Two Decades of Internet Video Streaming: A Retrospective View

BAOCHUN LI, University of Toronto
ZHI WANG, Tsinghua University
JIANGCHUAN LIU, Simon Fraser University
WENWU ZHU, Tsinghua University

For over two decades, video streaming over the Internet has received a substantial amount of attention from both academia and industry. Starting from the design of transport protocols for streaming video, research interests have later shifted to the peer-to-peer paradigm of designing streaming protocols at the application layer. More recent research has focused on building more practical and scalable systems, using Dynamic Adaptive Streaming over HTTP. In this article, we provide a retrospective view of the research results over the past two decades, with a focus on peer-to-peer streaming protocols and the effects of cloud computing and social media.

Categories and Subject Descriptors: C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*

General Terms: Performance, Experimentation

Additional Key Words and Phrases: Video streaming, multicast, P2P streaming, HTTP streaming, cloud computing, social media, multimedia streaming

ACM Reference Format:

Li, B., Wang, Z., Liu, J., and Zhu, W. 2013. Two decades of Internet video streaming: A retrospective view. ACM Trans. Multimedia Comput. Commun. Appl. 9, 1s, Article 33 (October 2013), 20 pages.
DOI: http://dx.doi.org/10.1145/2505805

1. INTRODUCTION

Driven by an insatiate appetite for bandwidth in the Internet, advances in media compression technologies, and accelerating user demand, video streaming over the Internet has quickly risen to become a mainstream “killer” application over the past two decades. Since the very early stages of the Internet [Forgie 1979], it has always been the belief that videos should be streamed to the users over the network: a user can begin playing a video segment before the entire video has been transmitted. This

Authors’ addresses: B. Li (corresponding author), Department of Electrical and Computer Engineering, University of Toronto, 10 King’s College Road, Toronto, Ontario M5S 3G4, Canada; email: bli@eecg.toronto.edu; Z. Wang and W. Zhu, Department of Computer Science and Technology, Tsinghua University, Beijing 100084, P. R. China; email: wangzhi04@mails.tsinghua.edu.cn, wwzhu@tsinghua.edu.cn; J. Liu, School of Computing Science, Simon Fraser University, 8888 University Drive, Burnaby, BC V5A 1S6, Canada; email: jcliu@cs.sfu.ca.

DOI: http://dx.doi.org/10.1145/2505805

is more preferred and fundamentally different from having to wait for a download to complete before playback. As a result of this preference, video streaming has received a substantial amount of research attention in the past two decades, from both academia and industry.

In this article, we seek to go back in the time machine and present a retrospective view of the history of Internet video streaming by covering its main milestones in the past two decades of research and development. Following the chronological order, we present three stages of research development on Internet video streaming.

Client-server video streaming. During the 1990s and early 2000s, research attention mostly focused on the design and implementation of new streaming protocols, such as the design of the Real-time Transport Protocol (RTP) specifically for streaming media. The initial set of streaming protocols were incorporated by the media players, which were used to receive video streams from the streaming servers over the Internet.

Peer-to-peer video streaming. To scale up to an ever increasing number of users, peer-to-peer (P2P) video streaming has been studied extensively in the past decade and has widely been applied to both live and on-demand video streaming. The design of P2P streaming protocols was based on the philosophy that end hosts, called peers, were able to serve as both clients and servers, as opposed to the traditional client-server design where end hosts were only able to consume video. Real-world systems, such as PPLive, succeeded to serve thousands of video streams to millions of users, while consuming a modest amount of bandwidth at streaming servers.

HTTP video streaming over the cloud. To use P2P streaming protocols, users were required to download and install dedicated applications that implemented these protocols. However, it is much more convenient for users to stream videos directly over the Web using a standard Internet Web browser, without the need to download and install third-party applications. With HTTP video streaming, a video stream is divided into a sequence of small chunks that can be downloaded using HTTP. Due to the convenience of using the standard HTTP protocol, HTTP streaming has been widely used in the industry, with streaming servers hosted by cloud computing platforms. As a result, we are now witnessing a migration to cloud computing and social media as the predominant means to host and share video streams.

