Coolstreaming: Design, Theory, and Practice

Susu Xie, Bo Li, *Senior Member, IEEE*, Gabriel Y. Keung, and Xinyan Zhang, *Student Member, IEEE*

***Abstract*—Peer-to-peer (P2P) technology has found much success in applications like file distributions and VoIP yet, its adoption in live video streaming remains as an elusive goal. Our recent success in Coolstreaming system brings promises in this direction; however, it also reveals that there exist many practical engineering problems in real live streaming systems over the Internet. Our focus in this paper is on a nonoptimal real working system, in which we illustrate a set of existing practical problems and how they could be handled. We believe this is essential in providing the basic understanding of P2P streaming systems.**

This paper uses a set of real traces and attempts to develop some theoretical basis to demonstrate that a random peer partnership selection with a hybrid pull-push scheme has the potentially to scale. Specifically, first, we describe the fundamental system design tradeoffs and key changes in the design of a Coolstreaming system including substreaming, buffer management, scheduling and the adopt of a hybrid pull-push mechanism over the original pull-based content delivery approach; second, we examine the overlay topology and its convergence; third, using a combination of real traces and analysis, we quantitatively provide the insights on how the buffering technique resolves the problems associated with dynamics and heterogeneity; fourth, we show how substream and path diversity can help to alleviate the impact from congestion and churns; fifth, we discuss the system scalability and limitations.

***Index Terms*—IPTV, measurement, peer-to-peer technology, video streaming.**

I. Introduction

THERE have been significant studies and numerous technical innovations for live video streaming, in which the recent development has been focusing on application layer multicast or overlay multicast, in which the key idea is to push multicast functionalities to the edges of networks [1]–[3]. It was argued that the native IP multicast encountered practically unsolvable problems and the adoption of some forms of end application multicast to bypass the need from infrastructure support has been proposed. In a typical overlay multicast architecture, end systems self-organize themselves into an overlay topology and data forwarding follows the links of the overlay. On the other hand, Coolstreaming, a *data-driven* scheme, is a completely different approach [4]. The key novelties are 1) peers gossip with one another for content availability information and 2) peers use a swarm-like technique, somewhat similar to the technique used in BitTorrent, for content delivery. We make a distinction in this paper by referring this as a peer-to-peer (P2P) live video streaming system, in which there is no explicit overlay topology construction. This was referred to as a treeless approach or swarming in [5]. We refer to the other approaches as overlay multicast, given the explicit construction of multicast tree(s).

Coolstreaming represented one of the earliest large-scale P2P video streaming experiments [4], [6], which has been widely referenced in the community as the benchmark (Google entries tops 300 000) that a P2P-based live streaming system has the potential to scale. Since its first release, while keeping the random partner selection, we have enhanced the system in nearly all aspects, specifically 1) the initial system adopted a simple pull-based scheme for content delivery based on content availability information exchange using buffer-map. This incurs per block overhead, and more importantly, it results in a longer delay in retrieving the video content. We have now implemented a hybrid pull and push mechanism, in which the video content is pushed by a parent node to a child node except for the first block. This lowers the overhead associated with each video block transmission, reduces the initial delay and increases the video playback quality; 2) a novel multiple substream scheme is implemented, which essentially enables multisource and multipath delivery for video streams. Observed from the results, this not only enhances the video playback quality and but also improves the effectiveness against system dynamics; 3) the gossip protocol was enhanced to handle the push function; 4) the buffer management and scheduling schemes are redesigned to deal with the dissemination of multiple substreams.

There are many issues relevant to the design of live video streaming systems such as system dynamics, heterogeneity, churn, network congestion, stream synchronization, multipath routing, topology stability and convergence [7]–[9]. There have been investigations on optimal resource allocation [10], scaling factors [11], incentive-based mechanism [12], fine-granularity control [5], priority based multicast [13], integration with network encoding technique [14]. However, there seems to be several misconceptions in the most basic understanding of a live video streaming system. This paper takes a different approach by contributing to this basic understanding, we will: 1) describe the key issues in a real working system; 2) closely examine a set of existing practical problems and how they could be handled in a real system; 3) focus the discussions on system dynamics and how it affects the overall performance.

Recall one of the fundamental principles in the Internet is the simplicity in its core functionalities. Keeping this simplicity in mind, our discussions in this paper focus on a reasonably scale nonoptimal working system. The purpose of this paper

Manuscript received November 16, 2006; revised July 29, 2007. This work was supported in part by grants from the RGC under the Contracts 616505, 616406, and 616207, by a grant from HKUST under Contract RPC06/07.EG27, by grants from NSF China under the Contracts 60429202 and 60573115, and by a grant from National Basic Research Program of China (973 Program) under Contract 2006CB303000. The associate editor coordinating the review of this manuscripts and approving it for publication was Prof. Klara Nahrstedt.

S. Xie, B. Li, and G. Y. Keung are with the Department of Computer Science, Hong Kong University of Science and Technology, Clearwater Bay, Hong Kong (e-mail: xiesusu@cse.ust.hk; bli@cse.ust.hk; phgab@cse.ust.hk).

X. Zhang is with the Roxbeam Corporation, Beijing, China (e-mail: tom.zhang@roxbeam.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2007.907469

is to demonstrate concretely how a working system resolves some of these issues and a set of realistic engineering problems that need to be addressed. We leverage a large set of traces we recently obtained from a Coolstreaming experiment conducted on Oct 26, 2006, and attempt to develop some theoretical basis to provide the intuitions and our understanding on how and why such a system works. Specifically, 1) based on a simple topology model, we illustrate how random peer selection can lead to topology convergence; 2) using a set of real traces and analysis, we quantitatively provide the insights on how the buffering technique can resolve the problems associated with dynamics and heterogeneity; 3) we show how substream and path diversity can help to alleviate the impact from network congestion and peer churns; 4) we discuss the scalability and limitations of a P2P based live video streaming system.

The rest of this paper is organized as follows. Section II reviews the related works. Section III describes the Coolstreaming system and discusses the key tradeoffs in the system's design. Section IV examines peer dynamics, and investigates the overlay convergence and system scalability. Section V concludes the paper.

II. Related Work

The existing works on P2P live video streaming can generally be classified into two categories: overlay multicast and data-driven approaches. In an overlay multicast approach, the key idea is to build a multicast tree at the application end. The end system multicast is one of the first protocols [2], in which it examines the key difficulties encountered in the native IP multicast and proposes a new mechanism by moving the multicast functionalities to the edge of the networks. A single-tree overlay suffers two drawbacks: 1) unfair contributions in that leaf nodes do not contribute upload capacity, which leads to poor resource utilization; 2) given the dynamic nature of nodes, the departure or failure of a high-level node can cause significant disruption and requires the reconstruction of the overlay topology. The multitree approach has been proposed to cope with this problem [15], [16]. This usually requires that the source encodes a video stream into multiple substreams and each substream is distributed over a separate overlay tree. These result in two improvements, 1) better resilience against churn and 2) fair bandwidth utilization from all peer nodes. However, multitree based systems demand the use of special multirate or/and multilayer encoding algorithms, which has not been demonstrated feasible in real systems over the Internet. Further, those systems still have to deal with the reconstruction of the multitrees in the presence of network dynamics and potentially restrictive contributions from leaf nodes. Perhaps most significantly, there has been no large scale real systems demonstrated using such an approach.

There have been several recent works, such as Chunkyspread [5], which focused on the efficiency associated with multitree construction and also explored the simplicity and scalability adopted from unstructured networks. Specifically, a randomized multitree was constructed to spread the slices of video stream, and the topology could be improved iteratively by swapping parents with respect to load and delay measurement [17]. Rao *et al.* evaluated the multitree framework and proposed a new concept called *contribution awareness* as an incentive to enable better contribution from peer nodes [18]. Reza *et al.* proposed to use the swarming technique to enable leaf nodes to contribute in uploading capacity [19].

