Challenges and Approaches in Large-Scale P2P Media Streaming

W.-P. Ken Yiu, Xing Jin, and S.-H. Gary Chan
Hong Kong University of Science and Technology

Large-scale multimedia streaming over the Internet requires an enormous amount of server and network resources. Traditional client–server approaches allocate a dedicated stream from the server for each client request, which is expensive and doesn't scale well. By using end hosts' huge bandwidth and computational capacity, peer-to-peer technologies shed new light on media streaming applications' development. Yet, locating supplying peers and content delivery path maintenance are two major challenges in this area.

As the bandwidth delivered to users increases, the market expects a growing interest in networked multimedia streaming applications. For example, emerging services might include on-demand video streaming that lets clients choose and watch favorite movies anytime; Internet Protocol television (IPTV) that provides more functional, interactive TV services; and digital video libraries that store documentaries that users can access online.

The simplest delivery technique for such applications is to allocate server and network resources for each client request. However, because of the heavy server load and limited network bandwidth at the server side as well as temperamental intercontinental links, this client–server approach doesn't scale well. (See the "Related Solutions in Media Streaming" sidebar for more details.)

Peer-to-peer technologies have emerged as a powerful and popular paradigm for many scalable applications, such as multicasting and file sharing among users all over the world (for example, see http://www.bittorrent.com).[1] However, providing streaming service over P2P networks is still a challenging task because of their inherent instability and unreliability. This article investigates two major challenges in providing P2P media streaming—locating supplying peers and maintaining content delivery paths—and compares the proposed approaches for tackling each problem.

P2P media streaming

In a P2P system, cooperative peers self-organize into an overlay network via unicast connections. Figure 1 (page 52) demonstrates a P2P media streaming system. Each peer, or *overlay node*, in an overlay network acts as an application-layer proxy, caching and relaying data for other peers. In this case, routers are only required to forward unicast packets, while end hosts take all interior and leaf positions in the multicast tree. (Note the simple client–server architecture is also an overlay tree with only the video server as the interior node.) Hence, P2P systems don't require specialized network infrastructure support.

In a P2P media streaming system, one or multiple supplying peers who have all or part of the requested media can forward the data to the requesting peers. In turn, the requesting peers can become supplying peers for other requesting peers. Because each peer contributes its own resources (storage and network bandwidth) to the system, the whole system's capacity is vastly amplified compared to the client–server architecture.

Research has shown that it's feasible to support large-scale media streaming over the Internet using a P2P approach,[2] but such systems still confront some design challenges:

- *Dynamic uptime*. In P2P networks, peers don't always stay online in the system. Supplying peers might suddenly crash or leave ungracefully. In this case, the requesting peers need to find new supplying peers to replace the failed ones. Therefore, the system should be robust enough to withstand such node failures.

- *Limited and dynamic peer bandwidth*. Unlike powerful video servers, peers have limited bandwidth capacities. Each supplying peer might only be able to support a few requesting peers (or multiple supplying peers are required to support one requesting peer). Also, the available bandwidth of supplying peers might fluctuate unexpectedly. Hence, the system should be able to adaptively adjust each supplying peer's sending rate to keep the streaming quality at requesting peers unaffected.

To deal with these challenges, researchers have proposed various solutions.[3,4] Some real-world systems (such as http://www.ppstream.com or http://www.pplive.com) have also been deployed to provide live media streaming services using the P2P approach.[5]

1070-986X/07/$25.00 © 2007 IEEE

Published by the IEEE Computer Society

Related Solutions in Media Streaming

Researchers have proposed many IP-multicast-based techniques such as periodic broadcasting in the last decade.[1,2] In IP multicast, routers are interior nodes in a multicast tree and are responsible for forwarding data down the tree.[3] All end hosts are leaves in the multicast tree. This approach is highly scalable because an infinite number of users can efficiently share a single channel. Unfortunately, applications of these techniques for worldwide multimedia streaming services remain limited because of a lack of widely deployed IP-multicast-capable networks.

Figure A1 depicts a simple video-on-demand (VoD) system architecture. A video server, connected to the Internet, is the origin of the videos. When a client wants to watch a movie, it contacts the server, and the server delivers the requested video to the client using a dedicated stream. In this case, even if the server has Internet access with a bandwidth of 1 gigabit per second (Gbps), it can only support up to approximately 3,000 clients with a streaming rate of 300 kilobits per second (Kbps). Because IP multicasting by the server is only scoped at local networks, a need exists for a more scalable solution that doesn't rely on network-layer multicast.

Another technique for delivering multimedia worldwide is called content delivery networks (CDN). CDN relies on a set of geographically distributed proxies/gateways over the Internet (for example, see http://www.akamai.com). As Figure A2 shows, streaming media is cached in the dedicated proxy servers, which are statically deployed beforehand. These proxies are connected in an overlay network. When a user requests a streaming video, the closest proxy server, instead of the origin, handles the request. This solution can provide worldwide streaming services; however, its deployment and maintenance costs are too expensive for small content providers.

References

1. Y. Guo et al., "Seamless Workload Adaptive Broadcast;" http://www.pv02.org.
2. K. Hua, Y. Cai, and S. Sheu, "Patching: A Multicast Technique for True Video-on-Demand Services," *Proc. ACM 6th Int'l Conf. Multimedia*, ACM Press, 1998, pp. 191-200.
3. S. Deering, "Multicast Routing in Internetworks and Extended LANs," *Proc. ACM Special Interest Group on Comm.* (Sigcomm), ACM Press, 1988, pp. 55-64.