The remainder of this article is organized correspondingly as three main sections. In Section 2, we present important results from the early stages of research on Internet video streaming. In Section 3, we discuss the use of the peer-to-peer design philosophy to make video streaming scalable, and present a small number of representative works. In Section 4, we focus on the migration to HTTP streaming, as well as the effects of cloud computing and online social networks on video streaming. Finally, we conclude the article with a summary of the lessons learned in the past two decades of research in Section 5.

2. INTERNET VIDEO STREAMING USING THE CLIENT-SERVER ARCHITECTURE

The development of video compression technologies in the 1980s and the growth and popularity of the Internet in the 1990s have motivated the concept of streaming videos over the Internet to a large number of clients. For much of the 1990s, research in both academia and industry focused on the design and implementation of new protocols for Internet video streaming from dedicated streaming servers. Built on top of the Internet Protocol (IP), these new video streaming protocols were designed to support Quality of Service (QoS) efficiently over the best-effort Internet. In this section, we present a small number of key milestones to show how traditional client-server video streaming protocols have evolved in the first decade, before the advent of peer-to-peer (P2P) video streaming. While we are not able to cover all the important research results in this section, there exist excellent tutorial papers in the literature on video streaming protocols in the early stages of research [Cleary 1995; Aurrecoechea et al. 1998].

2.1 Transport Protocols for Video Streaming

In the early 1990s, resource reservation protocols for achieving Quality of Service (QoS) over the Internet, such as RSVP [Zhang et al. 1993], were superseded by new and simpler transport protocols that did not depend on any means to reserve bandwidth in the best-effort Internet. To detect packet loss and compensate jitter during transmissions over an IP network, the Real-time Transport Protocol (RTP) was proposed for end-to-end real-time transfer of stream data. RTP was designed with two fundamental principles: application-layer framing (i.e., framing for video data should be performed properly by the application layer) and integrated layer processing (i.e., integrating multiple layers into one to allow efficient cooperation) [Schulzrinne et al. 1996]. Based on the User Datagram Protocol (UDP), RTP defined a standardized packet format for delivering video over IP and was designed for end-to-end real-time transfer of stream data. The RTP Control Protocol (RTCP), also based on UDP, was designed to monitor transmission statistics and QoS, and to achieve synchronization across multiple streams. Important commercial and open-source products, such as QuickTime, have supported RTP. RTP has also been adopted by 3GPP, which has been incorporated in nearly all mobile devices.

As new transport protocols became critically important for Internet video streaming, the Internet Engineering Task Force (IETF) standardized the RTP/RTCP/RTSP protocol suite [Schulzrinne et al. 1996,1998], designed specifically for Internet video streaming. Beyond RTP and RTCP, the Real Time Streaming Protocol (RTSP) [Schulzrinne et al. 1998] in the protocol suite was designed to create and maintain video sessions and to provide VCR-style control functionality, enabling users to pause, resume, or seek in video streams, just like local playback. RTSP is an example of session control protocols, similar to the Session Initiation Protocol (SIP) that was later designed [Rosenberg et al. 2002]. Session control protocols have since played an important role in the management of active sessions, even in recent protocols in the context of 3D immersive environments [Nahrstedt et al. 2011], where research progress has been made on session management protocols in 3D tele-immersive systems [Baldino et al. 2011].

2.2 Rate Control and Rate Shaping

To support video streaming over the Internet, it is believed that transport protocols should be designed to meet real-time video delivery needs, by maximizing the video quality in the presence of packet loss. Bursty packet losses have a devastating effect on the video quality, and they are typically caused by network congestion. To support smooth video streaming sessions, rate control and rate shaping protocols [Floyd et al. 2000; Jacobs and Eleftheriadis 1998] were proposed to minimize the possibility of network congestion, by matching the rate of the video stream to the available network bandwidth.

Rate control was a technique designed to determine the sending rate of a video stream based on the estimated available bandwidth in the network. Traditional rate control schemes may be classified into three categories [Wu et al. 2001]: source-based, receiver-based, and hybrid rate control. The objective of *rate shaping* was to match the streaming rate to the target bandwidth [Jacobs and Eleftheriadis 1998], and it was required for source-based rate control, as a stored video is precompressed at a rate that may not match the available bandwidth in the network. There were many types of rate shapers, such as the codec filter, frame-dropping filter, layer-dropping filter, frequency filter, and re-quantization filter [Wu et al. 2001]. In the late 1990s and early 2000s, SureStream was proposed and became part of a product line, called G2, from Real Networks. SureStream created multiple encodings of each content, so that a client may choose one of the available qualities, and was even capable of switching between encodings during playback [Thomas 1998]. Zhang et al. [2001] studied the problem of rate adaptation according to the estimated network bandwidth, using each video's rate distortion function under various network conditions.