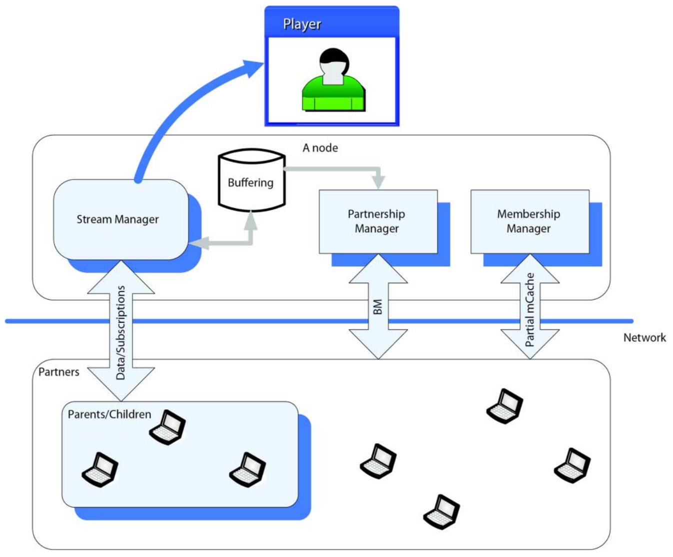


Fig. 1. Coolstreaming system diagram.

Within the framework of P2P live streaming, there are several fundamental issues that need to be resolved such as how an overlay topology evolves under a random peer selection scheme, under what condition can an overlay be stabilized, how can the system be scalable and what are the fundamental limitations and tradeoffs. None of the existing works have provided a comprehensive study on these important issues. To the best of our knowledge, this paper represents the first attempt to address these problems from a practical point of view. In this study, we demonstrate the real engineering problems we have encountered; concretely describe how those problems are resolved with various mechanisms; and how different components interact with one another. Further, we provide analysis whenever necessary and our intuitions behind the system design.

III. Detail of Coolstreaming

The original Coolstreaming system was developed in early 2004. Since the first release, it has attracted millions of downloads worldwide. The peak concurrent users reached over 80 000 with an average bit rate of 400 Kbps with clear global presence [6].

A. Basic Components

Fig. 1 depicts the current system design for Coolstreaming. There are three basic modules in the system: 1) *Membership manager*, which maintains partial view of the overlay. 2) *Partnership manager*, which establishes and maintains TCP connections, or partnership, with other peer nodes. It also exchanges the availability of stream data in the *buffer map* (BM) with the peer nodes, which we will explain later. 3) *Stream manager*, which is the key component for data delivery. Besides providing stream data to the media player, it also makes decisions on where and how to retrieve stream data. The central design in this system is based on the *data-driven* notion, in which every peer node periodically exchanges its data availability information with a set of

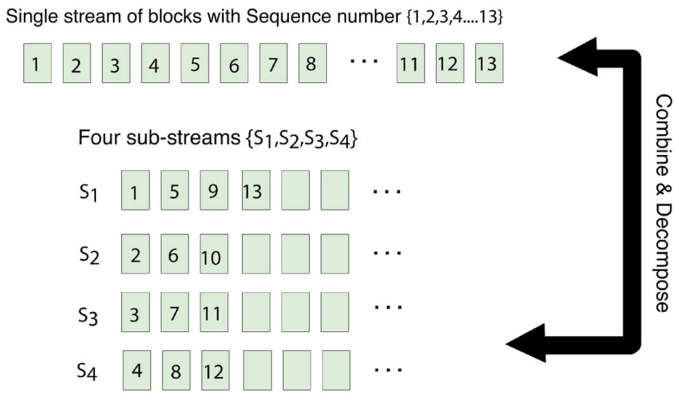

Single stream of blocks with Sequence number {1,2,3,4....13}
Four sub-streams {S1,S2,S3,S4}
S1
S2
S3
S4
Combine & Decompose

Fig. 2. Example of stream decomposition.

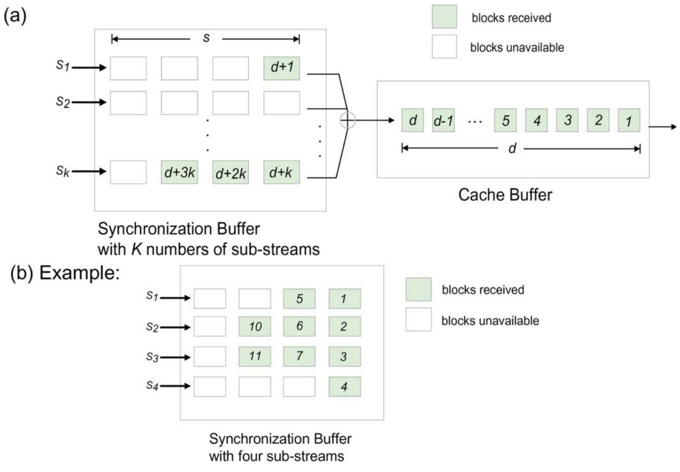

(a)
s
S1
S2
Sk
d+1
d+3k
d+2k
d+k
Synchronization Buffer with K numbers of sub-streams
Cache Buffer
d
d-1
blocks received
blocks unavailable
(b) Example:
Synchronization Buffer with four sub-streams

Fig. 3. (a) Structure of buffers in a node. (b) An example of combination process in Synchronization buffer.

partners to retrieve unavailable data, while also supplying available data to others.

B. Multiple Substreams

The video stream is divided into *blocks* with equal sizes, and each block is assigned a sequence number to represent its playback order in the stream. Since it is a live video streaming and we use TCP for transmissions, the sequence number also serves as a timestamp, which can be used to combine and reorder the blocks after reception.

One of the key factors contributing to the success in P2P file-sharing applications is the adoption of the gossip concept, in which a node can request different small chunks of file content from different nodes. This achieves significantly higher efficiency compared to other traditional systems [20]. This is also adopted in Coolstreaming. Specifically, a video stream is divided into multiple *substreams*; nodes could subscribe to different substreams from different partners. This, however, is nontrivial for video streaming given the synchronization and continuity requirements, which will be discussed in Section III-C.

In the Coolstreaming system, there is an important distinction between *parent-children relationship* and *partnership*. Partnership is established between two peer nodes with the TCP connection. This enables two peers to exchange block availability information. A parent-children relationship is established when a node (the child) is actually receiving substream(s) from another node (the parent). It is common in Coolstreaming that a parent node is sending multiple substreams to a child node. Here we define a substream degree of node p, denoted as D_p, which is the total number of substreams sent by node p. A stream is decomposed into K substreams by grouping video blocks according to the following scheme: the i-th substream contains blocks with sequence numbers $(nK+i)$, where n is a positive integer from zero to infinity, and i is a positive integer from 1 to K. This implies that a node can at most receive substreams from K parent nodes. An example is given in Fig. 2, in which the video stream is divided into four substreams.

C. Buffering

A *buffer map* or BM is introduced to represent the availability of the latest blocks of different substreams in the buffer. This information also has to be exchanged periodically among partners in order to determine which substream to subscribe to. The detailed structure of the buffer map is as follows: BM is represented by a series of $2K$-tuples, where K is the number of substreams. The first series of K-tuples records the sequence number of the latest received block from each substream; the substreams are specified by $\{S_1, S_2, \ldots, S_K\}$ and the corresponding sequence number of the latest received block given by $\{H_{S_1}, H_{S_2}, \ldots, H_{S_K}\}$. For example, $\{2K+1, 3K+2, 2K+3, \ldots, 4K\}$ implies that a node receives blocks up to the sequence number "$2K+1$" from the first substream, up to the sequence number "$3K+2$" from the second substream and so on. The second series of K-tuples represents the subscription of substreams from the partner. For example, if node A is requesting the subscription of the first and second substreams from node B, it sends out a message $\{1,1,0,0,\ldots,0\}$ to node B.