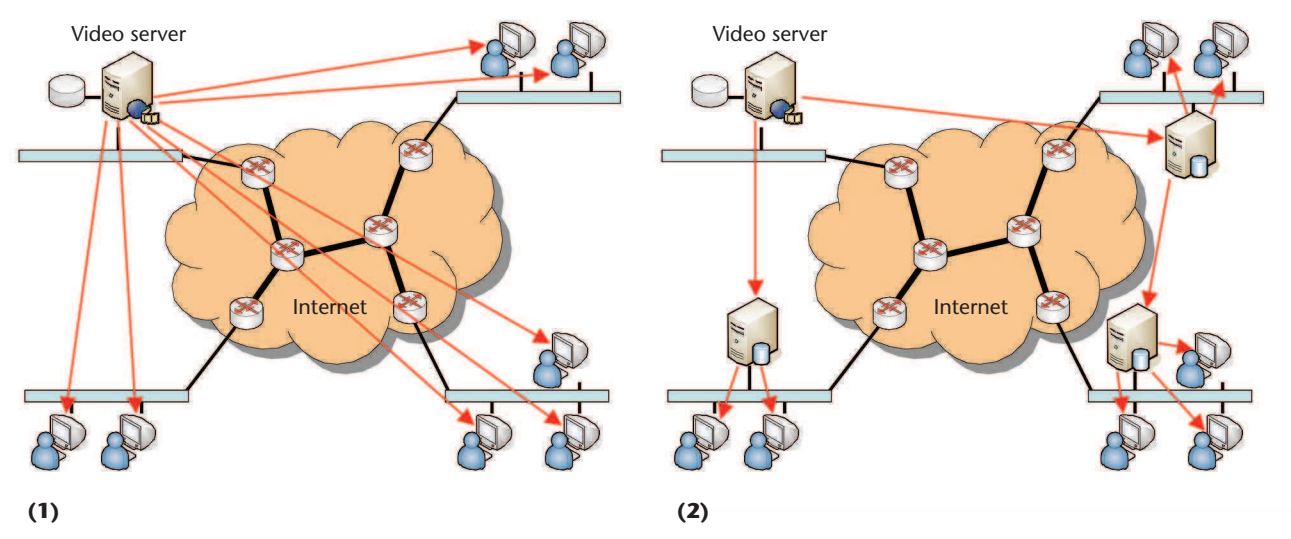

Figure A. Various video delivery techniques: (1) simple unicasts and (2) a content delivery network.

In the client–server approach, say YouTube.com, if the dedicated server is far away from the client, the bandwidth is usually too small to deliver the media content smoothly. This leads to users enduring frequent pauses while the buffer fills up. In P2P systems, on the other hand, each supplying peer works as a server, and clients choose a good server for the best performance. Because supplying peers are distributed over the network, finding a close supplying peer with enough bandwidth is critical to the streaming quality.

Locating supplying peers

Locating supplying peers is a challenge in a P2P media streaming system because each requesting peer must find supplying peers with enough bandwidth and preferably low latency to achieve a good service quality. Some common techniques for locating supplying peers in such

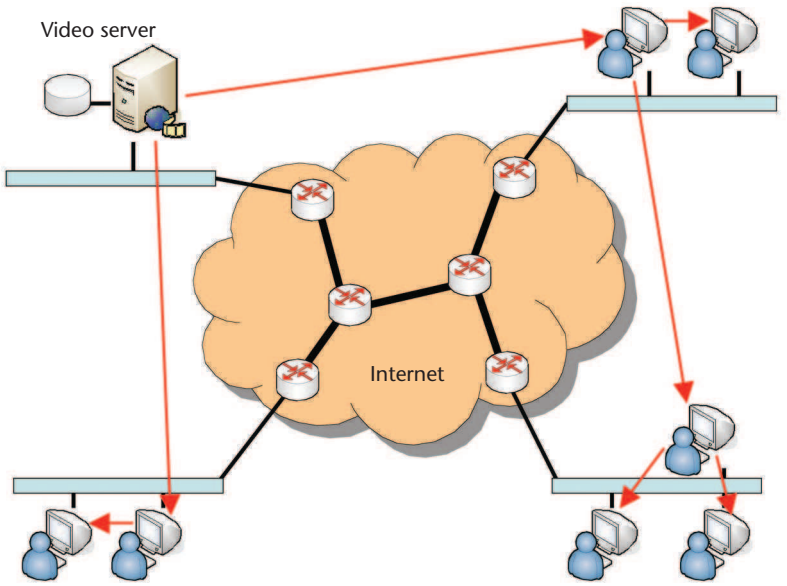

Figure 1. An overlay multicast video delivery technique for a peer-to-peer (P2P) media streaming system.

systems include a centralized directory, hierarchical overlay structure, distributed hash table (DHT) based approach, controlled flooding, and gossip-based approach.

Centralized directory

The simplest and most commonly used method for locating peers is to maintain a centralized directory of all peers in a directory server (such as http://www.ppstream.com or http://www.pplive.com).[6,7] All required peer information, including its network address, available bandwidth, fan-out degree, and starting point of access (for video-on-demand [VoD] systems), are in a directory server with a well-known IP address. The server can also keep the global overlay topology among peers.[6,8,9] Figure 2 demonstrates the flow of control messages in such a system.

