Opportunities and Challenges of Peer-to-Peer Internet Video Broadcast

Jiangchuan Liu*, Sanjay G. Rao†, Bo Li‡, and Hui Zhang§

*School of Computing Science, Simon Fraser University
Burnaby, British Columbia, Canada
Email: jcliu@cs.sfu.ca

†School of Electrical and Computer Engineering, Purdue University
West Lafayette, Indiana, USA
Email: sanjay@ecn.purdue.edu

‡Department of Computer Science and Engineering, Hong Kong University of Science and Technology
Clear Water Bay, Hong Kong
Email: bli@cs.ust.hk

§School of Computer Science, Carnegie Mellon University
Pittsburgh, Pennsylvania, USA
Email: hzhang@cs.cmu.edu

***Abstract*— There have been tremendous efforts and many technical innovations in supporting real-time video streaming in the past two decades, but cost-effective large-scale video broadcast has remained an elusive goal. IP multicast represented the earlier attempt to tackle this problem, but failed largely due to concerns regarding scalability, deployment, and support for higher level functionality. Recently, peer-to-peer based broadcast has emerged as a promising technique, which has been shown to be cost effective and easy to deploy. This new paradigm brings a number of unique advantages such as scalability, resilience and also effectiveness in coping with dynamics and heterogeneity.**

While peer-to-peer applications such as file download and voice over IP have gained tremendous popularity, video broadcast is still in its early stages and its full potential remains to be seen. This article reviews the state-of-the-art of peer-to-peer Internet video broadcast technologies. We describe the basic taxonomy of peer-to-peer broadcast and summarize the major issues associated with the design of broadcast overlays. We closely examine two approaches, namely, *tree-based* and *data-driven*, and discuss their fundamental trade-off and potential for large-scale deployment. Finally, we outline the key challenges and open problems, and highlight possible avenues for future directions.

I. Introduction

A large number of emerging applications, including Internet TV, broadcast of sports events, online games, and distance education, require support for video broadcast, i.e., simultaneous video delivery to a large number of receivers. The vision of enabling simultaneous video broadcast as a common Internet utility in a manner that any publisher can broadcast content to any set of receivers has been driving the research agenda in the networking community for over two decades. For much of the 1990's, the research and industrial community investigated support for such applications using the IP Multicast architecture [13]. However, serious concerns regarding its scaling, support for higher level functionality, and deployment have dogged IP Multicast. The sparse deployment of IP Multicast, and the high cost of bandwidth required for server-based solutions or Content Delivery Networks (CDNs) are two main factors that have limited broadcast to only a subset of Internet content publishers. While many network service providers have enabled IPTV services that deliver quality video to their own subscribers using packet switching, there remains a need for cost-effective, ubiquitous support for Internet-wide video broadcast, and the solutions will certainly be beneficial to IPTV as well.

In recent years, there has been significant interest in the use of peer-to-peer technologies for Internet video broadcast. There are two key drivers making the approach attractive. First, such technology does not require support from Internet routers and network infrastructure, and consequently is extremely cost-effective and easy to deploy. Second, in such a technology, a participant that tunes into a broadcast is not only downloading a video stream, but also uploading it to other participants watching the program. Consequently, such an approach has the potential to scale with group size, as greater demand also generates more resources. The scaling challenge for video broadcast is enormous. To reach 100 million viewers, delivery of TV quality video encoded in MPEG-4 (1.5Mbps) will require 1.5 Tbps aggregate capacity. To put things into perspective, consider two of the largest scale

Internet video broadcasts: the AOL broadcast of Live 8 concert in July 2005 [55], which at the peak has 175,000 simultaneous viewers, and the CBS broadcast of the NCAA tournament [54] in March 2006, which at the peak has 268,000 simultaneous viewers. Even with today's low bandwidth Internet video of 400 Kbps, the CBS/NCAA broadcast needed more than 100Gbps server and network bandwidth. As a comparison, Akamai, the largest commercial CDN service provider, reports a peak aggregate capacity of 200Gbps with its tens of thousands of servers [46].

Peer-to-peer technologies have emerged as important for a wide range of applications such as file download and voice over IP [46]–[48], [53]. However, video broadcast applications pose very different challenges than these other applications. Specifically, video broadcast imposes stringent real-time performance requirements in terms of bandwidth and latency. This is in contrast to file download applications like BitTorrent [47], where the objective is to download a complete file, and timeliness requirements are not critical. In fact, it may typically take several hours to a few days to download large files using BitTorrent, and such delays are clearly not feasible for video broadcast applications. While voice over IP applications also involve real-time requirements, video broadcast applications are much more challenging given they need to *simultaneously* support a large number of participants, deal with dynamic changes to participant membership, and cope with high bandwidth requirement of the video.

The distinguishing and stringent requirements of video broadcast necessitate fundamentally different design decisions and approaches. This article reviews the state-of-the-art of peer-to-peer technologies for Internet video broadcast, and presents a taxonomy of various solutions that have emerged. In particular, two broad approaches have emerged: *tree-based approaches* and *data-driven randomized approaches*. We examine typical examples and their differences. We then outline future challenges that must be addressed to make Internet video broadcast using peer-to-peer services a reality.

The remainder of this article is organized as follows. Section II briefly discusses the architectural choices for Internet broadcast. In Section III, we highlight the key difference between video broadcast and conventional peer-to-peer applications, and taxonomize the existing approaches for peer-to-peer video broadcast. Case studies for the typical approaches are presented in Section IV. We then present technical challenges and open issues in Section V. The deployment status of the practical peer-to-peer broadcast systems are reviewed in Section VI, followed by a discussion on the potential deployment challenges. Finally, Section VII concludes the article and highlights possible avenues for future directions.

II. Architectural Choices for Internet Broadcast

We first review the architectural choices for supporting Internet broadcast/multicast (see Fig. 1). There are subtle differences between broadcast and multicast: the former is to all the destinations and the latter is to a group of destinations

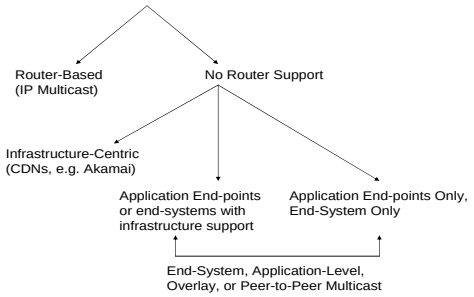

Fig. 1. Taxonomy of architectures for Internet broadcast

only. While broadcast is possible in air, cable networks, or local area networks, it simply cannot be carried over the global Internet. Nevertheless, given the popular use of this term in radio and TV industries, in this article, we do not distinguish it from multicast if the context is clear.

A. Router-Based Architectures: IP Multicast

In the Internet environment, the primary issue for broadcast/multicast is determining at which layer it should be implemented. There are two conflicting considerations that we need to reconcile. According to the end-to-end argument, a functionality should be 1) pushed to higher layers if possible; unless 2) implementing it at the lower layer can achieve significant performance benefits that outweigh the cost of additional complexity. In his seminal work in 1989 [13], Deering argued that this second consideration should prevail and multicast should be implemented at the IP layer. This view has since been widely accepted, leading to the IP multicast model.