2.3 Error Control

Due to the best effort nature of the Internet, packet loss is inevitable, thereby having a significant impact on the perceptual quality of video streams. Error control [Wang and Zhu 1998] was proposed to ensure a smooth video streaming experience to the users, even with the presence of packet losses. A large number of error control mechanisms have been proposed in the first decade of research, which can be loosely organized into the following categories.

—*Forward Error Correction* (FEC). The idea of FEC was to add redundant information so that the original message could be reconstructed in the presence of packet loss [Albanese et al. 1996]. Bolot and Turletti [1996] proposed using error control mechanisms based on FEC in video streaming and evaluated its benefits. Based on the type of redundant information to be added, FEC schemes can take the forms of channel coding, source coding, and joint source/channel coding.

—*Error-Resilient Encoding*. The objective of error-resilient encoding was to improve the robustness of compressed video to packet losses. The standardized error-resilient encoding schemes included resynchronization marking, data partitioning, and data recovery [Wang et al. 2000]. However, these approaches were designed for error-prone environments, like wireless channels, and were not applicable in the context of the Internet. For video streaming over the Internet, the boundary of a packet has already provided a synchronization point in the variable-length coded bit-stream at the receiver side.

—*Error Concealment*. Error concealment was a technique in which an error in video delivery was replaced by synthetic content, often interpolated from other parts of the signal. It can be executed by the source (i.e., forward error concealment), by the receiver (i.e., error concealment by postprocessing), and by both the source and receiver (i.e., interactive error concealment) to improve the robustness of the compressed video before packet loss actually happens [Wang and Zhu 1998].

—*Delay-Constrained Retransmission*. Retransmission is usually dismissed as a method to recover lost packets in real-time video, since a retransmitted packet may miss its playback time. However, if the one-way delay is relatively short with respect to the maximum allowable delay, a retransmission-based approach (called delay-constrained retransmission) may still be a viable option for error control [Podolsky et al. 1999].

2.4 Proxy Caching

In the early days of video streaming, both the number of videos and the number of users were small. Videos could be served by a single server, which in many cases not only stored the videos but also uploaded them to the users. Such an architecture quickly became infeasible when more and more videos were made available online. An important technique for improving the scalability and reducing the latency was *proxy caching*, which was based on the observation that different clients will request many of the same videos.

While proxy caching was widely used in Web content distribution, video caching had a number of different focuses. On one hand, since the content of a video object was rarely updated, management issues, such as cache consistency and coherence, were less critical in video caching. On the other hand, given the high resource requirement of videos, effective management of proxy cache resources (i.e., disk space, disk I/O, and network I/O) became more challenging. Sen et al. [1999] demonstrated that by only caching the prefixes of videos could a large amount of videos be served by a few megabytes of buffer space at the proxy storage. To improve caching efficiency, Hua et al. analytically studied the storage requirement in the context of on-demand streaming [Hua and Sheu 1997; Hua et al. 1998]. Yu et al. [2003] proposed a QoS-adaptive proxy caching scheme for video streaming over the Internet, considering the heterogeneous network conditions and video characteristics. Other effective partial caching strategies, such as sliding-interval caching, segment caching, and rate-split caching, have also been proposed in the literature [Liu and Xu 2004].

2.5 IP Multicast for Video Streaming

The ever increasing user population also called for a revisit to the original one-to-one Internet communication paradigm. Maintaining unicast sessions for each user quickly became infeasible as the number of users scaled up. Many emerging applications, including Internet TV and live event broadcast, required the support for video multicast, that is, simultaneously delivering a video stream to a large number of clients.

IP multicast [Deering and Cheriton 1990] was proposed as an extension to IP unicast, with the objective of providing efficient multipoint packet delivery. Given that the network topology was best known in the network layer, multicast routing at this layer was also the most efficient. IP multicast retained the semantics of IP and allowed users to dynamically join or leave multicast groups.