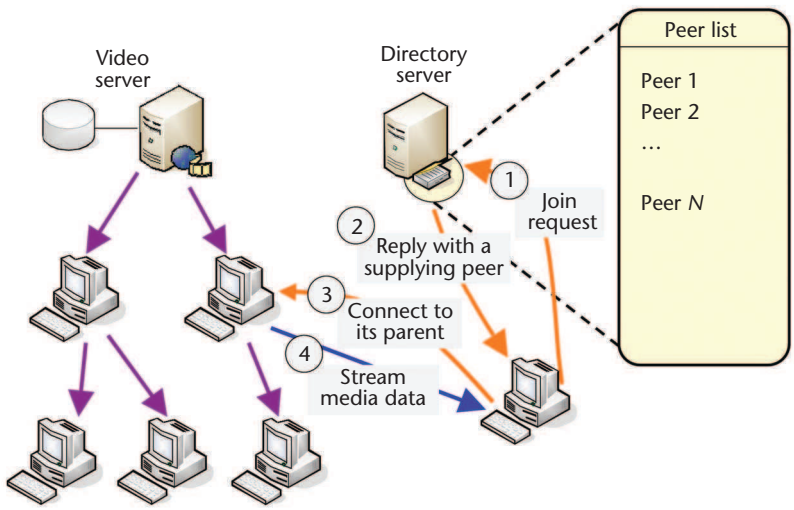

Figure 2. Peer location using centralized directory approach.

In this system, a new requesting peer's request is first directed to the directory server. Upon receiving a user request, the directory server selects the most suitable supplying peers from the stored peer list for the new user, according to its network address and the requested media. For example, a peer with large available bandwidth and a network location close to the requesting peer is chosen by the directory server to serve the requesting peer. If a user wants to leave the system, he or she must signal the directory server with a `LEAVE` message to clear his or her entry in the directory.

The advantages of this approach are its easy implementation and simple deployment. Centralization also greatly simplifies the join mechanism and consequently makes the join and leave procedures quick.

However, given N peers in the system, the directory server must maintain $O(N)$ states, which might overload the server when N is large. Furthermore, if a peer is unable to send a `LEAVE` message to the directory server (such as during a node failure), its state remains in the directory. To handle this problem, the peers must refresh their status at the server using periodic *keep-alive* (or *heartbeat*) *messages* or similar mechanisms. Transmitting these $O(N)$ keep-alive control messages also incurs high-bandwidth consumption at the server.

Another system weakness is that the directory server becomes a single point of failure. While the directory server is down, users can no longer join the system. Researchers[9] have argued that the directory server is usually also the source of data—such as for a video server. Hence, if the server (source) fails, it might not matter whether the directory is down.[9]

Hierarchical overlay structure

Another technique for locating supplying peers is to adaptively search over a hierarchical overlay structure built among the system's peers.[10–12] In this system, peers are organized into a hierarchical overlay structure such as an overlay tree. A newly joined client first contacts that overlay's rendezvous point (usually the data source). The rendezvous point then returns a list of connected peers $\{P_1, P_2, \ldots, P_k\}$ down one level in the hierarchy to the new client. The new client probes each peer in the list and finds out the most suitable peer P_x (according to the protocol's peer-selection criteria). Next, the new client con-

tacts P_x. Similarly, upon receiving the request, P_x replies with a list of connected peers down one level in the hierarchy. The new client probes each of them and selects the best one. The process repeats until the new client reaches a position in the structure where it can receive the required content with good quality of service.

Peers in Nice and Zigzag systems cooperatively organize themselves into a logical hierarchy of clusters.[11,12] As Figure 3 shows, clusters are managed into multiple levels using distributed algorithms. Each cluster has a cluster leader that's responsible for monitoring its cluster membership and is a member of a cluster in the upper level. Hence, some peers are cluster leaders in multiple levels. Cluster sizes can be between k and $3k$ (k is a system parameter) and are maintained using merge and split algorithms for bounding the out-degree of each peer. There's only one cluster in the topmost level where the source of the media resides.

To join the system, a newcomer first contacts the rendezvous point (the leader of the topmost cluster in Figure 3, labeled "F"). With that cluster's peer list attached in the leader's reply, the newcomer measures the distances between itself and all the peers in the list. Then, it selects the closest peer, which is a cluster leader in the lower level. After that, the newcomer sends another request to that leader. Again, that closest leader replies with its cluster member list. The probing process repeats until the new client finds its appropriate position in the architecture. By this successive probing, nearby peers are grouped together, making the data transmission based on that structure efficient.

This approach distributes the peer-management load over all the system's peers. For example, in contrast to $O(N)$ states maintained by the central server in a directory-based scheme, each peer in Zigzag only maintains $O(\log N)$ states. It also eliminates the single point of failure. Because the searching is based on the overlay structure, the maintenance of the structure among peers is critical to the searching performance. The control protocol for overlay maintenance hence must be resilient to node failure.

In Nice, for instance, members in the same cluster periodically exchange heartbeat messages to detect member failure. However, such maintenance protocols can complicate the system implementation. Because of frequent VCR-like operations, it's more challenging for VoD systems to maintain such a structural overlay for peer organization and data delivery. Therefore, this technique is more suitable for live streaming.

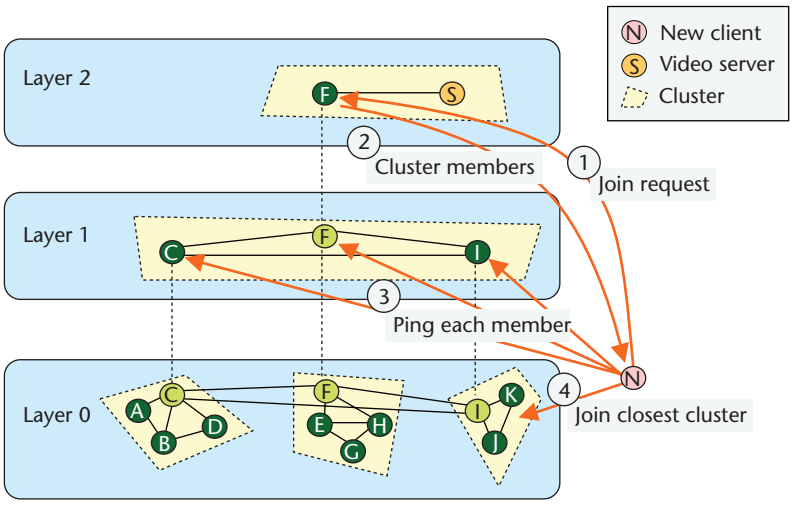

Figure 3. Join algorithm in the Nice architecture.