IP multicast is a loosely coupled model that reflects the basic design principles of the Internet. It retains the IP interface, and introduces the concept of open and dynamic groups, which greatly inspires later proposals. Given that the network topology is best-known in the network layer, multicast routing in this layer is also the most efficient. Unfortunately, despite the tremendous effort in the past 15 years, today's IP multicast deployment remains limited in reach and scope. The reason is complex and involves not only technical obstacles, but also, more importantly, economic and political concerns. First, IP multicast requires routers to maintain per-group state, which not only violates the "stateless" architectural principle, but also introduces high complexity and serious scaling constraints at the IP layer. Second, IP multicast is a best-effort service, and attempts to conform to the traditional separation of routing and transport that has worked well in the unicast context. However, providing higher level features such as error, flow, and congestion control has been shown to be more difficult than in the unicast case. Finally, IP multicast calls for changes at the infrastructure level, and this slows down the pace of deployment. In particular, there is a lack of incentive to install multicast-capable routers and to carry multicast traffic.

B. Non Router-Based Architectures

The placement of the multicast functionality was revisited in the new millennium, and several researchers have advocated

moving multicast functionality away from routers towards end systems [6], [9], [16], [43]. In these approaches, multicast related features, such as group membership, multicast routing and packet duplication, are implemented at end systems, assuming only unicast IP service. End systems participate in multicast communication via an overlay structure, in the sense that each of its edges corresponds to a unicast path between two nodes in the underlying Internet.

Moving multicast functionality to end systems has the potential to address many of the problems associated with IP multicast. Since all packets are transmitted as unicast packets, deployment is accelerated. It maintains the stateless nature of the network by requiring end systems, which subscribe only to a small number of groups, to perform additional complex processing for any given group. In addition, solutions for supporting higher layer features can be significantly simplified by leveraging well understood unicast solutions, and by exploiting application-specific intelligence.

It must be noted that moving multicast functionality away from routers involves performance penalties. For example, it is impossible to completely prevent multiple overlay edges from traversing the same physical link and thus some redundant traffic on physical links is unavoidable. Further, communication between end systems involves traversing other end systems, potentially increasing latency. Hence, many research efforts have focused on addressing these performance concerns with overlays.

C. Peer-to-Peer Architectures

Given that non-router based architectures push functionality to the network edges, there are several choices in instantiating such an architecture. On the one end of the spectrum is an *infrastructure-centric* architecture, where an organization that provides value-added services deploys proxies at strategic locations on the Internet. End systems attach themselves to nearby proxies, and receive data using plain unicast. Such an approach is also commonly referred to as Content Delivery Networks (CDNs), and has been employed by companies such as Akamai [46]. On the other end of the spectrum is a purely *application end-point* architecture, where functionality is pushed to the users actually participating in the multicast group. Administration, maintenance, responsibility for the operation of such a peer-to-peer system are distributed among the users, instead of being handled by a single entity.

The focus of this paper is on simultaneous video broadcast using the application end-point architecture, referred to as *peer-to-peer broadcast/multicast*. Such similar terms as *end-system multicast, overlay multicast, application-layer multicast*, have also been used in the literature. In the purest form, such architectures rely exclusively on bandwidth resources at application end-points. However, one could also conceive of hybrid architectures that seek to use the bandwidth resources of application end-points to the extent possible, but may leverage infrastructure resources where available. We will include such architectures in our discussion of peer-to-peer broadcast.

Category	Bandwidth-sensitive	Delay-sensitive	Scale
File download	$\times$	$\times$	Large
On-demand streaming	$\surd$	$\surd$	Large
Audio/video conferencing	$\surd/\times$	$\surd$	Small
Simultaneous broadcast	$\surd$	$\surd$	Large

TABLE I

A TAXONOMY OF TYPICAL PEER-TO-PEER APPLICATIONS.

The motivation behind basing applications on the peer-to-peer paradigm derives to a large extent from its ability to leverage the bandwidth resources of end systems actually participating in the communication. Addition of new participants not only requires more bandwidth support, but also involves additional bandwidth contributed by the new participants. In contrast, while an infrastructure-centric service can potentially deal with a smaller number of well-defined groups, it is unclear whether it can support the bandwidth requirements associated with deploying tens of thousands of high-bandwidth broadcast applications. Further, the application end-point architecture is instantaneous to deploy, and can enable support of applications with minimal setup overhead and cost.

While the application end-point architectures have the promise to enable ubiquitous deployment, the infrastructure-centric architecture can potentially provide more robust data delivery with dedicated, better provisioned and more reliable proxies placed at strategic locations. In contrast, the application end-point architectures potentially involve a wide range of autonomous users that may not provide as good performance but easily fail or leave at will. Individual user joining and leaving have more significant impact on the system performance. Thus, the key challenge for application end-point architectures is to function, scale and self-organize with a highly transient population of users, without the need of a central server and the associated management overhead.

III. PEER-TO-PEER VIDEO BROADCAST

In this section, we discuss the distinguishing characteristics of video broadcast applications. We then discuss why these correspond to a very different domain requiring very different solutions than many other peer-to-peer applications.

A. Contrast from other Peer-to-Peer Applications

A video broadcast system typically has a single dedicated source, which may be assumed not to fail, and is present throughout a broadcast session. The address of the source is known in advance, serving as a rendezvous for new users to join the session. There are several distinguishing characteristics of such a system:

- Large scale, corresponding to tens of thousands of users simultaneously participating in the broadcast.
- Performance-demanding, involving bandwidth requirements of hundreds of kilobits per-second.
- Real-time constraints, requiring timely and continuously streaming delivery. While interactivity may not be critical and minor delays can be tolerated through buffering, it is nevertheless critical to get video uninterrupted.

• Gracefully degradable quality, enabling adaptive and flexible delivery that accommodates bandwidth heterogeneity and dynamics.

The above characteristics combined yield a unique application scenario that differs from other typical peer-to-peer applications, including *on-demand streaming*, *audio/video conferencing*, and *file download* (see Table I).

Among these applications, on-demand streaming and audio/video conferencing also have stringent delay and bandwidth requirements. However, in on-demand streaming, the users are asynchronous, and it thus belongs to a different problem domain. Audio/video conferencing applications differ from broadcast applications in that they are interactive with latency being even more critical, and are multi-point, that is, may require any participant to be a source. However, such applications are typically of smaller scales, involving only a few hundred participants. Example systems of this kind include Skype [53] (limited to audio conversation), and research proposals such as Narada [10] and Gossamer [6].

Peer-to-peer file download applications such as BitTorrent [47], and EMule [48] involve information distribution to tens of thousands of participants. However, the stringent real-time and bandwidth requirements make video broadcast more challenging. For example, BitTorrent enables peers to exchange any segment of the content being distributed; the order in which they arrive is not important. By contrast, such techniques are not feasible in streaming applications. Further, given the timeliness requirements, streaming video applications typically must include techniques for graceful degradation of video quality rather than involving excessive delays.

Another key problem in peer-to-peer file download is to design techniques for efficient indexing and search, that is, to locate a massive number of files distributed among a large number of peers. Solutions in this space include Napster, Gnutella, and Distributed Hashing Table (DHT) techniques [34], [49], [51]. While the design of overlays for efficient indexing and searching a large video repository pose several issues, in peer-to-peer video broadcast, we are more interested in the efficiency of data communication.

B. Design Issues

The key problem in a peer-to-peer video broadcast system is to organize the peers into an overlay for disseminating the video stream. Following are the important criteria for overlay construction and maintenance.

• *Overlay efficiency*: The overlay constructed must be efficient both from the network and the application perspectives. For broadcast video, high bandwidth and low latencies are simultaneously required. However, given that applications are real-time but not interactive, a startup delay of a few seconds can be tolerated.