In the early 1990s, end-to-end adaptation schemes were mainly sender-driven, where the sender adjusted its transmission rate according to some feedback from a single receiver. This did not work well for multicasting to heterogenous receivers without a common target rate. One solution was stream replication, which could be viewed as a trade-off between single-rate multicast and multiple point-to-point connections [Kim and Ammar 2001]. Its feasibility was well justified in a typical multicast environment where the bandwidth of receivers usually followed some clustered distribution. As a result, a limited number of streams could be used to match these clusters to achieve a reasonably good performance.

With advances in scalable or cumulative layered video coding, McCanne et al. [1996] proposed the first practical Receiver-driven Layered Multicast (RLM) protocol. RLM mapped different layers of a scalable video source to distinct multicast groups and let the receivers independently decide the number of layers to subscribe, so that they are commensurate with their individual capacities. Amir et al. [1995] proposed an architecture where video transmission can be decomposed into multiple sessions with different bandwidth requirements using an application-level gateway. Li and Nahrstedt [1999] proposed to design a middleware framework for adapting the quality of applications based on application-specific needs. Ooi et al. [2000] further studied the multicast heterogeneous network environment and designed a programmable gateway system for processing videos at strategic locations in the network to reduce the bandwidth requirements.

2.6 Application-Layer Multicast for Video Streaming

Today, the scope and reach of IP multicast remain limited, as many Internet Service Providers (ISPs) simply block or disable IP multicast due to various security and economic concerns [Diot et al. 2000]. The idea of using the application layer for multicast, called *application-layer multicast (ALM)*, was proposed around 2000 [Sheu et al. 1997; Chu et al. 2000]. Though both application-layer and IP multicast require intermediate nodes in the network topology to support the replication of data packets, it was much easier to implement multicast at the application layer on end hosts, as opposed to the network layer at switches and routers [Hosseini et al. 2007]. Cui et al. [2004] showed that with respect to bandwidth consumption on the backbone network, the benefit introduced by application-layer multicast overshadowed the topological inefficiency introduced.

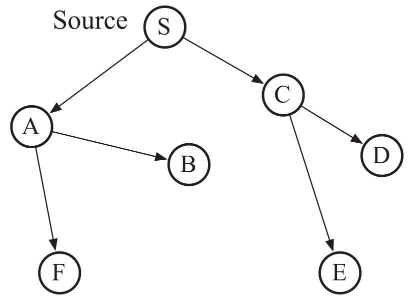

Fig. 1. Application-layer multicast with a single tree.

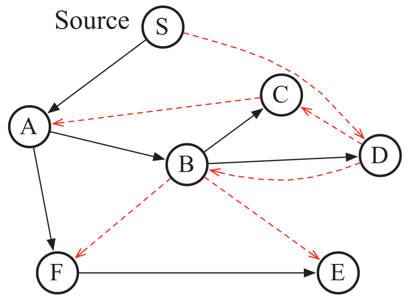

Fig. 2. Application-layer multicast with multiple trees.

Chu et al. [2000] showed that application-layer multicast performed reasonably well compared to IP multicast, incurring a comparable delay and a reasonable amount of bandwidth penalty. In the initial proposals, a single multicast tree was constructed for each multicast session. For example, Overcast [Jannotti et al. 2000] attempted to organize a bandwidth-efficient tree by letting peers join near the root and then migrating them down the tree to a position according to bandwidth availability. Tran et al. [2003] proposed ZIGZAG in which the video server distributes chunks to many clients by organizing them into an appropriate tree rooted at the server. The advantage of such a design, shown in Figure 1, was its simplicity.

The design of a single multicast tree suffered from a number of problems. First, such a design may not be fair to all the peers. As we can easily see from Figure 1, when the tree was formed, some peers were chosen to be interior nodes that must contribute their upload bandwidth to support others, while others were leaf nodes that were not required to contribute any upload bandwidth. Even if fairness was not a concern, the multicast rate that a child node can enjoy was restricted by the upload bandwidth available at its parent. In case the parent node left the multicast session, the children would be left in the cold, waiting for a new parent.