Each node maintains an internal buffer, whose structure is illustrated in Fig. 3(a). It is composed of two parts: *synchronization buffer* and *cache buffer*. A received block is first put into the synchronization buffer for each corresponding substream. The blocks will be stored in chronological order in the synchronization buffer according to their sequence numbers. They will be combined into one stream when blocks with continuous sequence numbers have been received from each substream. An example is given in Fig. 3(b), in which the combination process stops at the third substream while it awaits the block with sequence number 8.

D. Overlay Construction

A *membership manager* is an essential module for overlay construction and maintenance. Each node in the system has an unique identifier and maintains a membership cache (mCache) containing a partial list of the currently active nodes in the system. A node uses this information to establish TCP connections with one another. Each node in the mCache is considered as a *member* in the overlay topology. The system consists of an origin or source node, a boot-strap node and member nodes, where the *boot-strap* node serves as the entry point and the source node provides continuous live video streaming.

The gossip protocol is used for overlay construction. Such a technique has been widely used in BitTorrent and other P2P systems [9], [20]–[22], which achieves excellent resilience against random failures and enables decentralized operation [4]. In Coolstreaming, a newly joined node contacts the boot-strap node for a list of peer nodes and stores that in its own mCache. The boot-strap node performs two simple operations: 1) providing a randomized list of the currently active peer nodes to the newly joined node; and 2) possibly updating its mCache by including the newly joined node. The newly joined node stores this list in its own mCache. Based on the contact list of nodes in its mCache, the node randomly selects a few nodes to establish TCP connections, i.e., partnership. Once the partnership is established between a pair of nodes, they will exchange their mCache contents. This is only executed when the partnership is first established. The maximum number of entries in each mCache is determined by a system parameter M. Given the size of mCache is finite, the exchange of mCache can possibly cause a node to remove some entries from its mCache. This is necessary so that each node can periodically update its mCache entry to maintain a list of currently active nodes in the system.

Partnership can be broken from time to time due to many reasons and nodes will need to perform *partner reselection* to maintain the continuity of video streams. For example, this occurs when a node is receiving insufficient bit rates from its partners, in which case the node has to drop some partners and reestablish new partnership with other peers. There are other factors that can cause partnership disconnection such as node departure and the impact from NAT and firewall. A node once recognizing the unavailability of partner nodes will simply remove the entries from its mCache. Such information, however, will not be flooded immediately in the system. Instead, the entries of the unavailable nodes will gradually and gracefully fade out from the mCache over a period of time with the periodical exchange of mCache entries among peer nodes. Consequentially, this reduces the control messages and also helps to maintain the mCache up-to-date.

E. Content Delivery

In the actual substream transmission, Coolstreaming adopts a hybrid *push and pull* scheme. When a node subscribes to a substream by connecting to one of its partners via a single request (pull) in BM, the requested partner, i.e., the parent node, will continue pushing all blocks in need of the substream to the requested node. In the Coolstreaming system, a parent node will not voluntarily drop a child node, so it is up to the children to dynamically monitor the in-coming connections and trigger any parent reselection if necessary. This will be further elaborated in the next section.

With the exchange of BM information, the newly joined node can obtain the availability of blocks from its parent nodes. Next the node has to determine from which part of the stream should the newly joined node request the video stream; in other words, it needs to determine the initial sequence number of the block that the node will start to retrieve. This turns out to be a nontrivial problem. Suppose the range of the blocks available in all its partners are from n to m, intuitively, the node should request from a block with sequence number somewhere in the middle. The rationale for this is if the node requests the block starting from the largest sequence number m, the partner nodes might not have sufficient follow-up blocks to satisfy the continuity requirement for the video stream; on the other hand, if the node requests the block from the lowest sequence number n, this can result in two problems: 1) such blocks might no longer be available once it is pushed out of the partners' buffer due to the playout; 2) it might also take considerable amount of time for the newly joined node to catch up with the current video stream as depicted in (3) later, which would incur long initial delay. From the above argument, in the Coolstreaming, by examining partners' BM, the node finds the largest sequence number m of the received blocks among its partners. The newly joined node will start to subscribe from a block shifted by a system parameter T_p, which will be introduced in peer adaptation in the next section.

Once the initial sequence number is determined, the node checks the availability of blocks in its partners' BM and then selects appropriate partner nodes as its parents for each substream. This process is essentially the same as when a node performs *peer adaptation*. Parent nodes continue pushing the available blocks of substreams to the newly joined node. The node stores the received blocks from substreams in the corresponding synchronization buffer. These will be combined later to form a single continuous stream to the cache buffer.

TABLE I
SYSTEM PARAMETERS OF COOLSTREAMING

O	the source of the live video stream
R	bit rate of the live video stream
K	number of sub-streams
B	length of a peer's buffer in unit of time
T_s	out-of-synchronization threshold, i.e. upper bound of acceptable deviation between sub-streams
T_p	maximum allowable latency for a partner behind others
T_a	the period a peer re-select a parent if needed
M	upper bound on the number of partners

IV. System Dynamics

Understanding the system dynamics is crucial in a P2P streaming system. In this section, we focus our discussions on peer adaptation and system dynamics. Table I summarizes the parameters and notations.

A. Peer Adaptation

Since congestion and peer churns occur frequently in the Internet, it is important for any P2P system to do peer adaptation, i.e., to search for new parent(s). In simple terms, a peer node needs to constantly monitor the status of the on-going substream transmissions and reselects new parents when existing TCP connections are inadequate in satisfying the streaming requirement. The questions are what triggers this adaptation, i.e., how to detect possible congestion or churns, and how to reselect new parent(s).

The status of insufficient upload capacity could be detected by the following metric: First, we introduce two thresholds $\{T_s, T_p\}$, which are related to the sequence number of blocks for different substreams in each node (say, node A). T_s can be interpreted as the threshold of the maximum sequence number deviation allowed between the latest received blocks in any two substreams in node A. T_p is defined as the threshold of the maximum sequence number deviation of the latest received blocks between the partners and the parents of node A. We denote $H_{S_i,A}$ as the sequence number of the latest received block for substream S_i at node A. For monitoring the service of substream j by corresponding parent p, two inequalities can be introduced

$$\max\{|H_{S_i,A} - H_{S_j,p}| : i \leq K\} < T_s \quad (1)$$
$$\max\{H_{S_i,q} : i \leq K, q \in \text{parnters}\} - H_{S_j,p} < T_p. \quad (2)$$

Inequality (1) is used to monitor the buffer status of received substreams for node A. If this inequality does not hold, it implies that at least one substream is delayed beyond threshold value T_s. This is an indication for either congestion or insufficient upload capacity for this substream, which will subsequently trigger peer adaptation. The second Inequality (2) is used to monitor the buffer status in the parents of node A. Node A compares the buffer status of current parents to that of its partners. If this inequality does not hold, it implies that the parent node is considerably lagging behind in the number of blocks received when comparing to at least one of the partners, which currently is not a parent node for the given node A. This will also trigger node A to perform peer adaptation by switching to a new parent node from its partners.

When a peer selects a new parent from its partners, the selected partner must satisfy the two inequalities. If there is more than one qualified partners, the peer will choose one of them randomly. A parent node, however, will always accept requests and it will simply push out all blocks of a substream in need to the requesting node. Apparently such a peer adaptation can potentially cause stream disruption and instability of the overlay topology. A cool-down timer is introduced to confine nodes to perform peer adaptation once only within a cool-down period of time T_a.