DHT-based approach

Another distributed approach for locating supplying peers is based on a DHT,[13,14] a common P2P searching technique, usually for locating nodes that store a desired object (such as a file) in P2P networks.[15,16] In DHT, each peer is assigned a peer ID by hashing its own IP address using a common known hash function such as SHA-1,[17] and each object is also associated with a key in the same space of peer IDs by hashing the object itself. The peer with an ID equal to the hashed key is responsible for storing the object's location (or the actual object). The primitive functions `PUT` and `GET` are available in DHT, as in a conventional hash table data structure. With an object's hashed key, a query for the object is routed through several nodes in the DHT to the node responsible for storing the object. There's a routing table maintained at each node in the DHT, based on which peers route queries. The routing load is also evenly distributed over all the peers in the DHT.

SplitStream is a P2P live video streaming system. It organizes its peers into multiple overlay multicast trees.[14] To achieve better forwarding load balancing among nodes, SplitStream assigns each node to be an internal node at most in one overlay tree and leaf nodes in all other trees. It uses Pastry, a DHT protocol, to locate parents and assign positions of nodes in each tree.[16] SplitStream splits the content into multiple stripes. Each of the stripes has an identifier stripe ID starting with a different digit. Each stripe is multicast in its own designated tree. All interior

nodes of a designated tree must have node IDs sharing the same prefix with the associated stripe ID. Thus, these interior nodes are restricted to becoming leaf nodes in other overlay trees.

A node's parent in a tree is the first node in the routing path from that node to the node with a node ID equal to the tree's stripe ID. Hence, the DHT routing protocol (Pastry) intrinsically determines the locations of each node's parents in SplitStream. The multicast tree is then implicitly constructed by merging the paths from all the nodes to the root. This algorithm can potentially violate the degree bounds of interior nodes, and SplitStream applies several heuristic methods to redistribute the data-forwarding load among nodes. Using DHT protocol to locate supplying peers takes advantage of well-developed DHT protocols, which are scalable and offer good load balancing.

Related work[15,16] has proven that query messages are routed through only $O(\log N)$ nodes for each lookup, and each node only needs to maintain $O(\log N)$ states in its routing table. Also, the load of routing requests is evenly distributed over all the peers in the network. Furthermore, using a well-developed DHT protocol simplifies the system's design and implementation. The system then can focus on the media streaming functionalities and needn't deal with the complicated peer management.

Other approaches and comparisons

Several other approaches exist for locating supplying peers. In GnuStream, locating supplying peers is based on a P2P search system called Gnutella.[18] Gnutella's searching process is based on controlled flooding of the query over the overlay mesh built among the system's peers. The query from the requesting peer is sent to all its neighboring peers. Upon receiving the query, the peer rebroadcasts the query to all its neighbors except the one that sent the query.

The query message is associated with a time-to-live (TTL) value. Each broadcasting by a peer decreases the TTL by one. This query broadcasting continues until the TTL becomes zero. The peers who receive the query and possess the requested object will reply with the query's origin, indicating the requested object's presence. Depending on the degree of connectivity among peers, the flooding of queries can generate a lot of network traffic. Besides, objects located out of the search scope (which the TTL determines) wouldn't be found in the system.

In the CoolStreaming system, a node needs to search for peers called *partners*, which the node collaborates with to download the requested live streaming media.[19] Each CoolStreaming node keeps a partial list of active peers, of size $O(\log N)$ in the system. A newly arrived node first contacts the server, which redirects the new node to a randomly selected peer from its partial membership list. The list of partners is then obtained from the selected peer. To maintain a partial list at each node, the system employs a distributed gossip-based membership management protocol called the Scalable Membership Protocol (SCAMP).[20] The protocol is highly scalable and provides a uniform partial view at each node. To accommodate overlay dynamics, each node periodically exchanges membership messages with neighbors to announce its existence. Because this technique doesn't differentiate users at various play points of the media, it's only applicable for live media streaming systems.

Hierarchical overlay structure, DHT-based, controlled flooding, and gossip-based approaches are more scalable than a centralized directory in terms of the number of states they must maintain on the server. However, a centralized directory is the easiest to implement and can respond to requests quickly. Therefore, most systems assume using a centralized directory to locate peers. Table 1 summarizes the properties of the aforementioned approaches for locating supplying peers.

Content delivery path maintenance

In P2P multimedia streaming, the audio or video stream consists of small data blocks. Each block has a numerical sequence number, which receivers can use to correctly reorder blocks for playback. Because there's a playback deadline for each block, data delivery must be timely. Due to the dynamic nature of end-host uptime and limited bandwidth, it's challenging to provide robust and scalable transmission in such systems.

Tree-based multicast

The most intuitive and common way to deliver multimedia content to a large group of peers is to build a single multicast tree, in which all interior nodes and leaf nodes are the peers.[13,21] As Figure 4a shows, an interior node must forward data to other nodes, while a leaf node doesn't need to forward any received data. An overlay tree among peers can be constructed in either a centralized or distributed manner.