• *Scalability and load balancing:* Since broadcast systems can scale to tens of thousands of receivers, the overlay must scale to support such large sizes, and the overhead associated must be reasonable even at large scales.

• *Self-organizing*: The construction of overlay must take place in a distributed fashion and must be robust to dynamic changes in group membership. Further, the overlay must adapt to long-term variations in Internet path characteristics (such as bandwidth and latency), while being resilient to inaccuracies. The system must be self-improving in that the overlay should incrementally evolve into a better structure as more information becomes available.

• *Honor per-node bandwidth constraints:* Since the system relies on users contributing bandwidth, it is important to ensure that the total bandwidth a user is required to contribute does not exceed its inherent access bandwidth capacity. On the other hand, users also have heterogeneous inbound bandwidth capabilities, and it is desirable to have mechanisms to ensure they can receive different qualities of video, proportional to their ability.

• *System Considerations:* In addition to the above algorithmic considerations, several important system issues must be addressed in the design of a complete broadcasting system. Examples include the choice of transport protocol and the interaction with video players. Further, a key challenge of peer-to-peer systems involves the presence of large fractions of users behind NATs and firewalls - the connectivity restrictions posed by such peers may severely limit the overlay capacity.

C. Approaches for Overlay Construction

A large number of proposals have emerged in recent years for peer-to-peer video broadcast [4]–[6], [8], [9], [14], [16], [19], [22], [27]–[29], [36], [37], [41], [43]. While these proposals differ on a wide-range of dimensions, in this article, we focus on the approach taken towards the overlay structure used for data dissemination. In particular, the proposals can be broadly classified into two categories, namely, **tree-based** and **data-driven randomized** overlay construction, which we discuss below.

Tree-Based Approaches: The vast majority of the proposals to date can be categorized as a tree-based approach. In such an approach, peers are organized into structures (typically trees) for delivering data, with each data packet being disseminated using the same structure. Nodes on the structure have well-defined relationships, for example, “parent-child” relationships in trees. Such approaches are typically push-based, that is, when a node receives a data packet, it also forwards copies of the packet to each of its children. Since all data packets follow this structure, it becomes critical to ensure the structure is optimized to offer good performance to all receivers. Further, the structure must be maintained, as nodes join and leave the group at will – in particular, if a node crashes or otherwise stops performing adequately, all of its offspring in the tree will stop receiving packets, and the tree must be repaired. Finally, when constructing tree-based structures, loop avoidance is an important issue that must be addressed.

Tree-based solutions are perhaps the most natural approach, and work well with widely available video codecs. However, one concern with tree-based approaches is that the failure of nodes, particularly those higher in the tree may disrupt delivery

of data to a large number of users, and potentially result in poor transient performance. Further, a majority of nodes are leaves in the structure, and their out-going bandwidth is not being utilized. In response to these concerns, researchers have been investigating more resilient structures for data delivery. In particular, one approach that has gained popularity is multi-tree based approaches [5], [28], which we discuss further in Section IV-B.

Data-Driven Approaches: [1] Recently, researchers have proposed data driven approaches for peer-to-peer broadcast [29], [41]. Data-driven overlay designs sharply contrast to tree-based designs in that they do not construct and maintain an explicit structure for delivering data. The underlying argument is that, rather than constantly repair a structure in a highly dynamic peer-to-peer environment, we can use the availability of data to guide the data flow.

A naive approach to distributing data without explicitly maintaining a structure is to use gossip algorithms [15]. In a typical gossip algorithm, a node sends a newly generated message to a set of randomly selected nodes; these nodes do similarly in the next round, and so do other nodes until the message is spread to all. The random choice of gossip targets achieves resilience to random failures and enables decentralized operation. However, gossip cannot be used directly for video broadcast because its random push may cause significant redundancy with the high-bandwidth video. Further, without an explicit structure support, minimizing startup and transmission delays become significant problems.

To handle this, approaches such as Chainsaw [29] and CoolStreaming [41] adopt pull-based techniques. More explicitly, nodes maintain a set of partners, and periodically exchange data availability information with the partners. A node may then retrieve unavailable data from one or more partners, or supply available data to partners. Redundancy is avoided, as the node pulls data only if it does not already possess it. Further, since any segment may be available at multiple partners, the overlay is robust to failures – departure of a node simply means its partners will use other partners to receive data segments. Finally, the randomized partnerships imply that the potential bandwidth available between the peers can be fully utilized.

The data-driven approach at first sight may appear similar to techniques used in file download solutions like BitTorrent [47]. However, the crucial difference here is that the real-time constraints imply that segments must be obtained in a timely fashion. Thus, an important component of a data-driven broadcast systems is a scheduling algorithm, which strives to schedule the segments that must be downloaded from various partners to meet the playback deadlines.

IV. Case Studies

In this section, we present concrete case studies on peer-to-peer video broadcast system. We use End System Multicast (ESM) and CoolStreaming as representative examples for tree-based and data-driven systems, respectively. As discussed in Section VI, both systems have been deployed among real users. We also discuss the case of using multiple trees, which represents a natural way to enhance tree-based approaches, originally proposed in [5], [28], and being adopted in recent versions of ESM.

A. Example Tree-based Approach: ESM

The ESM system [9] employs a structure-based overlay protocol which is distributed, self-organizing, performance-aware, and constructs a tree rooted at the source. The tree is optimized primarily for bandwidth, and secondarily for delay.

Group Management: Each ESM node maintains information about a small random subset of members, as well as information about the path from the source to itself. A new node joins the broadcast by contacting the source and retrieving a random list of members that are currently in the group. It then selects one of these members as its parent using the parent selection algorithm. To learn about members, a gossip-like protocol is used. Each node A periodically picks one member (say B) at random, and sends B a subset of group members that A knows, along with the last timestamp it has heard for each member. When B receives a membership message, it updates its list of known members. Finally, members are deleted if its state has not been refreshed in a period.

Membership Dynamics: Dealing with graceful member leave is fairly straight-forward: members continue forwarding data for a short period, while its children look for new parents using the parent selection method described below. This serves to minimize disruptions to the overlay. Members also send periodic control packets to their children to indicate existence.

Performance-Aware Adaptation: Each node maintains the application-level throughput it is receiving in a recent time window. If its performance is significantly below the source rate, then it selects a new parent as described in the parent selection algorithm. One key parameter is the *detection time*, which indicates how long a node must stay with a poor performing parent before it switches to another parent. The ESM system employs a default detection time of 5 seconds. The choice of this timeout value has been influenced by the fact that a congestion control protocol is running on the data path (TCP or TFRC). Switching to a new parent requires going through a slow-start phase, which may take 1 - 2 seconds to get the full source rate. The protocol may need to adaptively tune the detection time because nodes may not be capable of receiving the full source rate, there may be few good and available parent choices in the system, or nodes may experience intermittent network congestion on links close to them. Changing parents under these conditions may not be fruitful.