Since the selection of a new parent is based on Inequalities (1) and (2), here we only consider the sequence numbers instead of the parent's available upload bandwidth. In that case, a parent with insufficient upload bandwidth can be selected, and its upload capacity can be insufficient to support all children nodes. This is because the parent will continue accepting new children as long as its total number of partners is less than the upper bound M introduced earlier. In this case, all children nodes have to compete for the insufficiently aggregated upload capacity from the parent. We call this situation *peer competition*, in which eventually one or more nodes will lose and trigger peer adaptation. This causes a chain reaction of peer adaptations. During the process, the inequalities can be violated and some temporary parents may be selected and abandoned before a capable parent is located.

B. Peer Dynamics

Since peer adaptation is used to deal with congestion and peer churn, it is necessary to study how system dynamics evolves and the impacts from system parameters of relevance.

First, we will discuss the relative timing involved in the peer adaptation process. The average bit rate required for one single substream transmission is R/K. Consider the case that node p (the parent) is pushing data to node q (the child). Suppose that node p is capable of providing upload bandwidth (i.e., bit rate) $r_\uparrow > R/K$, in this case node q can eventually catch up with node p in terms of the missing blocks from a substream. This can be referred to as a *catch up process*. Let us assume initially there are l blocks missing from node q comparing to node p, then time $t_\uparrow$ for the catch up process can be easily computed by

$$\begin{aligned} r_\uparrow \cdot t_\uparrow &= R/K \cdot t_\uparrow + l \\ t_\uparrow &= \frac{l}{r_\uparrow - R/K}. \end{aligned} \quad (3)$$

Now, let us examine the chain reaction of peer adaptation. Suppose the parent of node p is incapable of supporting its children nodes including p; p loses the competition and if p cannot find a new capable parent fast enough, its children (such as node q) can be stalled. The time, denoted as $t_\downarrow$, for a child q of parent p to abandon p as its parent, i.e., the substream obtained from node p in node q lags behind other substreams beyond threshold T_s. In other words, if p can find a new capable parent within time $t_\downarrow$, there is no need for node q to perform peer adaptation.

A node can still receive blocks from those temporary parents, suppose with an average bit rate $r_\downarrow$, and $r_\downarrow < R/K$. Then time $t_\downarrow$ can be computed by

$$\begin{aligned} t_\downarrow \cdot r_\downarrow + l &= t_\downarrow \cdot R/K \\ t_\downarrow &= \frac{l}{R/K - r_\downarrow}. \end{aligned} \quad (4)$$

Suppose the substream degree of p is D_p when q is accepted as a child, and p can satisfy all its children before q's subscription. After q is accepted, the upload bandwidth for each substream transmission of p decreases from R/K down to $r_\downarrow$, where

$$r_\downarrow = \frac{D_p}{D_p + 1} \cdot R/K. \quad (5)$$

The result of competition depends on the buffer status of the children nodes at the beginning of the competition. Suppose it takes t_{lose} time for one of the children to lose the competition due to a subscribed substream lagging behind others from t_δ to T_s in unit of blocks, i.e., it violates Inequality (1). We have

$$\begin{aligned} R/K \cdot t_{\text{lose}} - \frac{D_p}{D_p + 1} \cdot R/K \cdot t_{\text{lose}} &= (T_s - t_\delta) \\ t_{\text{lose}} &= \frac{(D_p + 1)(T_s - t_\delta)}{R/K}. \end{aligned}$$

As discussed earlier, the system confines nodes to perform peer adaptation once only in time period T_a. Node q will unsubscribe from p if there is no other children of p losing the competition within T_a. Thus, we can calculate the probability for a child to lose the competition by

$$\begin{aligned} P(t_{\text{lose}} \leq T_a) &= P\left(\frac{(D_p + 1) \cdot (T_s - t_\delta)}{R/K} \leq T_a\right) \\ &= P\left(t_\delta \geq T_s - \frac{T_a \cdot R/K}{D_p + 1}\right). \end{aligned} \quad (6)$$

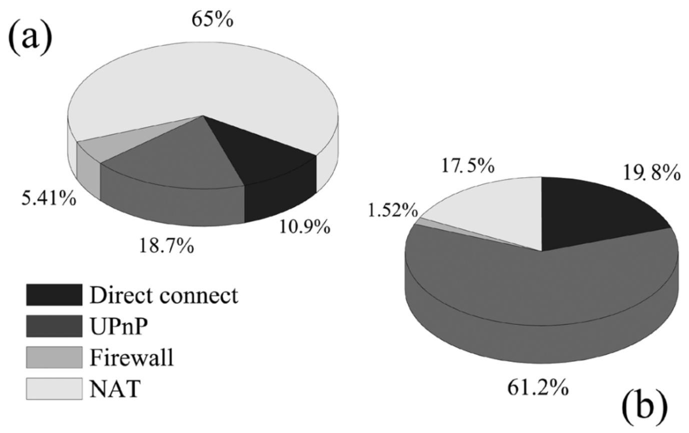


Fig. 4. (a) User Type Distribution; (b) User Contribution Distribution.

V. Results and Discussions

In this section, we present the results obtained from a live streaming event using Coolstreaming on 27 September, 2006. Specifically, we will examine the overlay topology, user distributions, video playback quality and sensitivity against system size, user types and system dynamics. Finally, we will discuss the issues related to scalability.

A. Overlay Topology and Peer Classifications

The Coolstreaming system employs a lightweight protocol that is based only on the local information received from each user. At the high level, based on their IP addresses, users can be classified as private or public users. By checking whether we are successful to open various TCP connections, we can further classify users into the following four types.

- *Direct-connect*: peers have public addresses with both incoming partners and outgoing partners;
- *UPnP*: peers have private addresses with both incoming partners and outgoing partners. Actually, peers can be aware of UPnP devices in the network since they will explicitly acquire public IP addresses from the devices;
- *NAT*: peers have private addresses with only outgoing partners;
- *Firewall*: peers have public addresses with only outgoing partners.

As defined earlier, a partnership represents a connection between two users, which is used to exchange video availability information. Here, we use outgoing partners to illustrate the fact that it is only feasible for NAT or firewall users to initialize the partnership establishment, not the other way around. However, once a NAT or firewall user establishes a partnership with another node, it can still upload video to other node whenever it is capable of doing so. In other words, a NAT or firewall user can still become the parent for another peer node.

We illustrate the user distribution according to their categories and the upload contributions in Fig. 4. Observed from the figure we can see uneven contributions from peer nodes in the P2P streaming system. Specifically, 30% or so peer nodes in the overlay, i.e., nodes under UPnP and direct-connect, contribute more than 80% of the upload bandwidth. This is consistent with the results obtained in [23].

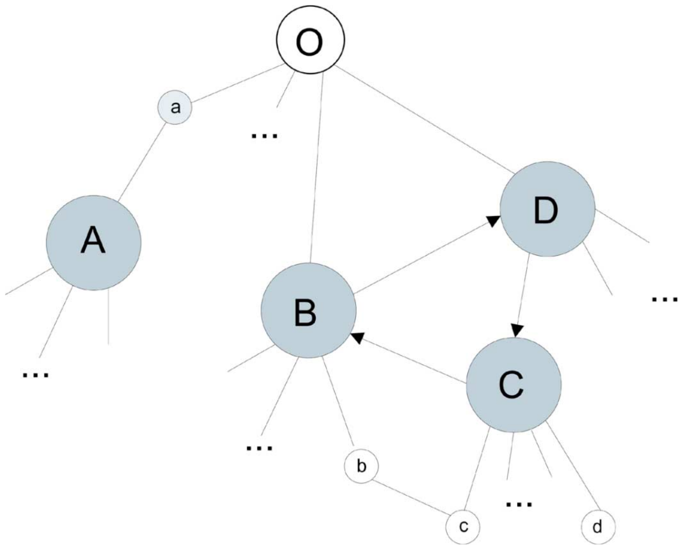


Fig. 5. Conceptual view of live video streaming Overlay.