One of the issues in overlay tree construction

Table 1. Comparisons between various approaches for locating supplying peers.

Approach	Scalability	Single Point of Failure	Search Guarantee	Server States	Peer States	Implementation
Centralized directory	Low	✓	✓	$O(N)$	$O(1)$	Simplest
Hierarchical overlay structure	High	✗	✓	$O(1)$	$O(\log N)$	Most difficult
DHT based	High	✗	✓	$O(1)$	$O(\log N)$	Medium
Controlled flooding	Medium	✗	✗	$O(1)$	$O(1)$	Medium
Gossip based	High	✗	✓	$O(\log N)$	$O(\log N)$	Medium

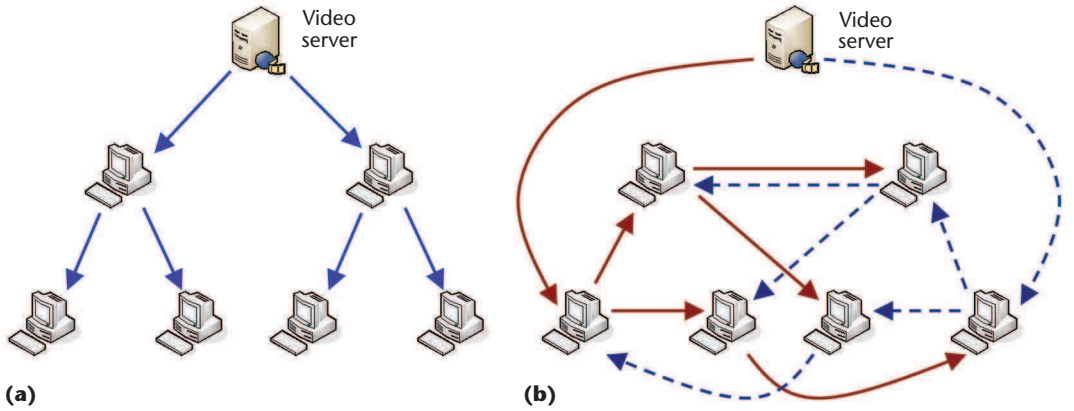

Figure 4. Tree-based overlay multicast. (a) Single tree. (b) Multiple trees.

is to make sure the tree is loop free. It's easier for a central server to construct a loop-free overlay tree because the central server knows all the peer connections.[9] In contrast, maintaining a distributed overlay tree requires a careful control protocol design.[21]

One example, oStream, is a P2P on-demand media streaming system.[21] The authors developed a temporal dependency model; with a buffer on each client, content recently received and buffered at a client C_i can be used to serve another client C_j if C_j's requested media is within C_i's buffer. C_i is the predecessor of C_j, and C_j is the successor of C_i. Based on this model, the system constructs a directed graph called a *media distribution graph*, in which each edge is defined from a predecessor to its successor with that edge's transmission cost. Then, the minimal spanning tree—called a *media distribution tree* (MDT)—on the MDG can be found by the system using the minimal spanning tree algorithm to minimize the overall transmission cost. The system can set the transmission cost on each edge to be the latency of the corresponding link to minimize the overall latency of the multicast tree.

Figure 5 (next page) illustrates an example where S is the video server (media source) that can handle all the client requests. With a larger buffer size, a client can also serve more clients. R_2 can serve R_3 and R_4 because its buffer is large enough, but R_2's buffer doesn't contain the period of media from o_3 to o_2. In this case, R_3 and R_4 can first obtain from the server S, the periods o_3 to o_2 and o_4 to o_2, respectively, and then switch their parents to R_2.

This technique, well-known in IP-multicast-based solutions, is called *patching*.[22] Creating and maintaining a shortest-path multicast tree provides an optimized architecture for real-time content delivery.

Still, a tree structure suffers from several disadvantages:

- *It isn't fair.* All the leaf nodes don't need to forward data, but interior nodes are required to forward data to at least two children nodes (otherwise, the data paths become long). The number of leaf nodes increases much faster than the number of interior nodes. Because only the interior nodes carry the forwarding load, the system load distribution becomes unbalanced.

- *It's fragile and prone to severe service disruption.* Each node is connected to its only parent with a single link. If the parent suddenly halts or the link is broken because of congestion, the child node and all its descendants immediately suffer from data shortage (that is, a