Parent Selection: When a node (say A) joins the broadcast, or needs to make a parent change, it probes a random subset of nodes it knows. The probing is biased toward members that have not been probed or have low delay. Each node B that responds provides information about: (i) the performance

[1] Also referred to as *mesh*-based approach [26], [38]

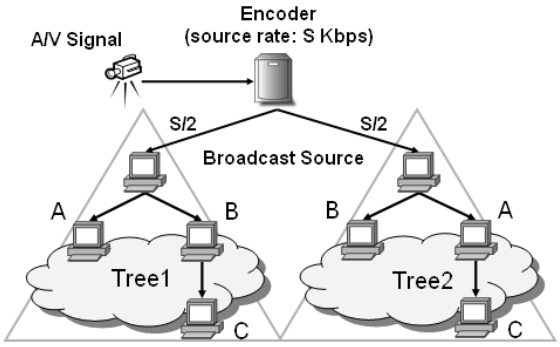

Fig. 2. A multi-tree broadcast with two trees.

(application throughput) it is currently receiving, and delay from the source; (ii) whether it is degree-saturated or not; and (iii) whether it is a descendant of A. The probe also enables A to determine the round-trip time to B. A waits for responses for a timeout period of 1 second, a large enough value of Internet round-trip times that maximizes the number of responses received from members. From the responses A receives, it eliminates its descendants and members that are saturated. The system uses statically configured values to assess the number of children a parent can support, and requires users to indicate whether there is at least a 10 Mbps up-link to the Internet.

For each node B that has not been eliminated, A evaluates the performance (throughput and delay) it expects to receive if B were chosen as a parent. For example, the expected application throughput is the minimum of the throughput B is currently seeing and the available bandwidth of the path between B and A if the estimate is available. History of past performance is maintained so if A has previously chosen B as parent, then it has an estimate of the bandwidth of the path between B and A. If the bandwidth to nodes is not known, then A picks a parent based on delay. A identifies the node B that could best improve performance, and switches to the parent B either if the estimated application throughput is high enough for A to receive a higher quality stream, or if B maintains the same bandwidth level as A's current parent, but improves delay. The latter heuristic helps to increase tree efficiency by clustering nearby nodes.

B. Example Resilient Structure Approach: Multi-trees

As mentioned earlier, a single-tree based approach suffers from two limitations: disruptive delivery due to failures of high-level nodes, and under-utilized out-going bandwidth in leaf nodes. More resilient structures, in particular, multi-tree [5], [28], thus have been introduced. Here, the source encodes the stream into sub-streams and distributes each sub-stream along a particular overlay tree. The quality experienced by a receiver depends on the number of sub-streams that it receives. There are two key advantages of the multi-tree solution. First, the overall resiliency of the system is improved, as a node is not completely disrupted by the failure of an ancestor on a given tree. Second, the potential bandwidth of all nodes can be utilized, as long as each node is not a leaf in at least one tree.

Figure 2 illustrates how broadcast content is delivered with a multi-tree approach using two trees. The source distributes a stream rate $S/2$ over each tree, where S is the source rate. C receives $S/2$ from tree, with potentially different parents to reconstruct the original content. Nodes A and B each can contribute a bandwidth $S/2$, and allocate their bandwidth in Tree2 and Tree1, respectively. In a single tree approach, it would simply have not been possible to utilize the contributions of these nodes.

C. Example Data-driven Approach: CoolStreaming

CoolStreaming [41] is one of the first data-driven systems. A CoolStreaming node consists of three key modules: (1) a membership manager, which helps the node maintain a partial view of other overlay nodes; (2) a partnership manager, which establishes and maintains partnership with other nodes; (3) a scheduler, which schedules the transmission of video data.

Group and Partner Management: Like ESM, CoolStreaming requires newly joining nodes to contact the origin server to obtain an initial set of partner candidates. Each node also maintains a partial subset of other participants in the group. In particular, CoolStreaming employs an existing Scalable Gossip Membership protocol, SCAM, to distribute membership messages, which enables scalable, light-weight, and uniform partial view at each node.

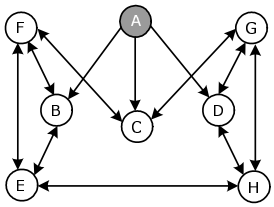

Fig. 3. An illustration of partnerships in CoolStreaming with A being the source node.

The key aspect of the design where CoolStreaming differs from tree-based approaches is the lack of a formal structure for delivering data. More explicitly, a video stream is divided into segments of a uniform length, and the availability of the active segments in the buffer of a node is represented by a Buffer Map (BM). Each node continuously exchange its BM with its partners, and then determines which segment is to be fetched from which partner accordingly. An example of the partnership in CoolStreaming is shown in Fig. 3. Such partnerships are adaptively configured throughout a broadcast session.

Scheduling Algorithm: Timely and continuous segment delivery is crucial to video broadcast, but not to file download. In BitTorrent, the download phases of the peers are not synchronized, and the segments can be downloaded out-of-order. In CoolStreaming, the playback progress of the peers is roughly synchronized, and any segment downloaded after

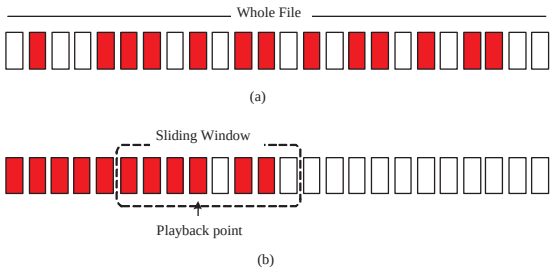

Fig. 4. Buffer snapshots of BitTorrent (a) and CoolStreaming (b), where shaded segments are available in the buffer.

its playback time will be useless. A sliding window thus represents the active buffer portion, as shown in Fig. 4.

Suggested by experimental results, CoolStreaming adopts a sliding window of 120 segments, each of 1-second video. A BM thus consists of a bitstring of 120 bits, each indicating the availability of the corresponding segment. The sequence number of the first segment in the sliding window is recorded by another two bytes, which can be rolled back for extra long video programs (>24 hours). Given the BMs of a node and its partners, a schedule is then generated for fetching the expected segments from the partners. For a homogeneous and static network, a simple round-robin scheduler may work well, but for a dynamic and heterogeneous network, a more intelligent scheduler is necessary. Specifically, the scheduling algorithm strikes to meet two constraints: the playback deadline for each segment, and the heterogeneous streaming bandwidth from the partners. If the first constraint cannot be satisfied, then the number of segments missing deadlines should be kept minimum, so as to maintain a continuous playback. This problem is a variation of the *Parallel machine scheduling*, which is known NP-hard. It is thus not easy to find an optimal solution, particularly considering that the algorithm must quickly adapt to highly dynamic network conditions. Therefore, simple heuristics of fast response time have been developed in CoolStreaming.

Failure Recovery and Partnership Refinement: A CoolStreaming node can depart either gracefully or accidentally due to an unexpected failure. In either case, the departure can be easily detected after an idle time and an affected node can quickly react through re-scheduling using the BM information of the remaining partners. Besides this built-in recovery mechanism, CoolStreaming also lets each node periodically establish new partnerships with nodes randomly selected from its local membership list. This operation serves two purposes: first, it helps each node maintain a stable number of partners in the presence of node departures; second, it helps each node explore partners of better quality, e.g., those constantly having a higher upload bandwidth and more available segments.

V. Technical Challenges and Open Issues

While the research on peer-to-peer broadcast has made great strides in recent years, there are several technical challenges and open issues to be overcome before ubiquitous Internet video broadcast can be enabled by peer-to-peer solutions. We discuss some of these issues in this section.