It is usually difficult to capture the exact snapshot of the overlay topology in a real system. Based on the above observations, we conjecture a conceptual overlay in Fig. 5. Node O is the source node, peers $\{A; B; C; D\}$ are direct-connect/UPnP nodes and others are nodes behind NAT and firewall. The overlay is illustrated as a mesh structure, but generally resembles a tree-like topology with a few random links between selected peers (this will be further explained in Section V-A1). Each link represents one or more substream transmissions. We next discuss peer classification and the basic property of the overlay topology.

1) Peer Classification: Some of the direct-connect/UPnP peers often stay in the overlay for a long time and have relatively large amount of upload bandwidth. Children of direct-connect/UPnP peers can obtain sufficient upload bandwidth for substream transmissions, thus can be fairly stable. As we will describe in the next subsection, there are usually less chances for peer competition among the children belonging to direct-connect/UPnP peers. The degree of direct-connect/UPnP peers often reaches the maximum allowed number by the system. Peers $\{A; B; C; D\}$ are such direct-connect/UPnP peers in Fig. 5.

Some of NAT or firewall users also tend to stay in the overlay for a long time but have only limited upload bandwidth. The contribution to the overlay is restricted and their children suffer from peer competitions. However, they could stay close to the source. Peer $\{a\}$ is one of such peers.

The remaining peers are those that usually stay in the system for a short period of time, often with very little upload capacity. Most of them are NAT/firewall peers. They are often positioned far from the source and have to reselect parents relatively often. Peers $\{b; c; d\}$ are such peers in the figure.

2) Overlay Structure: There are several interesting properties in the conceptual view of the overlay in Fig. 5.

- Large amount of peers tends to clog under direct-connect/UPnP peers;
- Connections among NAT/firewall peers (we refer to as random links), i.e., b and c in the figure, are relatively rare.

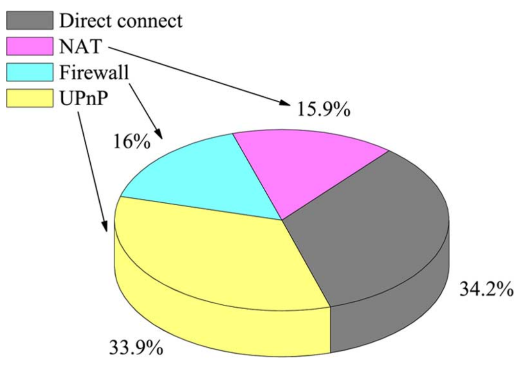


Fig. 6. Distribution of changing partnership.

We next explain the rationale behind these properties. The overlay of the streaming system is subject to change, i.e., peer adaptations occur from time to time. A peer node tends to be stable under two conditions. First, the parent node can provide sufficient upload bandwidth and the parent node itself is stable. Second, according to (6), the larger the substream degree of the parent, the lesser the probability that the children will lose when peer competition happens; in other words, if a node subscribes to a NAT/firewall parent, whose substream degree is often relatively small, the node tends to be more vulnerable. Usually, even if a peer selects a NAT/firewall peers as the parent at the beginning, as it suffers from insufficient upload bandwidth and is frequently subject to peer adaptation, eventually it can convert to a direct-connect/UPnP peer for its parent.

The stability of a direct-connect/UPnP peer also explains why peers tend to clog under direct-connect/UPnP peers. For example, if a peer selects a NAT/firewall peer as the parent, like $\{A; a\}$ in Fig. 5, such subscription can be easily broken due to the insufficient upload bandwidth from node a. The peer will eventually be stabilized under some direct-connect/UPnP peers. If the system runs long enough, most of the peers will likely become children of direct-connect/UPnP peers.

Fig. 6 shows the distribution of changing partnership for different user types from 12:00 to 23:59, and direct-connect/UPnP peers experienced higher partnership change rates than NAT/firewall peers. Peers tend to clog under direct-connect/UPnP peers, however during peak hours, some direct-connect/UPnP peers are incapable to stream additional children. Consequently, those children peers perform peer adaptations and break the partnership with their incapable parents.

A natural conclusion can be drawn from this overlay structure is that the video quality perceived by nodes heavily depends on that perceived by direct-connect/UPnP peers. The existence of direct-connect/UPnP peers in the P2P streaming system is not only critical for overlay stability, but also helps to reduce latency. As discussed earlier (see (4)), the catch-up process can be significantly speeded up with the selection of a direct-connect/UPnP peer as a parent node. Another implication is that the source needs to be placed in a node with sufficient upload capacity; otherwise frequent peer adaptations can occur and this makes the overall system unstable.

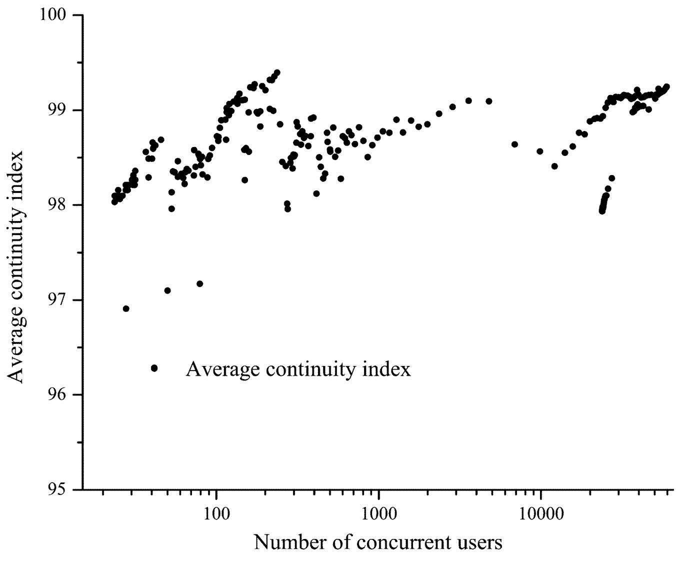


Fig. 7. Average continuity index against system size.

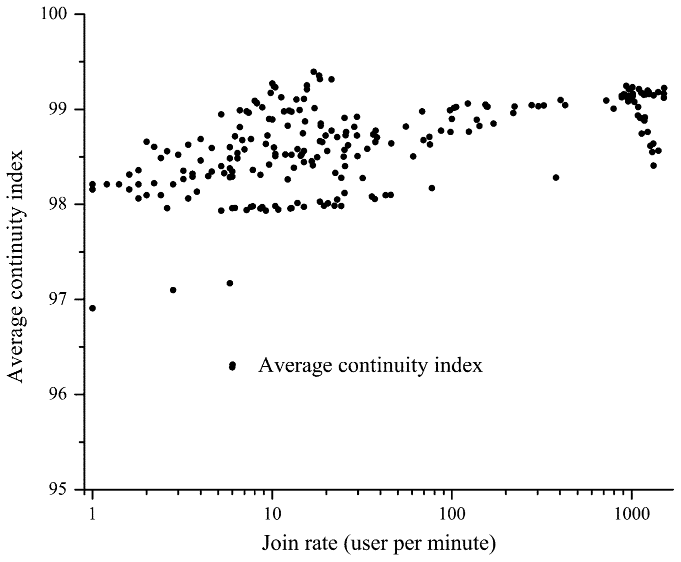


Fig. 8. Average continuity index against join rate.