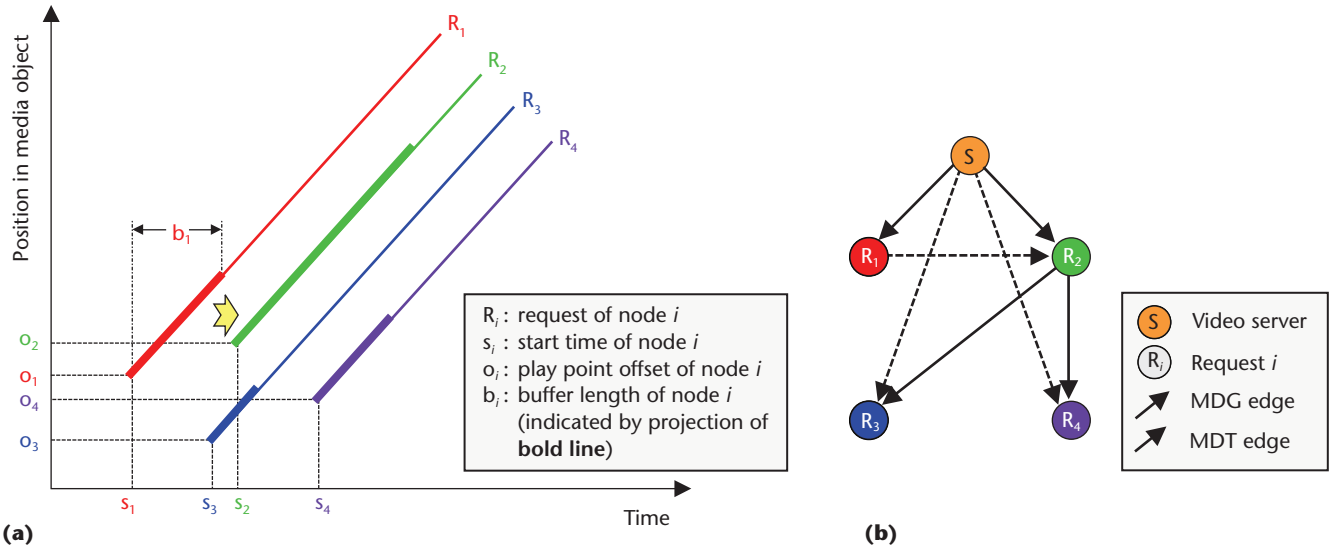

Figure 5. Illustration of a temporal dependency model and the construction of a media distribution tree. (a) Temporal dependency model. (b) Media distribution graph (MDG) and media distribution tree (MDT).

buffer underflow) and a recovery scheme becomes necessary.

- *An interior node might not be able to offer high-bandwidth video streaming to its children because of its limited bandwidth.* Furthermore, bandwidth is guaranteed to be monotonically decreasing as it goes down the tree; therefore, a node many hops away from the source might not receive enough bandwidth even though its parent has large outgoing bandwidth. Besides, for interactive VoD systems, it's even more challenging to maintain the overlay tree due to frequent VCR operations. Hence, supporting efficient interactive VoD streaming in P2P networks is still an active research topic.

Due to these weaknesses, the tree-based multicast approach is only suitable to small group streaming applications such as videoconferencing or multiparty gaming.

Multiple trees

To circumvent the drawbacks of single multicast tree systems, researchers have proposed building multiple overlay trees for delivering multimedia content to users.[8,14] For example, in Figure 4a, the multicast is accomplished by the peers using a single overlay tree in which each interior node forwards data to two children nodes. An interior node's forwarding load is two. (The leaf nodes don't forward any data.)

On the other hand, as Figure 4b shows, the video server splits the video stream into two equal-sized substreams, and each substream is delivered along a distinct overlay tree. Because each node forwards at most two substreams, a node's maximum forwarding load in this system is only one (or $1/2 \times 2$). This approach distributes the forwarding load among nodes and exploits the bandwidth of the links among end hosts, which is unused in the single tree approach. Therefore, the system can deliver streaming media at a higher bit rate (or higher quality) and the system's throughput increases.

The CoopNet system forms multiple trees at the central server,[8] which adds a new node to one tree in which the node is going to be an interior node. The server chooses that tree either randomly among all the trees or deterministically according to the algorithm. In the deterministic case, the server picks the tree with the least number of interior nodes to balance the numbers of interior nodes across different trees. After that, the nodes will be added as leaf nodes in all the remaining trees.

SplitStream also manages the nodes into multiple *interior-node-disjoint trees*—that is, a node can be an interior node in at most one tree—but in a distributed manner with the assistance of the Pastry DHT protocol.[14,16] With the use of multiple description coding (MDC), a node departure only causes the loss of a single description (out of M encoded descriptions) on average because the departed node is likely a leaf node in most trees.[23,24] Because MDC allows decoding of any received subset of the M descriptions, the node can still decode the media even if a few descriptions are lost. Therefore, the system becomes more resilient to node failure.

In addition, the ability to decode an arbitrary

subset of descriptions means peers can join more trees to obtain more descriptions if they have more bandwidth. Using this approach, bandwidth heterogeneity among peers can be addressed if peers are free to join any number of system trees. However, a node must wait for all required descriptions to arrive before decoding. As a result, the delivery delay is the maximum delay among all the overlay path delays from the source to the node.

The major drawback of this design is that it's hard to optimize all the trees at the same time (even in a centralized approach) because each node takes part in all the trees.

Pull-based gossiping

As opposed to tree-based approaches, gossip protocols (also known as *epidemic protocols*) don't rely on any regular overlay structure to deliver data to a user group.[6,19] In traditional gossip protocols, a node randomly picks a subset of target nodes and sends recently received data to them. At the same time, the node receives data from other nodes. The random choice of gossip targets provides resilience to random failures. Research has shown that taking the number of target nodes of the order of the system size's logarithm is necessary for group members to receive the disseminated data with a high probability.[20] (More specifically, if there are N nodes and each node gossips to $\log(N) + k$ other nodes, where k is a constant, the probability that everyone gets the data converges to $e^{-e^{-k}}$.)

For high-bandwidth media streaming applications, however, this kind of push-based gossiping isn't suitable because of the excessive data duplication in the random dissemination process. Instead, a pull-based gossip protocol is adopted for P2P media streaming.

In this system, peers periodically exchange data availability information with their neighbors. Upon receiving the data availability information from a peer x, a peer y chooses the data segments it doesn't possess and sends a request indicating the demanded data segments to x. Then, x delivers the requested data segments to y. This pull-based approach greatly reduces the redundancy in data delivery led by push-based gossip protocols.