A. Tree based vs. data driven, could there be any hybrid?

Both tree-based structured and data-driven structureless overlays have shown their success in practical deployment, and yet neither completely overcomes the challenges from the dynamic peer-to-peer environment. The selling point for data-driven systems is their simplicity, but they suffer from a latency-overhead trade-off [37], [40]. If nodes choose to send notifications for every segment arrival, then the overhead will be increased. Periodical notifications containing buffer maps reduces the overhead but increase the latencies. A tree-based system does not suffer from this trade-off, but has to face the inherent instability, maintenance overhead, and bandwidth under-utilization. A natural question is therefore whether we can combine them to realize a hybrid overlay that is both efficient and robust.

The combination can be achieved in different dimensions. An example is Chunkyspread [37], which splits a stream into distinct slices and transmits over separate but not necessarily disjoint trees. The participating nodes also form a neighboring graph, and the degree in the graph is proportional to its desired transmission load. This hybrid design greatly simplifies the tree construction and maintenance, and largely retains its efficiency and achieves fine-grained control over load.

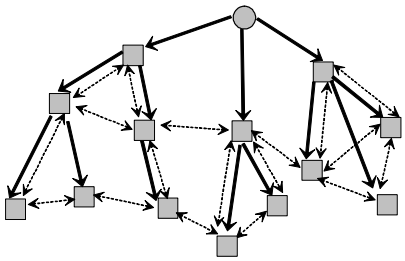

Fig. 5. An illustration of a hybrid tree and data-driven design.

Another direction is a more explicit tree-bone based approach [38]. Our trace studies have shown strong evidence that most of the data blocks delivered through a mesh-based data-driven overlay essentially follow a specific tree structure or a small set of trees. The similarity of the trees, defined as the fraction of the common links, can be as high as 70%. The overlay performance thus closely depends on the set of common internal nodes and their organization. This suggests that, while maintaining a prior topology for all the nodes is costly, optimizing the organization for a core subset is worth consideration. In particular, if such a subset consists mainly of the stable nodes, with others being organized through a mesh, we can expect high efficiency with low overhead and delay simultaneously. Fig. 5 shows such an example. While only a single tree structure is shown, a multi-tree-based backbone can also be deployed in practice. The key challenge is that a set of stable overlay nodes need to be identified and positioned at appropriate locations in the tree. Such a requirement can

conflict with the bandwidth and delay optimization in tree construction. An additional complication when discussing stability is the dependence on human behavior - that is, on how long the user decides to stay. This might in turn be correlated to the performance seen by the user.

B. Incentives and fairness

So far we have made an implicit assumption that users can and are willing to collaborate. In reality however this is not always the case. Measurement studies have shown that, in some peer-to-peer broadcast systems, a small set of nodes are requested to contribute 10 to 35 times more uploading bandwidth than downloading bandwidth [3]. Such overhead will hinder any potential users from being cooperative. These autonomous users can be selfish and misbehave in order to maximize their benefits [23]. As a result, there could be many free riders in a peer-to-peer system that either refuse to contribute or avoid contributing bandwidth, e.g., in tree construction, always acting as a leaf by declaring a poor outbound bandwidth. This situation, non-existent in IP multicast, can seriously affect the overall service quality experienced by cooperative peers. Therefore, a proper incentive mechanism is critical to the performance of a peer-to-peer system.

Designing incentive mechanisms for video broadcast applications is more challenging than traditional file download applications, due to the real-time requirements. In particular, one solution in the file download context involves use of reputation based on past performance. However, this is feasible because the total time to download a file can often be long providing sufficient time to collect enough credits or build reputation. Further, file download users can tolerate slow download rates for a period of time by keeping the program running at the background. In contrast, users in video broadcast applications stay for shorter times, and will simply leave the system when the playback quality is not satisfying. A micro-payment mechanism may be a good solution that enables video broadcast users to cooperate. However, this often asks for a centralized broker for coordination that can hinder the scalability of a peer-to-peer systems. Systems like Coopnet [28] assume each node contributes as much bandwidth as it receives - however this does not consider the heterogeneity in node bandwidth.

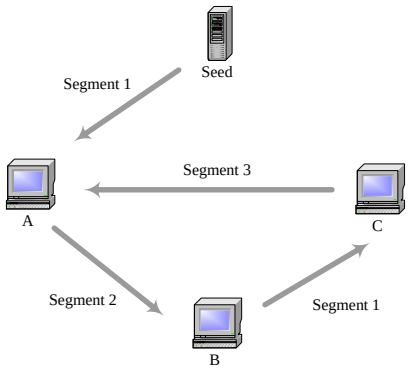

Fig. 6. An example of the tit-for-tat strategy.

BitTorrent-like applications adopt a *tit-for-tat* strategy to solve the incentive problem. Tit-for-tat is a highly effective strategy in game theory originally proposed for the iterated prisoner's dilemma. The strategy works well in peer-to-peer file download because the segments of a file are downloaded independently. As depicted in Fig. 6, peers A, B, and C download different segments from each other. This forms a feedback loop; for example, uploading segment 2 from A to B will be feedback to A by the upload of segment 3 from C to A, which stimulate peer A to cooperate. However, this approach does not trivially extend to video broadcast because of the timeliness requirements involved. The design of a scalable, light-weight incentive mechanism which can be incorporated into video broadcast application remains an open problem.

C. Access Bandwidth Scarce Regimes

For a peer-to-peer system to be self-sustaining, the contribution or upload bandwidth from nodes must exceed the bandwidth that nodes can receive. The incentive and fairness measurements, if implemented, can improve the overall upload bandwidth. However, a key challenge is that the asymmetric nature of nodes means that nodes behind DSL and cable can receive several hundreds of kilobits per second, but can fundamentally only donate less. We note this challenge is particularly important for streaming applications compared to other peer-to-peer applications. File download applications in such environments will simply see much slower delivery times, and given that applications like Skype are not bandwidth intensive, this is not much of an issue. Further, this issue may not be as critical in countries such as Japan and Korea where high-speed Internet connectivity is prevalent.

To formally characterize the resources available in the environment, a metric called the *Resource Index* was introduced in [8]. The *Resource Index (RI)* is the ratio of the number of receivers that the members in the group could *potentially sustain* to the number of receivers in the group for a particular source rate. The number of nodes that can be potentially sustained is the sum of the existing nodes and the number of free slots in the system. An RI less than 1 indicates not all participating nodes can receive the full source rate, an RI of 1 indicates the system is saturated, and as the RI gets higher, the environment becomes less constrained and it becomes more feasible to construct a good overlay tree. As reported in [8], several environments may see RI values lower than 1.

This issue can be handled in several ways. A first direction involves frameworks for application-level adaptation. A recent work [35] has considered resource-scarce regimes where nodes have heterogeneous upload bandwidth. The idea is to use a multi-tree framework, where not all nodes receive the full bandwidth. The amount of bandwidth a receiver is actually entailed to depends on the total contribution that it makes, thus ensuring nodes that contribute more receive better quality, yet all nodes achieve some basic quality. A second direction may involve using additional nodes not in the peer-to-peer system, called *waypoints*. For instance, one could imagine longer-lived peer-to-peer communities, where only a subset of nodes in the

community are actually present in any given broadcast. Then, it may be possible to leverage the bandwidth resources of other nodes not in the broadcast but present in the community.

Finally, we note that in addition to access bandwidth availability in the environment, the feasibility of constructing overlay structures may be further impacted by other factors such as connectivity restrictions posed by NATs and firewall, as we discuss in Section V-G.

