidth consumed over the raw streaming rate R. It measures the bandwidth surplus due to packet retransmission, redundant packet transmission or control overhead. Note that with transmission loss, the bandwidth dilation is unlikely to be less than the expected transmission loss rate 6%.

B. Illustrative Results

We first compare in Fig. 3 the average packet delay versus number of peers for different schemes. Clearly, the average packet delay generally increases with number of peers in the network. The average packet delay of pull is relatively high, due to the buffermap exchange and the round-trip delay for the pull requests. The hybrid scheme cuts delay significantly (by around 50%) by pushing part of the video in substreams. Our scheme achieves even lower average packet delay than hybrid (by near 25%), because the substreams are pushed according to the SA problem, which is optimized for low delay.

We also compare in Fig. 4 the playback delay of the three schemes versus number of peers. The CDF of the playback delay of the three schemes for 500 peers is also shown in Fig. 5. Similar to the average packet delay, our scheme reduces the playback delay substantially compared to the hybrid and pull schemes (by around 20% and 55%, respectively). In our scheme and the hybrid scheme, the pulled packets usually account for the playback delay. This shows that the delay of pulled packets are also reduced. The explanation is rather intuitive. Since in our scheme the pushed packets (in substreams) are optimized for minimum delay, the lost packets can be detected and pulled earlier compared to the current hybrid approaches. The playback delay for our scheme shows less variation than the pull and hybrid schemes across peers.

Fig. 6 shows the residual loss rate for the three schemes against number of peers in the network. A low residual loss rate is essential for providing smooth playback at peers. Generally, the residual loss rate for all schemes are similar. Since the lost packets are pulled and re-pulled, the residual loss rates are far below the transmission loss rate.

Fig. 7 shows the bandwidth dilation of the three schemes versus number of peers. Note that the expected transmission loss rate is 6%. The bandwidth dilations of pull is comparatively lower (around 7%), while that of our scheme

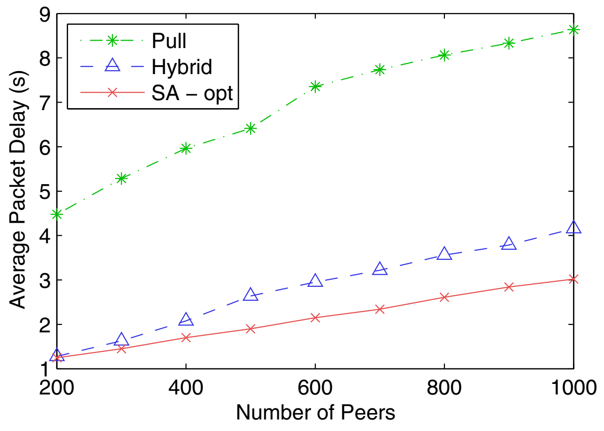

Fig. 3: Average packet delay versus number of peers.

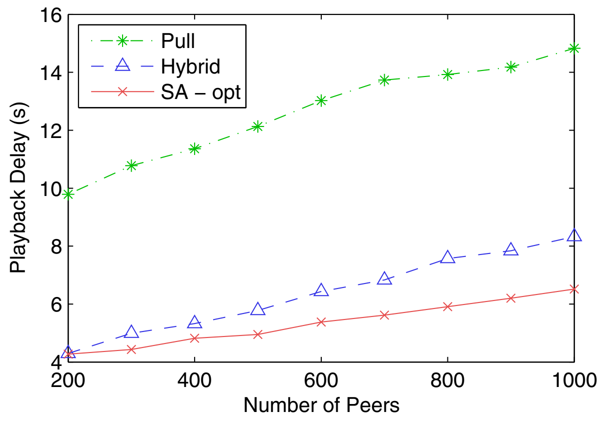

Fig. 4: Playback delay versus number of peers.

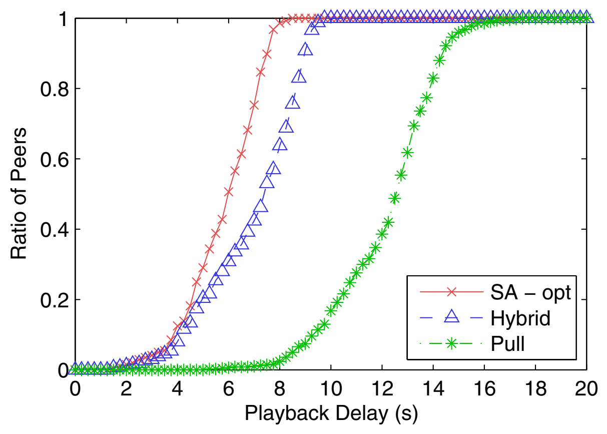

Fig. 5: Playback delay distribution.