There have also been other proposals for conquering the data-redundancy problem. For example, the erasure coding scheme used in PeerStreaming applies forward error correction (FEC) to make the probability of data duplication extremely low.[25]

Another solution is to allow re-encoding by intermediate nodes. This technique—used by the rStream system[26]—is called network coding.[27] There, intermediate nodes decode the encoded data blocks and recode them into another set of blocks before sending them out. This scheme greatly reduces the chance of getting the same encoded block from the network.

CoolStreaming employs pull-based gossiping for live media streaming.[19] In this system, the video stream is divided into segments of uniform length, and the availability of the segments in a node's buffer is represented by a buffer map (BM). Each node continuously exchanges its BM with other peers (or partners). Upon the receipt of BMs from multiple partners, the node assigns the requested data segments to each of the partners according to their data availability and available uploading bandwidth. Research has also shown that the average overlay path length from the source to a node is $O(\log N)$.[19] This preserves the media data's freshness, which is crucial to a live media streaming application.

Hybrid Overlay Network (HON) combines the gossip protocol with an overlay tree for providing P2P VoD streaming.[6] Different from live streaming, partners in this case are the peers with their buffered content partially overlapped. Therefore, among the potential partners, each node continuously exchanges segment availability information with random targets. Apart from the gossip-dissemination protocol, the system also builds an overlay tree according to their play points such that the parent node's play point is before the children nodes' play points. Therefore, a parent node should obtain and play data segments before its children nodes do. When a segment isn't available at a node for playback within a time threshold (that is, the roundtrip time between the node and its parent plus the segment transmission time), the node contacts its tree parent to fetch the segment. The data delivery over the overlay tree complements the gossip protocol and works as the last resort to retrieve the missing video segments.

The gossip protocol lets each peer in the system retrieve data from multiple parents and, at the same time, serve multiple children. Compared to tree-based protocols, this approach greatly improves resource utilization and load balancing. In addition, the service's stability is also enhanced because of the redundancy of service providers.

Unfortunately, because of the partner selection process's randomized nature, the quality of overlay delivery paths such as bandwidth and delay can't be optimized or even guaranteed.

Table 2. Comparisons between various approaches for content delivery.

Approach	Allow Optimization	Resilient to Node Failure	Multiple Suppliers	Load Balancing	Achievable Transmission Rate	Implementation
Single tree	✓	Poor	✗	Medium	Medium	Easy
Multiple trees	✗	Good	✓	Good	High	Difficult
Pull-based gossip	✗	Good	✓	Good	High	Easy

Comparisons

Among the three aforementioned approaches, single tree multicast is the only one that allows easy optimization for latency or other metrics, especially in a centralized approach. It's hard to guarantee low latency in gossip-based systems or to optimize multiple trees at the same time. While an overlay tree is fragile, multiple trees and gossip protocols are resilient to node/link failure. For load balancing, a single tree clearly isn't as good as the other approaches. Both multiple trees and gossip-based approaches can achieve a high transmission rate, but a single overlay tree's bandwidth monotonically decreases along the tree from the root to leaves. The ease of implementation is the advantage of single tree and gossip-based approaches. However, coordination among multiple overlay trees is difficult.

Table 2 further compares the various content delivery path maintenance approaches.

Conclusion

In this article, we focused only on two research issues related to media streaming. Others include resilience to node/link failure, preventing free riding and providing incentive mechanism, provisioning efficient user interactivity for VoD service, and offering services to users in networks behind network address translators (NATs) and firewalls. These issues are still active research topics.

As high-bandwidth wireless access becomes available everywhere, there will be a great demand on streaming applications such as news on demand through mobile devices. The techniques used in P2P media streaming could be applied in the wireless environment. However, unlike the Internet, connections in wireless networks are even more dynamic and unstable. Efforts are needed to cope with the challenges. **MM**

Acknowledgments

This work was supported, in part, by the Competitive Earmarked Research Grant HKUST 6156/03E of the Research Grant Council in Hong Kong and the Innovation and Technology Commission of the Hong Kong Special Administrative Region, China, GHP/045/05.

References

1. Y. Chu, S. Rao, and H. Zhang, "A Case for End System Multicast," *Proc. ACM Sigmetrics*, ACM Press, 2000, pp. 1-12.
2. K. Sripanidkulchai et al., "The Feasibility of Supporting Large-Scale Live Streaming Applications with Dynamic Application End-Points," *Proc. ACM SIGCOMM*, ACM Press, 2004, pp. 107-120.
3. X. Liao et al., "AnySee: Peer-to-Peer Live Streaming," *Proc. IEEE Infocom*, IEEE Press, 2006.
4. M. Gao and M.H. Ammar, "Scalable Live Video Streaming to Cooperative Clients Using Time Shifting and Video Patching," *Proc. IEEE Infocom*, IEEE Press, 2004, pp. 1501-1511.
5. X. Hei et al., "Insights into PPLive: A Measurement Study of a Large-Scale P2P IPTV System," *Proc. Workshop on Internet Protocol TV (IPTV) Services over World Wide Web*, May 2006.
6. M. Zhou and J. Liu, "A Hybrid Overlay Network for Video-on-Demand," *Proc. IEEE Int'l Conf. Comm.* (ICC), IEEE Press, 2005, pp. 1309-1311.
7. Y. Guo et al., "A Peer-to-Peer On-Demand Streaming Service and Its Performance Evaluation," *Proc. IEEE Int'l Conf. Multimedia and Expo* (ICME), IEEE CS Press, vol. 1, 2003, pp. 649-652.
8. V.N. Padmanabhan, H.J. Wang, and P.A. Chou, "Supporting Heterogeneity and Congestion Control in Peer-to-Peer Multicast Streaming," *Proc. 3rd Int'l Workshop Peer-to-Peer Systems 2004* (IPTPS), LNCS 3279, Springer, 2005, pp. 54-63.
9. V.N. Padmanabhan et al., "Distributing Streaming Media Content Using Cooperative Networking," *Proc. ACM Network and Operating System Support for Digital Audio and Video* (NOSSDAV), ACM Press, 2002, pp. 177-186.
10. T.T. Do, K.A. Hua, and M.A. Tantaoui, "P2VoD: Providing Fault Tolerant Video-on-Demand Streaming in Peer-to-Peer Environment," *Proc. IEEE Int'l Conf. Comm.* (ICC), IEEE Press, 2004, pp.1467-1472.
11. D.A. Tran, K.A. Hua, and T.T. Do, "A Peer-to-Peer Architecture for Media Streaming," *IEEE J. Selected*