B. Convergence of the Overlay

It is a general belief that the performance of a P2P streaming system is heavily dependent on the self-convergence capability of the overlay topology, i.e., the ability to self-evolve into a stable topology. This is important in dealing with congestion and churns. Figs. 7 and 8 show the average continuity index defined in [4], which is the number of blocks that arrive before playback deadlines over the total number of blocks. It maintains at a high level (95% or so) even with increasing system size and join rate. Both figures show excellent self-convergence capability during large system size (large number of concurrent users) and high join rate. We next closely examine how this self-evolution occurs, and how the overlay topology migrates.

We first define that a node is stable if it receives adequate bit rates for all substreams. We next define the *self-convergence* property. When congestion or churn occurs, a stable node can become unstable. Self-convergence can be defined as the property that such unstable nodes will eventually become stable by selecting new capable parents via peer adaptation. However, it is important to point out that it is not an inherent property that a P2P streaming system is self-convergence. We next closely examine the factors that impact this property.

In a P2P streaming system, it is possible that a chain reaction in peer adaptations could occur. As we described earlier, once node q becomes unstable, it will perform peer adaptation to find a new parent p, which has to satisfy the criteria from (1) and (2). The overlay can be affected by three possibilities during a peer adaptation.

A–(*adaptation absorption*): p has enough free upload bandwidth over R/K. Consequentially, the overlay turns into a new stable state.

B–(*adaptation relay*): p does not have sufficiently free upload bandwidth to serve q. q will then compete for upload bandwidth with p's other children. If q wins the competition, q will remain stable and the losing peer will have to perform peer adaptation.

C–(*adaptation amplification*): If q cannot find a capable parent in time, or q continuously loses in the competitions, q's children will then become unstable. The number of unstable peers in the overlay is hence amplified to be D_q+1, where D_q is the substream degree of q.

We can model peer adaptation by a *continuous time branching process* [24]. Since we are only concerned with whether the overlay will converge to a stable state eventually, we consider the corresponding *embedded generation process*, which counts the numbers of stable and unstable nodes in each generation. To simplify our discussion, we assume q selects one parent p when peer adaptation occurs. Under this assumption, the reproduction rate ξ_q of an unstable peer q and expected value of $\xi_q(E(\xi_q))$ can then be computed by

$$\begin{aligned} \xi_q &= 0\cdot P(A) + P(B) + (D_q+1)\cdot P(C) \\ &= 1 - P(A) + D_q \cdot P(C) \\ E(\xi_q) &= 1 - P(A) + E(D_q)\cdot P(C) \end{aligned} \tag{7}$$

where $P(A), P(B)$ and $P(C)$ are the probabilities that events $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ happen, respectively. Base on the theory of branching process [24], if $E(\xi)$ is less than 1, the population of unstable peer eventually becomes extinct after several generation steps, which is called *subcritical* branching process. We now have the theorem on self-convergence:

Theorem 1: An overlay will be of self-convergence $\Longleftrightarrow$ peer adaptation is a *subcritical* branching process, or $E(\xi) < 1$ in (7).

Proof of Theorem 1:

$$\begin{aligned} &E(\xi_q) = 1 - P(A) + E(D_q)\cdot P(C) < 1 \\ &P(A) > E(D_q)\cdot P(C). \end{aligned}$$

Based on our previous discussion, *Theorem 1* can be proved by showing the probability of adaptation absorption $P(A)$ is larger than the probability of adaptation amplification $P(B)$ under the requirement of subcritical branching process, which means unstable nodes will eventually become stable by selecting new capable parents via peer adaptation and the overlay can subsequently converge.

The expected value of substream degree in an overlay should equal to the total number of uploading substreams by nodes over the population, which is approximately K (i.e., $E(D_q) = K$). Then, the probability of event $\mathbf{C}$ happens at time $t_\downarrow$, which is the end of the chain reaction of peer adaptations, is

$$P(C) = [1 - P(A) - P(B)]^{\frac{t_\downarrow}{T_a}}.$$

From (4) and (5), we have

$$\frac{t_\downarrow}{T_a} = (D_p+1)\cdot\frac{l}{T_a}. \tag{8}$$

Then the problem of overlay convergency becomes to be

$$E(\xi_p) = \frac{P(A)}{K\cdot[1-P(A)-P(B)]^{(D_p+1)\cdot l/T_a}}. \tag{9}$$

From (8) and (9), we can see the larger the values of $P(A)$ and $t_\downarrow$, the more likely the topology can self-converge, specifically:

- $P(A)$ is an increasing function with the increase of free upload bandwidth in the overlay;
- D_p plays an important role on the extinction of unstable nodes, where p is a partner selected as a temporary parent. This implies that those peers with larger substream degrees can also contribute to the convergence of the overlay;
- Larger latency between peers help overlay convergence.

In order to better understand overlay convergence, we performed a series of simulations with all detailed functionalities in Coolstreaming, including overlay construction, partnership maintenance, BM management and content delivery. The overlay consisted 10 000 peer nodes as a reasonable scale system and each node interacts with others independently. Partnership and parent-children relationship establishment processes are almost the same as a real system, except that messages (i.e., BMs) in the simulation can only be exchanged periodically within fixed time periods. Since the time periods are relatively short compared to the whole simulation process, the simulation results showed a similar performance as in the real cases. We simulated three different configurations to examine overlay convergence of Coolstreaming. Configuration **A** contains 5% of 100 Mbps upload capacity peers; 20% of 10 Mbps peers; 25% of 512 Kbps peers; 50% of zero upload capacity peers. Configuration **B** is 15%; 20%; 15%; 20%, respectively; configuration **C** is 25%; 20%; 5%; 50%, respectively. The bit rate of the video stream is 512 Kbps. There are six substreams and each peer has maximum ten partners. Buffer length is set to be 60 seconds and cool-down period T_a is 10 seconds. For simplification, we set the two thresholds of peer adaptation T_s and T_p to be fixed at 20 seconds. We simulated the real system by randomly allocating peers into the overlay. Specifically, we measured the number of unstable peers, which could not obtain smooth video streaming from their parent peers, since this has the most significant impact on overlay convergence.

Fig. 9 shows the number of unstable peers along simulation time under different overlay configurations. Having higher upload capacity peers in the overlay, the system has rapidly decreased the number of unstable peers. From the results, it is ob-

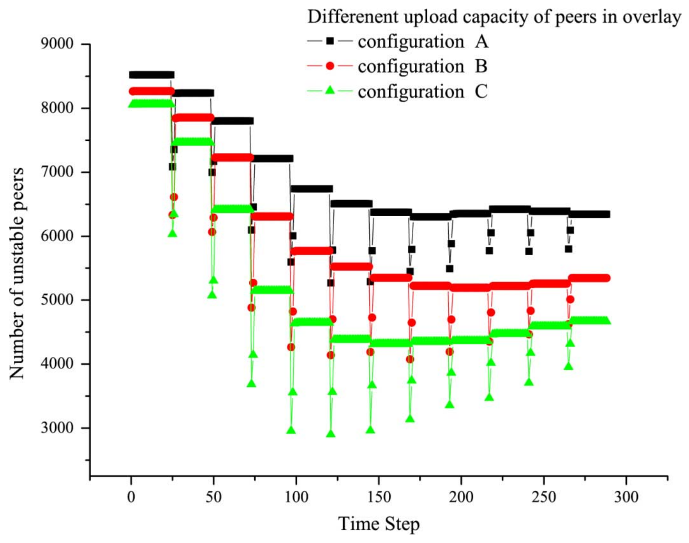


Fig. 9. Number of unstable peers along simulation time under different overlay configuration.