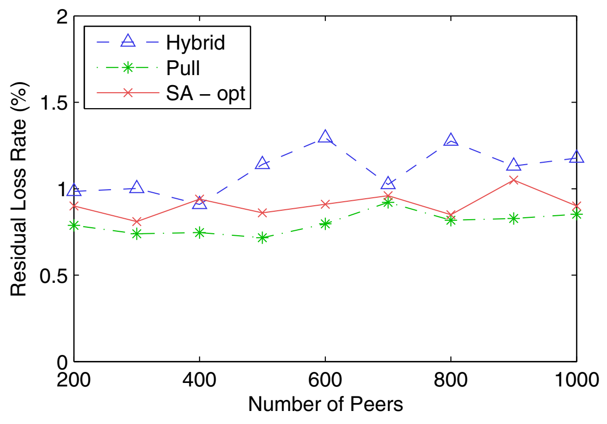

Fig. 6: Residual loss rate versus number of peers.

and hybrid are comparatively higher (near 10%). In pull, every packet is pulled explicitly so data packets are seldom redundant. The transmission loss and control overhead account for the bandwidth dilation in pull. In our scheme and the hybrid approach, redundant packets are possible because of asynchronization between push and pull, due to propagation delay of control signal. Although the pull requests are greatly reduced in our scheme and the hybrid approach, the redundant packets still result in slightly higher bandwidth dilation.

V. Conclusions and Future Work

In this paper, we propose a scheduling algorithm for P2P live streaming with low delay. Our scheme pushes video packets in substreams and recovers from packet loss using pulling. Our scheme can work on arbitrary overlay to achieve low delay scheduling, in the presence of packet transmission loss.

In order to deliver a major part of packets quickly to the child, packets are pushed in substreams from parents. Given heterogeneous contents, delays and bandwidths of parents, we formulate the Substream Assignment (SA) problem to assign substreams to parents to achieve minimum delay. The SA problem can be optimally solved in polynomial time

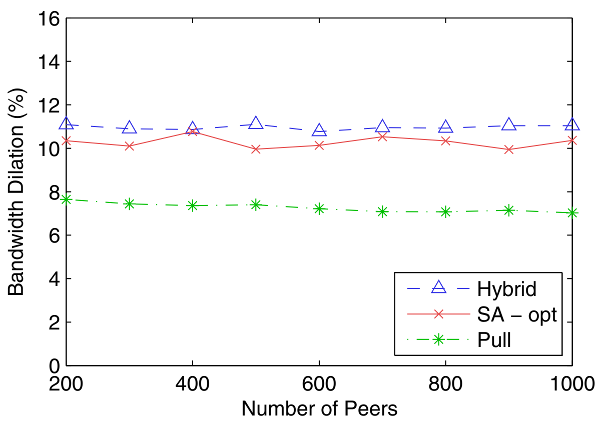

Fig. 7: Bandwidth dilation versus number of peers.

by transforming it into a Max-Weighted Bipartite Matching problem. Our simulation results show that our scheme achieves substantially lower delay, as compared with the current pull and hybrid pull-push approaches.

We discovered that the pulling, which is initiated at child, incurs some control delays. It would be good if part of the recovery process of lost packets can start earlier. In the future, we would like to further reduce the delays by studying the use of other techniques, possibly such as network coding [12].

References

[1] Nazanin Magharei, Reza Rejaie, and Yang Guo, "Mesh or multiple-tree: A comparative study of live P2P streaming approaches," in *Proc. IEEE INFOCOM*, 2007, pp. 1424–1432.
[2] Xinyan Zhang, Jiangchuan Liu, Bo Li, and Tak-Shing Peter Yum, "Coolstreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming," in *Proc. IEEE INFOCOM*, 2005, pp. 2102–2111.
[3] Nazanin Magharei and Reza Rejaie, "PRIME: Peer-to-peer receiver-driven MEsh-based streaming," in *Proc. IEEE INFOCOM*, 2007, pp. 1415–1423.
[4] Bo Li, Susu Xie, Gabriel Y. Keung, Jiangchuan Liu, Ion Stoica, Hui Zhang, and Xinyan Zhang, "An empirical study of the coolstreaming+ system," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 9, pp. 1627–1639, 2007.
[5] Meng Zhang, Qian Zhang, Lifeng Sun, and Shiqiang Yang, "Understanding the power of pull-based streaming protocol: Can we do better?," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 9, pp. 1678–1694, 2007.
[6] Meng Zhang, Lifeng Sun, Yechang Fang, and Shiqiang Yang, "iGridMedia: The system to provide low delay peer-to-peer live streaming service over internet," *Peer-to-Peer Networking and Applications*.
[7] Z. Liu, Y. Shen, K. W. Ross, S. Panwar, and Y. Wang, "Substream trading: Towards an open p2p live streaming system," in *Proc. IEEE ICNP*, 2008.
[8] Kin-Wah Kwong and H. K. Tsang, "Building heterogeneous peer-to-peer networks: protocol and analysis," *IEEE/ACM Transactions on Networking*, vol. 16, no. 2, pp. 281–292, 2008.
[9] Dongni Ren, Y.-T.H. Li, and S.-H. Chan, "On reducing mesh delay for peer-to-peer live streaming," in *Proc. IEEE INFOCOM*, 2008, pp. 1058–1066.
[10] Christos H. Papadimitriou and Kenneth Steiglitz, *Combinatorial optimization : algorithms and complexity*, Prentice Hall, 1982.
[11] Alberto Medina, Anukool Lakhina, Ibrahim Matta, and John W. Byers, "BRITE: An approach to universal topology generation," in *Proc IEEE MASCOTS*, 2001, p. 346.
[12] Chen Feng and Baochun Li, "On large-scale peer-to-peer streaming systems with network coding," in *ACM Multimedia*, 2008, pp. 269–278.