Areas in Comm. (JSAC), vol. 22, no. 1, 2004, pp. 121-133.
12. S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable Application Layer Multicast," *Proc. ACM Special Interest Group on Comm.* (Sigcomm), ACM Press, 2002, pp. 205-217.
13. A. Sharma, A. Bestavros, and I. Matta, "dPAM: A Distributed Prefetching Protocol for Scalable Asynchronous Multicast in P2P Systems," *Proc. IEEE Infocom*, IEEE Press, 2005, pp. 1139-1150.
14. M. Castro et al., "SplitStream: High-Bandwidth Multicast in Cooperative Environments," *Proc. ACM 19th Symp. Operating Systems Principles* (SOSP), ACM Press, 2003, pp. 298-313.
15. I. Stoica et al., "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications," *IEEE/ACM Trans. Networking*, vol. 11, no. 1, 2003, pp. 17-32.
16. A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems," *Proc. IFIP/ACM Int'l Conf. Distributed Systems Platforms (*Middleware*)*, 2001, pp. 329-350; http://research.microsoft.com/~antr/PAST/pastry.pdf.
17. *FIPS 180-1, Secure Hash Standard*, US Dept. of Commerce/NIST, Nat'l Tech. Information Service, 1995.
18. X. Jiang et al., "GnuStream: A P2P Media Streaming System Prototype," *Proc. IEEE Int'l Conf. Multimedia and Expo* (ICME), IEEE CS Press, vol. 1, 2003, pp. 325-328.
19. X. Zhang et al., "CoolStreaming/DONet: A Data-Driven Overlay Network for Live Media Streaming," *Proc. IEEE Infocom*, IEEE Press, 2005, pp. 2102-2111.
20. A.J. Ganesh, A.-M. Kermarrec, and L. Massoulie, "Peer-to-Peer Membership Management for Gossip-Based Protocols," *IEEE Trans. Computers*, vol. 52, no. 2, 2003, pp. 139-149.
21. Y. Cui, B. Li, and K. Nahrstedt, "oStream: Asynchronous Streaming Multicast in Application-Layer Overlay Networks," *IEEE J. Selected Areas in Comm.* (JSAC), vol. 22, no. 1, 2004, pp. 91-106.
22. K. Hua, Y. Cai, and S. Sheu, "Patching: A Multicast Technique for True Video-on-Demand Services," *Proc. ACM 6th Int'l Conf. Multimedia*, ACM Press, 1998, pp. 191-200.
23. V.K. Goyal, "Multiple Description Coding: Compression Meets the Network," *IEEE Signal Processing Magazine*, vol. 18, no. 5, 2001, pp. 74-93.
24. J. Chakareski, S. Han, and B. Girod, "Layered Coding vs. Multiple Descriptions for Video Streaming over Multiple Paths," *Proc. 11th ACM Int'l Conf. Multimedia*, ACM Press, 2003, pp. 422-431.
25. J. Li, *PeerStreaming: A Practical Receiver-Driven Peer-to-Peer Media Streaming System*, tech. report MSR-TR-2004-101, Microsoft Research, 2004.
26. C. Wu and B. Li, "rStream: Resilient Peer-to-Peer Streaming with Rateless Codes," *Proc. 13th ACM Int'l Conf. Multimedia*, ACM Press, 2005, pp. 307-310.
27. R. Ahlswede et al., "Network Information Flow," *IEEE Trans. Information Theory*, vol. 46, no. 4, 2000, pp. 1204-1216.

W.-P. Ken Yiu is a PhD candidate with the Department of Computer Science and Engineering at the Hong Kong University of Science and Technology. His research interests include computer networks, peer-to-peer systems, multimedia networking, and network security. Yiu received his MPhil in computer science from the Hong Kong University of Science and Technology. He was awarded the Academic Achievement Medal from the Hong Kong University of Science and Technology and Sir Edward Youde Memorial Fellowship from Sir Edward Youde Memorial Fund. He is a student member of the IEEE Computer Society.

Xing Jin is pursuing his PhD with the Department of Computer Science and Engineering at the Hong Kong University of Science and Technology. His research interests include overlay multicast, Internet topology inference, end-to-end measurements, and peer-to-peer streaming. He was awarded the Microsoft Research Fellowship. He is a junior editor of the *Journal of Multimedia*.

S.-H. Gary Chan is an associate professor with the Department of Computer Science and Engineering at the Hong Kong University of Science and Technology. His research interests include multimedia networking, peer-to-peer technologies and streaming, and wireless communication networks. Chan received his PhD in electrical engineering from Stanford University. He is a member of Tau Beta Pi, Sigma Xi, and Phi Beta Kappa.

Readers may contact the authors at the Dept. of Computer Science and Eng., Hong Kong Univ. of Science and Technology, Clear Water Bay, Kowloon, Hong Kong; {kenyiu, csvenus, gchan}@cse.ust.hk.