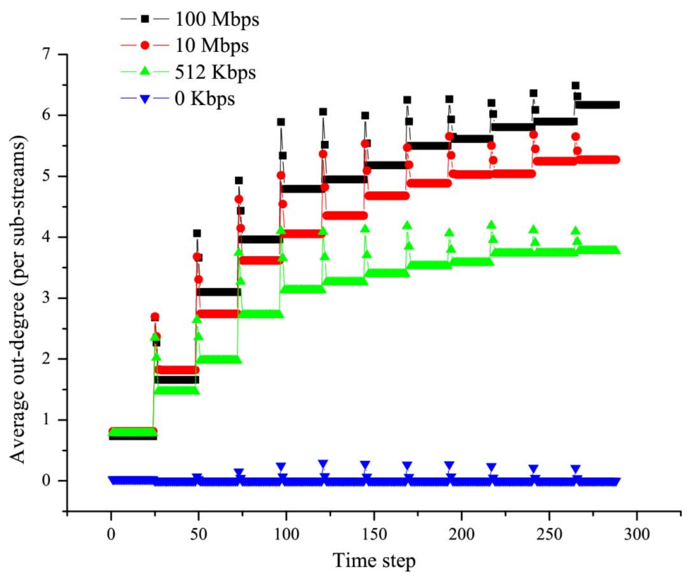


Fig. 10. Average out-degree for different type of peers along simulation time.

served that all configurations can achieve stable overlay after 100 time steps. We also investigated the average out-degree for different types of peers. Fig. 10 plots the average out-degree for different types of peers along simulation time. This clearly illustrates the importance of high upload capacity peers, those we call super-peers whose out-degree is fulfilled faster in random partner selection than other peers.

We can conclude from these discussions that overlay convergence is affected by the availability of upload capacity, number of substreams and buffer size. Intuitively, if we divide a video stream into more substreams (i.e., larger K) and add more buffers (i.e., larger B), peer adaptation resulted from a single substream will have less impact and the larger buffer can further smooth the migration. However, both have negative impacts on video playback quality in terms of synchronization and delay. Therefore, these two parameters have to be carefully selected in a real system.

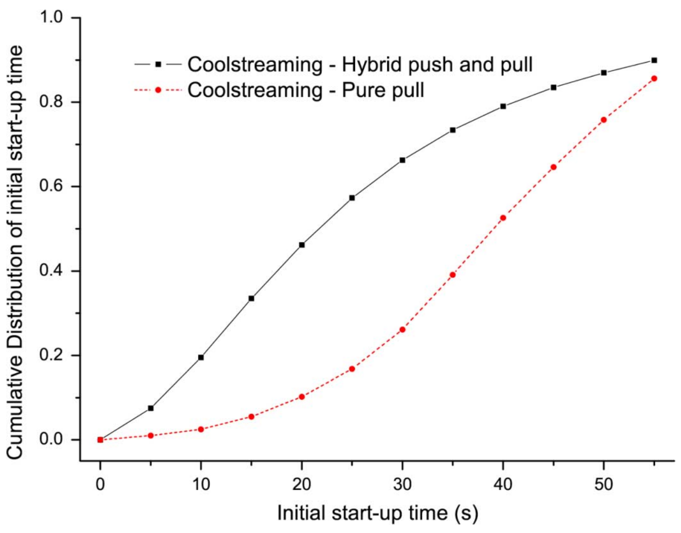


Fig. 11. Comparison of initial start-up delay between pure-pull based and hybrid push-pull.

C. Scalability

Scalability is considered as one of the inherent characteristics in a P2P system. This is made possible by each participating peer's actively contribution of resources. In a macro-view of such a system, it is generally believed that having more participants increases the likelihood for each peer node to retrieve contents from one another. However, a closer examination reveals that this is not always the case given that each newly joined participant demands certain level of service and also competes for available yet finite resources. In BitTorrent-like file sharing applications, the integrity of a file is the only metric of relevance. It is not surprising to see that sometimes it takes hours even days for a user to successfully download a file. We believe that this tolerance in time plays an important role in the scalable property of BitTorrent-like applications. From the model in [20], the download time of users perceived also depends heavily on the number of seeds within the system. It is common in BitTorrent-like applications that peers stay in the system even after the completion of download, thus act as seeds for others.

P2P live video streaming applications possess significant difference in service requirement and user behavior. The quality of service is driven by stream continuity and timing requirement, which essentially invalidate the trade-off of the delay for scale exhibited in P2P file applications. As discussed earlier, the existence of direct-connect/UPnP peers plays an important role in supporting a large number of peers in a P2P streaming system. This was also investigated in [23], which it showed that there existed a critical value in the ratio of the number of super-peers and the number of non-super-peers in the system.

The hybrid pull and push mechanism also significantly improves the system scalability. This can be partly observed from Fig. 11, which shows that the hybrid architecture can significantly reduce the initial start-up delay. Perhaps more importantly, this is demonstrated in Fig. 12, which illustrated higher and more stable continuity index can be maintained. In particular, when there is a significant fluctuation in terms of the number of users in the system, i.e., churn caused by the user

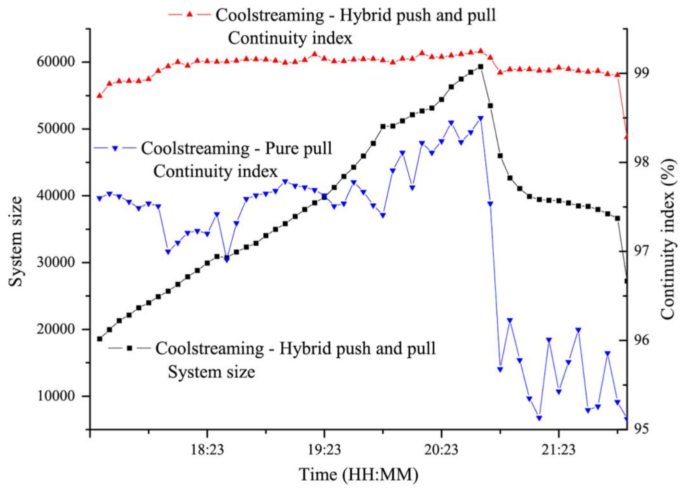


Fig. 12. Comparison of continuity index and number of users.

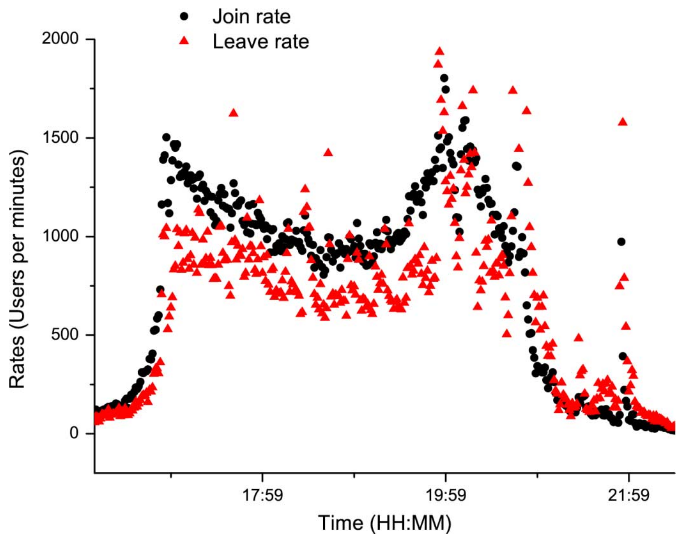


Fig. 13. Join rate, leave rate, and difference between the rates.

departures, there is a visible disruption in the video playback quality in the earlier pull-based system while this is maintained consistently in the new hybrid system. There are two reasons behind this improvement: first, the hybrid pull and push mechanism reduces signal overhearing associated each block transmission, which further results in a faster catch up process for newly joined nodes. Consequently this helps to reduce the initial startup delay; second, the hybrid architecture improves the efficiency in data transmission as the outgoing (uploading) bandwidth is more effectively utilized.