D. Extreme Peer Dynamics and Flash Crowd

During a flash crowd, there is a large increase in the number of users joining the broadcast in a short period of time. This poses challenges for a peer-to-peer broadcast system as it has to rapidly assimilate the new peers into the distribution structure without significantly impacting the video quality of existing and newly-arrived peers. The opposite situation is when a large number of users leave a broadcast during a short period of time. The peer-to-peer broadcast system has to repair the delivery structure to minimize the service interruption. During a high churn situation, users arrive and depart at a very high rate, in which case the peer-to-peer broadcast system has to continue to adapt with the peer dynamics.

As discussed in [33], flash crowd and high churn situation are very common. The extreme scenario can be very difficult to handle. Consider the example of a popular concert broadcast that attracts 1 million users. If these users arrive within the the first 100 seconds of the concert, the peak arrival rate will be 10,000 peers per second. If the video quality is not good for the initial period, a user is more likely to quit. This not only represents a failure of the system to provide service to this particular user, but also generates a peer departure event, thus more churn in the system. Designing peer-to-peer video broadcast system that is robust to extreme peer dynamics is still an open research problem.

E. Support for Heterogeneous Receivers

Heterogeneity also exists in the download bandwidth – for example hosts behind Ethernet, dial-up and DSL have very different downloading capabilities. Supporting receivers at a single video rate is not appropriate, as it can either overwhelm slower receivers, or provide insufficient quality to powerful receivers. The need for such support is unique to streaming applications, and distinguishes them from BitTorrent-like systems.

ESM adopts a pragmatic approach to supporting receiver heterogeneity. Video is encoded at multiple bit-rates in parallel and is broadcast simultaneously, along with the audio stream. Unicast congestion control is run on the data path between every parent and child, along with a prioritized packet forwarding scheme. Audio is prioritized over video streams, and lower quality video is prioritized over higher quality video.

The design above involves overhead, when used with ordinary codecs. To address this, various streaming systems have proposed using scalable coding techniques such as layered coding [12], [25]. A cumulative layered coder, or scalable coder, generates a stream consisting of multiple layers, achieved through scaling frame rate, size, or quality. A receiver, depending on its capability, can subscribe to the base layer only with the basic playback quality, or subscribe to additional layers that progressively refine the reconstruction quality.

Recent proposals such as [5], [28] leverage another specialized coding algorithms called Multiple Descriptive coding (MDC), as also discussed in Section IV-B. An MD coder generates multiple streams (referred to as descriptions) for the source video. Any subset of the descriptions, including each single one, can be used to reconstruct the video. A simple implementation of MD coding can be achieved by splitting even and odd numbered frames. Advanced methods including interleaving of sub-sampled lattice, MD scalar quantization, and MD transform [39]. The descriptions are then distributed over multiple paths, preferably disjoint, to enhance robustness and to accommodate user heterogeneity.

While the scalable coding techniques hold promise, particularly considering that the H.264 Annex G/MPEG-4 scalable video coding is going to be finalized in 2007 [30], they are yet to see extensive deployment. Further, scalable coding has efficiency concerns because of the iterative motion estimation and transform for all the layers. Transporting the layers incurs bandwidth penalty as well, e.g., the extra bits for synchronizing layers. Multiple description coding requires each description to carry sufficient information about the original video. This can further reduce the compression efficiency. On the receiver's side, a scalable or MD video stream requires a high computation power to assemble and decode multiple layers. Further progress is required on these fronts before they can be used in actual peer-to-peer streaming systems.

F. Network Coding: Coding at Peers ?

A conventional assumption in many communication networks, in particular, the Internet, is that the intermediate nodes should do nothing on the data packets but forwarding. Network coding is a new tool in information theory that drastically breaks with this conventional assumption [2]. The fundamental insight in network coding is that if data can be encoded in intermediate nodes then the optimal multicast throughput can be achieved. While historically Forward Error Correction (FEC) or transcoding have been applied in certain network nodes, they are application-tailored services for individual streams only. Network coding instead treats coding as a network primitive and targets global network optimization.

Avalanche [18] is a typical example that applies random linear network coding in peer-to-peer file download, and shows that the throughput can be 2-3 times better with coded data blocks. Recent studies also show that network coding enhances the robustness, adaptability, and data availability of a peer-to-peer overlay, because the information is evenly spread in the coded data blocks [17].

The potential of network coding in peer-to-peer video broadcast has yet to be explored. There are additional issues arising from the unique features of streaming video. First, unlike file download, video streaming is loss-tolerant. With network

Child	Parent		
	Public	NAT	Firewall
UDP Transport			
Public	$\surd$	$\surd$	$\surd$
NAT	$\surd$	?$\star$	?
Firewall	$\surd$	?	?$\star$
TCP Transport			
Public	$\surd$	$\surd$	$\surd$
NAT	$\surd$	$\star$	$\times$
Firewall	$\surd$	$\times$	$\star$

TABLE II

CONNECTIVITY MATRIX. $\surd$ MEANS CONNECTIVITY IS ALWAYS POSSIBLE. ? MEANS CONNECTIVITY IS POSSIBLE FOR SOME CASES OF NAT/FIREWALL AND $\star$ MEANS CONNECTIVITY IS ONLY POSSIBLE IF THE NODES ARE IN THE SAME PRIVATE NETWORK.

REGION	USER NUM
CHINA	32217
HONG KONG	20725
UNITED STATES	3290
SPAIN	2989
KOREA	1834
CANADA	1707
GREAT BRITAIN	1326
TAIWAN	1213
FRANCE	1088
ITALY	1059
SINGAPORE	578
GERMANY	555
JAPAN	519
OTHERS	2163
TOTAL	71652

TABLE III

GEOGRAPHICAL DISTRIBUTION OF COOLSTREAMING USERS.

coding, however, available data blocks might not be decodable if one or more blocks are missing before playback deadline. Second, given the unbounded session time of live streaming, the buffer at each node has to be updated over time to remove obsolete data. This is different from bulk file download where the buffer is just allocated for the file with minimizing the filling up time being the key objective. Existing proposals suggest that a stream be divided into *generations* [7]. Clearly, coding efficiency improves with a longer generation, but startup delay increases if a new peer joins in the middle of a generation. These problems are further aggravated since video packets are of different importance and the streaming rate is not constant. Interactions with layered coding or multiple description coding can be even more complex.

G. Implementation Issues

Finally, to deploy real peer-to-peer broadcast systems over the Internet, there are many non-trivial implementation issues to be addressed. We now highlight three that we have encountered during building practical systems.

NATs and Firewalls. NATs and firewalls impose fundamental restrictions on pair-wise connectivity of nodes on an overlay, and may prohibit direct communication with one another. Whether communication is possible between two nodes depends on several factors such as the transport protocol (UDP or TCP), the particular kind of NAT/firewall (see [21] for a classification), and whether the nodes are located behind the same private network. Table II characterizes these restrictions for the different transport protocols, where columns represent parents and rows represent children. For example, communication is not possible between two NATed nodes using TCP unless they happen to be in the same private network. Given that in Internet environments, over 50% of nodes can be located behind NATs and firewalls, the connectivity constraints are a significant challenge to the viability of a peer-to-peer approach to video broadcast.