Fig. 13 plots the join rate and leave rate along system time, it illustrates user behavior during peak hours in a typical weekday (17: 00–23:00). Coolstreaming achieves scalability as observed from Fig. 12, in which a large number of users can be accommodated by the system under burst arrivals (often referred as *flash crowd*); at the same time, the continuity index of the system also maintains at a very high level (around 95%). We next discuss the key factors that influence the scalability of P2P streaming systems.

One difficulty in the discussion of scalability in a P2P streaming system is the fact that this notion of *scalability* is essentially tied up with system performance, especially users' perceived video playout quality. The argument is that the simple notion of scalability only refers to the number of users that a system can accommodate; however, in a P2P streaming system, unless each node can receive a sustainable video stream with satisfied quality of service, this simple notion of scalability is largely meaningless. Generally speaking, there are three factors that influence the scale of a P2P streaming system, system *latency* and *overlay stability*.

VI. Conclusion

Coolstreaming represents a milestone in P2P live video streaming systems. It demonstrated practically that a simple random peer selection algorithm coupled with the multiple substream transmission technique has the potential to scale. This paper has provided a comprehensive study on the P2P streaming systems based on the Coolstreaming architecture and experimental results. There are two points that we like to emphasize from this study: 1) a working system is essential in providing basic understanding; 2) there is a large number of practical problems that have to be dealt with in real engineering. We believe much remains to be done, but the results obtained here are promising and display the potential for such an approach to scale to the Internet.

References

[1] P. Francis, *Yoid: Extending the Internet Multicast Architecture* [Online]. Available: http://www.icir.org/yoid
[2] Y. Chu, S. G. Rao, and H. Zhang, "A case for end system multicast," in *Proc. ACM Sigmetrics*, Jun. 2000.
[3] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O. Jr., "Overcast: Reliable multicasting with an overlay network," in *Proc. OSDI*, Oct. 2000.
[4] X. Zhang, J. Liu, B. Li, and P. Yum, "DONet/Coolstreaming: A data-driven overlay network for live media streaming," in *Proc. IEEE Infocom*, Mar. 2005.
[5] V. Venkararaman, K. Yoshida, and P. Francis, "Chunkspread: Heterogeneous unstructured end system multicast," in *Proc. IEEE ICNP*, Nov. 2006.
[6] X.-Y. Zhang, J. Liu, and B. Li, "On large-scale peer-to-peer live video distribution: Coolstreaming and its preliminary experimental results," in *Proc. IEEE MMSP'2005*, Oct. 2005.
[7] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *Proc. ACM SIGCOMM*, Aug. 2002.
[8] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron, "Scribe: A large-scale and decentralized application-level multicast infrastructure," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 8, Oct. 2002.
[9] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan, "Resilient multicast using overlays," in *Proc. ACM SIGMETRICS*, Jun. 2003.
[10] Y. Cui, Y. Xue, and K. Nahrstedt, "Optimal resource allocation in overlay multicast," in *Proc. IEEE ICNP*, Nov. 2003.
[11] T. Small, B. Liang, and B. Li, "Scaling laws and tradeoffs in peer-to-peer live multimedia streaming," in *Proc. ACM Multimedia'06*, Oct. 2006.
[12] W. Wang and B. Li, "Market-based self-optimization for autonomic service overlay networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 12, Dec. 2005.
[13] M. Bishop, S. Rao, and K. Sripanidkulchai, "Considering priority in overlay multicast protocols under heterogeneous environments," in *Proc. IEEE Infocom*, Sep. 2006.
[14] M. Wang and B. Li, "Lava: A reality check of network coding in peer-to-peer live streaming," in *Proc. IEEE Infocom*, May 2007.

[15] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: High-bandwidth content distribution in cooperative environments," in *Proc. SOSP*, Oct. 2003.

[16] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," in *Proc. NOSSDAV*, May 2002.

[17] V. Vishnumurthy and P. Francis, "On heterogeneous overlay construction and random node selection in unstructured P2P networks," in *Proc. IEEE Infocom*, Apr. 2006.

[18] Y. Sun, M. Bishop, and S. Rao, "Enabling contribution awareness in an overlay broadcast system," in *Proc. ACM SIGCOMM*, Sep. 2006.

[19] N. Magharei and R. Rejaie, "PRIME: Peer-to-peer receiver-drIven MEsh-based streaming," in *Proc. IEEE Infocom*, May 2007.

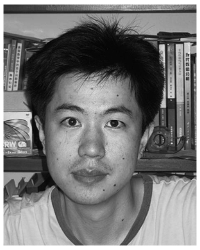
[20] D. Qiu and R. Srikant, "Modeling and performance analysis of BitTorrent-like peer-to-peer networks," in *Proc. ACM SIGCOMM*, Aug. 30–Sept. 3 2004.

[21] P. T. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massoulie, "From epidemics to distributed computing," *IEEE Computer*, May 2004.

[22] A. J. Ganesh, A.-M. Kermarrec, and L. Massoulie, "Peer-to-peer membership management for gossip-based protocols," *IEEE Trans. Comput.*, vol. 52, no. 2, Feb. 2003.

[23] R. Kumar, Y. Liu, and K. W. Ross, "Stochastic fluid theory for P2P streaming systems," in *Proc. IEEE Infocom*, May 2007.

[24] P. Haccou, P. Jagers, and V. A. Vatutin, *Branching Processes: Variation, Growth, and Extinction of Populations*, 2005.



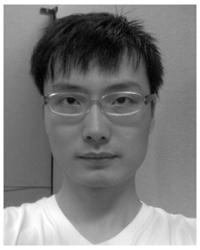
Susu Xie received the B.S. degree from Tsinghua University, Beijing, China, in 1998. After several years experience in software development in real-time multimedia applications, he is currently pursuing the Ph.D. degree at the of Hong Kong University of Science and Technology.

His research interests are in the area of Internet and wireless networks with emphasis on multimedia applications.

Bo Li (S'89–M'92–SM'99) received his B.Eng. (summa cum laude) and M.Eng. degrees in the computer science from Tsinghua University, Beijing, China, in 1987 and 1989, respectively, and the Ph.D. degree in the electrical and computer engineering from University of Massachusetts, Amherst, in 1993.

Between 1993 and 1996, he worked on high performance routers and ATM switches in IBM Networking System Division, Research Triangle Park, NC. Since 1996, he has been with the Department of Computer Science, Hong Kong University of Science and Technology. Since 1999, he has also held an adjunct researcher position at the Microsoft Research Asia (MSRA), Beijing. His current research interests are on adaptive video multicast, peer-to-peer streaming, resource management in mobile wireless systems, across layer design in multihop wireless networks, content distribution and replication.

Dr. Li is currently a Distinguished Lecturer of the IEEE Communications Society. He has served on the editorial board for a large number IEEE journals such as IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, tie was the also a guest editor for three special issues of IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS. He was the Co-TPC Chair for IEEE Infocom'2004.



Gabriel Y. Keung received the B.S. and M.Phil. degrees from the Department of Physics, Hong Kong University of Science and Technology (HKUST) , in 2002 and 2004, respectively. He is currently pursuing w the Ph.D. degree at the Department of Computer Science and Engineering at HKUST.

His previous research interests are in the area of wireless communication networks, with emphasis on resource allocation and mobility management for multimedia traffic. His current research interests are in area of Internet and P2P networks with emphasis on multimedia applications.

Xinyan Zhang (S'03) received the B.S. degree in computer science from Tsinghua University, Beijing, China, in 2001 and the M.Phil. degree from the Department of Information Engineering, Chinese University of Hong Kong in 2004.

He developed and implemented Coolstreaming, one of the largest P2P global live streaming systems at the time (2004–2005). He is with Roxbeam Corporation, Beijing.