Transport Protocol. The transport protocol used in peer-to-peer broadcast systems has important implications. It has been long debated whether TCP is suitable for streaming media. Several research proposals have suggested use of TFRC as the transport protocol, which enables rate-smoothed TCP-friendly transmission and is supposed to suit streaming better. However, TCP is readily available, widely tested, and may often be engineered to work well, and thus is not necessarily a poor choice. Indeed, real implementations such as [8], [41] have employed TCP as the transport protocol. Another complicating factor in the choice of the transport protocol is they may have different levels of penetration of NATs and firewalls, and may be treated differently by various enterprise policies.

Startup delay and Buffer interaction. The startup and channel switching delays remain problems in peer-to-peer broadcast. A new peer may spend 10 to 15 seconds to join a peer-to-peer overlay, and take another 10 to 15 seconds to launch the media player and buffer certain data. The delay can be significantly longer for some unpopular channels, and will be further prolonged if using TCP and network coding. Also note that the existing peer-to-peer broadcast systems generally separates the streaming engine and the playback engine. Popular players such as Apple QuickTime, RealPlayer, and Microsoft MediaPlayer have been used for the latter, which simplifies system design and ensures compatibility and portability. Given each engine has its own buffer, an interesting question is whether the overall buffering time will be increased or not. While a naive design will certainly increase the latency, efficient use of this 2-stage buffer might deliver better performance, e.g., use the player's for smoothing and the streaming engine's for aggressive pre-fetching.

VI. DEPLOYMENT STATUS AND CHALLENGES

A. Deployment Status

Several peer-to-peer video broadcast systems are being built and deployed both by the research community, and by industry. Prominent research efforts include the CoolStreaming and ESM systems. Key industrial efforts include PPLive [52], TVAnts [44], TVUPlayer [45], GridMedia [40], and Zattoo [56]. Many broadcasts have attracted peak group sizes of thousands of participants – for example, the CoolStreaming system has had more than 50,000 concurrent users in the system and 25,000 users in one channel at peak times. Even larger user bases have been reported with other systems [20].

The deployment has reached a wide portion of the Internet users across multiple continents, in home, academic and commercial environments, and behind various access technologies. Table III summarizes the distribution of the IP addresses of the CoolStreaming users from 20:26 Mar 10 GMT to 2:14 Mar 14, 2005, totally around 78 hours. We believe this demonstrates the deployment potential of the peer-to-peer broadcast architectures - in contrast, the usage of the MBone [50] was more prevalent in academic institutions.

The service quality of these deployments has been promising, with most users reporting satisfactory viewing experience [8], [41], [42]. They have indicated that the current Internet has enough available bandwidth to support acceptable-quality streaming of 300-450 Kbps. Further, the experience has been that these results hold at larger scales: with higher user participation, the statistical results are even better. This validates the theoretical scaling law for peer-to-peer streaming [32]. Given that broadcasts often attract more simultaneously online users than that of individual file download, this also partially explains why peer-to-peer video broadcast systems can sustain a high and constant downloading speeds, though there are additional real-time constraints in scheduling as compared to BitTorrent.

B. Deployment Challenges

There is a general belief that streaming video will have a significant impact on the future Internet and will ultimately deliver its much anticipated revenues. Recently there have been several developments that seem to be promising in that direction: 1) Streaming video has gained popularity in enterprise applications, especially in distance education and online business; 2) Since the successful trial of research prototypes such as ESM and CoolStreaming, several peer-to-peer video broadcast platforms have proliferated to large scale with millions of subscribers online, as discussed previously; 3) The success of Youtube and its recent acquisition by Google also confirms the mass market interest in Internet video sharing, where peer-to-peer broadcast may serve as an underlying vehicle.

However, there are still major obstacles in mainstream adoption of peer-to-peer broadcast services. A key challenge pertains to the conflicting interests faced by network and content service providers, and the differences between how the Internet and the traditional video content providers operate. The Internet triumphs on placing intelligences at the edge of the network rather than inside the network core, which facilitate the rapid development and deployment of new services; Traditional video content providers however rely on dedicated networks, e.g., cable networks, that offer a few well-defined services with stringent and centralized control.

For Internet service providers, peer-to-peer video broadcast demonstrates the great flexibility of the Internet and certainly opens new business opportunities. However, peer-to-peer file download applications have already put unprecedented pressure on the network capacity. Video broadcast demands more resources, and it is known that some of the existing systems are aggressive in consuming bandwidth [3]. Internet service providers soon have to face the problem of re-provisioning their capacity and service, as well as to revisit the charging models for subscribers.

On the other hand, video content providers such as TV stations have to carefully evaluate this new type of service in order to protect their revenue in current program offering. It is still unclear what the appropriate revenue model for peer-to-peer broadcast with users scattered over the global Internet is. In fact, while users have positively commented on the existing deployment, whether they are willing to pay for the otherwise free streaming services today remains a question. The service fluctuation in terms of delay and in the worst case program disruption, which cannot be fully eliminated in the best-effort Internet environment, put challenges to any payment-based business.

Such problems are further complicated given that telecom and cable companies often have dual roles as both content and network providers, and there are also restrictions on their service offerings from government regulations. All these issues have to be properly addressed before the commercial-grade video broadcast using peer-to-peer services becomes a reality over the global Internet.

VII. Summary

Despite multiple unsuccessful starts, Internet video has come of age. In the very near future, video may become the dominant type of traffic over the Internet, dwarfing other types of traffic. Among the three video distribution modes: broadcast, on-demand streaming, and file download, broadcast is the most challenging to support due to the strong scalability and performance requirements. Peer-to-peer solutions represent the most promising technical approaches for Internet video broadcast due to the self-scaling property of this architecture.

In this article, we reviewed the state-of-art of peer-to-peer Internet video broadcast. On one hand, peer-to-peer solutions have shown great promise in supporting video broadcast, as witnessed by their increasingly widespread deployments. On the other hand, there are a number of key technical challenges that need to be overcome before the peer-to-peer solutions can approach the service quality of conventional broadcast and cable TV. In the near term, most of the challenges have to do with the limited amount of access capacity in the Internet. As broadband networks become more ubiquitous and higher-speed, the issues of peer dynamics and incentive will become more important. In the longer term, a key challenge lies in the tussle among content service providers, consumers, and network service providers. To be successful, peer-to-peer solutions not only need to provide compelling services to content providers and consumers, but also need to address the concerns of network service providers.

References

[1] E. Adar and B. A. Huberman, “Free riding on gnutella”, *First Monday*, 2000.

[2] R. Ahlswede, N. Cai, S. Li, and R. Yeung, “Network information flow”, *IEEE Transactions on Information Theory*, vol. 46, pp. 1204-1216, 2000.

[3] S. Ali, A. Mathur and H. Zhang, "Measurement of commercial peer-to-peer live video streaming", in *Proc. Workshop on Recent Advances in P2P Streaming*, Waterloo, ON, Canada, August 2006.

[4] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast", in *Proc. ACM SIGCOMM'02*, Pittsburgh, PA, August 2002.

[5] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron and A. Singh, "SplitStream: High-bandwidth multicast in cooperative environments", in *Proc. ACM SOSP'03*, New York, USA, October 2003.

[6] Y. Chawathe, S. McCanne, and E. A. Brewer, "An architecture for Internet content distribution as an infrastructure service", http://yatin.chawathe.com/-yafin/papers/scattercast.ps

[7] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding, in *Proc. 41st Allerton Conf. Communication, Control and Computing*, Monticello, IL, October 2003.

[8] Y.-H. Chu, A. Ganjam, T.S. E. Ng, S. G. Rao, K. Sripanidkulchai, J. Zhan, and H. Zhang, "Early deployment experience with an overlay based Internet broadcasting system", in *Proc. USENIX Annual Technical Conference*, June 2004.

[9] Y.-H. Chu, S. G.Rao, and H. Zhang, "A case for end system multicast", in *Proc. ACM SIGMETRICS'00*, June 2000.

[10] Y.-H. Chu, S. G.Rao, S. Seshan, and H. Zhang, "Enabling conferencing applications on the Internet using an overlay multicast architecture", in *Proc. ACM SIGCOMM'01*, August 2001.

[11] B. Cohen, "Incentives Build Robustness in BitTorrent", in *Proc. P2P Economics Workshop*, Berkeley, CA, 2003.

[12] Y. Cui and K. Nahrstedt, "Layered peer-to-peer streaming", in *Proc. NOSSDAV'03*, June 2004.

[13] S. Deering and D. Cheriton, "Multicast routing in datagram internetworks and extended LANs", *ACM Transaction on Computer Systems*, vo. 8, no. 2, pp. 85-110, May 1990.

[14] H. Deshpande, M. Bawa, and H. Garcia-Molina, "Streaming live media over peer-to-peer network", *Technical Report*, Stanford University, 2001.

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

[16] P. Francis, "Yoid: Extending the Interent multicast architecture," *http://www.icir.org/yoid/*.

[17] C. Gkantsidis, J. Miller, and P. Rodriguez, "Comprehensive view of a live network coding P2P system", in *Proc. ACM SIGCOMM/USENIX IMC'06*, Brasil, October 2006.

[18] C. Gkantsidis and P. Rodriguez, "Network coding for large scale content distribution", in *Proc. IEEE INFOCOM'05*, Miami, FL, Mar. 2005.

[19] M. Heffeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava, "PROMISE: Peer-to-peer media streaming using CollectCast", in *Proc. ACM Multimedia'03*, Berkeley, CA, November 2003.

[20] X. Hei, C. Liang, J. Liang, Y. Liu, and K.W. Ross, "Insights into PPLive: A measurement study of a large-scale P2P IPTV system", in *Proc. Workshop on Internet Protocol TV (IPTV) services over World Wide Web in conjunction with WWW2006*, Edinburgh, Scotland, May 2006.

[21] C. Huitema, J. Rosenberg, J. Weinberger, and R. Mahy, "STUN - Simple traversal of UDP through network address translators", *IETF-Draft*, December 2002.

[22] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: High bandwidth data dissemination using an overlay mesh", in *Proc. ACM SOSP'03*, New York, USA, October 2003.

[23] D. Li, J. Wu, Y. Cui, and J. Liu, "QoS-aware Streaming in Overlay Multicast Considering the Selfishness in Construction Action", in *Proc. IEEE INFOCOM'07*, Anchorage, Alaska, USA, May 2007.

[24] X, Liao, H, Jin, Y, Liu, L, M. Ni, and D, Deng, "AnySee: Scalable live streaming service based on inter-overlay optimization", in *Proc. IEEE INFOCOM'06*, 2006.

[25] J. Liu, B. Li, and Y.-Q. Zhang, "Adaptive video multicast over the Internet", *IEEE Multimedia*, vol. 10, no. 1, pp. 22-31, Jan./Feb. 2003.

[26] N. Magharei, R. Rejaie, and Y. Guo, "Mesh or Multiple-tree: A comparative study of live P2P streaming approaches", in *Proc. of IEEE INFOCOM'07*, Anchorage, Alaska, USA, May 2007.

[27] V. N. Padmanabhan, H. Wang, P. Chou, "Resilient peer-to-peer streaming", in *Proc. of IEEE ICNP'03*, November 2003.

[28] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking", in *Proc. NOSSDAV'02*, USA, May 2002.

[29] V. Pai, K. Tamilmani, V. Sambamurthy, K. Kumar, and A. Mohr, "Chainsaw: Eliminating trees from overlay multicast", in *Proc. The 4th International Workshop on Peer-to-Peer Systems (IPTPS)*, February 2005.

[30] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the Scalable Video Coding Standard", *IEEE Transactions on Circuits and Systems for Video Technology*, to appear, 2007.

[31] T. Small, B. Liang, and B. Li, "Scaling laws and tradeoffs in peer-to-peer live multimedia streaming", in *Proc. ACM Multimedia'06*, Santa Barbara, CA, October 2006.

[32] T. Small, B. Liang, and B. Li, "Scaling laws and tradeoffs in peer-to-peer live multimedia streaming", in *Proc. ACM Multimedia'06*, Santa Barbara, CA, October 2006.

[33] K. Sripanidkulchai, A. Ganjam, B. Maggs, and H. Zhang, "The feasibility of supporting large-scale live streaming applications with dynamic application end-points", in *Proc. ACM SICOMM'04*, August 2004.

[34] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for Internet applications", in *Proc. of ACM SIGCOMM'01*, San Diego, CA, August 2001.

[35] Y.-W. Sung, M. Bishop and S. G. Rao, "Enabling contribution awareness in an overlay broadcasting system," in *Proc. ACM SIGCOMM'06*, Pisa, Italy, September 2006.

[36] R. Tian, Q. Zhang, Z. Xiang, Y. Xiong, X. Li, W. Zhu, "Robust and efficient path diversity in application-layer multicast for video streaming", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 8, pp. 961-972, August 2005.

[37] V. Venkataraman, P. Francis, and J. Calandrino, "ChunkySpread: Multi-tree unstructured peer-to-peer multicast", in *Proc. The 5th International Workshop on Peer-to-Peer Systems*, February 2006.

[38] F. Wang, Y. Xiong, and J. Liu, "mTreebone: A hybrid tree/mesh overlay for application-layer live video multicast", in *Proc. The 27th IEEE International Conference on Distributed Computing Systems (ICDCS'07)*, Toronto, Canada, June 2007.

[39] Y. Wang, A. R. Reibman, S. Lin, "Multiple description coding for video delivery", *Proceedings of the IEEE*, vol. 93, no. 1, pp. 57 - 70, January 2005.

[40] M. Zhang, J.-G. Luo, L. Zhao, and S.-Q. Yang, "A peer-to-peer network for live media streaming - using a push-pull approach," in *Proc. ACM Multimedia'05*, November 2005.

[41] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "CoolStreaming/DONet: A Data-driven Overlay Network for Peer-to-Peer Live Media Streaming", in *Proc. INFOCOM'05*, Miami, FL, USA, March 2005.

[42] X. Zhang, J. Liu, and B. Li, "On large-scale peer-to-peer live video distribution: CoolStreaming and its preliminary experimental results", in *Proc. IEEE Multimedia Signal Processing Workshop (MMSP'2005)*, Shanghai, China, October 2005.

[43] S. Q. Zhuang, B. Y. Zhao, and A. D. Joseph, "Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination", in *Proc. NOSSDAV'01*, New York, June 2001.

[44] http://cache3.tvants.com/

[45] http://tvunetworks.com/

[46] http://www.akamai.com

[47] http://www.bittorrent.com

[48] http://www.emule-project.net

[49] http://www.gnutella.com

[50] http://www.mbone.net

[51] http://www.napster.com

[52] http://www.pplive.com/en/index.html

[53] http://www.skype.com

[54] http://www.techweb.com/wire/networking/183700547

[55] http://www.thewhir.com/marketwatch/aol070505.cfm

[56] http://www.zattoo.com/