To improve fairness in application-layer multicast, it has been proposed since 2003 that streams can be split to multiple *slices* and distributed across a *forest* of multiple interior-node-disjoint multicast trees [Castro et al. 2003], or a mesh overlay on top of a tree [Kostić et al. 2003]. Figure 2 illustrates an example of multicasting with two trees, constructed with the intention that a majority of nodes that are interior nodes in one tree will be leaf nodes in the other tree. By distributing the responsibility of contributing uploading bandwidth to most of the peers in the multicast session, the fairness problem is mitigated, yet the robustness problem remains to be solved.

3. PEER-TO-PEER VIDEO STREAMING

In contrast to the client-server streaming model where the consumption and supply of resources (e.g., bandwidth) is always decoupled, peer-to-peer streaming evolves in which *peers* are both suppliers and consumers of resources. The peer-to-peer design philosophy takes advantage of the ability of participating end hosts, or peers, in a multicast group to contribute their uplink bandwidth. The peer-to-peer design has two major advantages. First, it does not require the support from the underlying network infrastructure, and as a result, it is cost-effective and easy to be deployed. Second, in the peer-to-peer design, a peer is not only downloading a video stream, but also uploading it to other peers watching the same program. As a result, it has the potential to scale up with the group size, as a stronger demand also generates more suppliers.

In retrospect, application-layer multicast and peer-to-peer share the same design philosophy. Application-layer multicast is largely *push-based*, in that video streams are being pushed to the

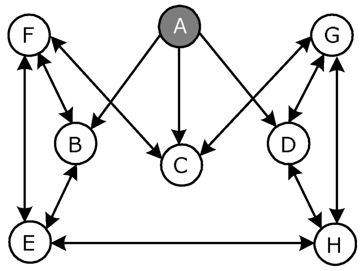

Fig. 3. An illustration of neighbors in the pull-based peer-to-peer streaming system with A being the source.

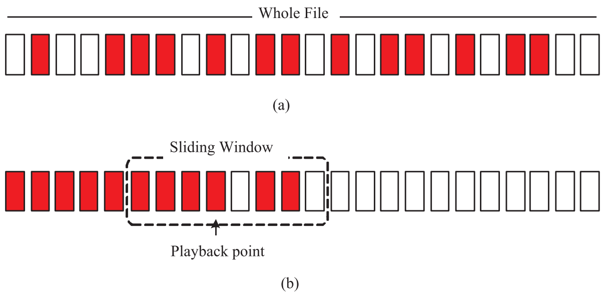

Fig. 4. Buffer snapshots of (a) BitTorrent and (b) CoolStreaming, where shaded segments are available in the buffer.

receivers along one or more trees. As the tree topologies have been constructed before the streaming session starts, each receiver in the trees only needs to forward the video stream to its downstream receivers, with very minimal delays due to such forwarding.

The design philosophy in BitTorrent has converged with academic solutions in application-layer multicast, and a new generation of *pull-based* peer-to-peer streaming protocols on random mesh topologies has emerged, independently proposed in Chainsaw [Pai et al. 2005] and CoolStreaming [Zhang et al. 2005b]. In such protocols, peers exchange information with their neighbors periodically about what video segments each of them has in their buffers, and a missing video segment must be explicitly requested and transferred from one of the neighbors who has it. In comparison, application-layer multicast based on multicast trees adopts a more rigid design [Venkataraman et al. 2006; Magharei et al. 2007], in that the structure of each tree needs to be actively managed as peers join and leave the session. As a result, pull-based protocols are much simpler to design and more amenable to real-world implementations [Zhang et al. 2007].

On the flip side of the coin, the main challenge of designing pull-based protocols is the timing constraints that new streaming protocols must observe: if video segments do not arrive in time, they are not useful when it comes to the time of playing them back. In this section, we first present an example of pull-based streaming protocols and then illustrate how network coding is able to help address the challenges of timing constraints in pull-based streaming.

3.1 Traditional Pull-Based P2P Video Streaming

CoolStreaming [Zhang et al. 2005b] was one of the first real-world implementations of pull-based peer-to-peer streaming protocols. In CoolStreaming, a peer maintains a partial view of other peers as its *neighbors* and schedules the transmission of video segments by sending outgoing requests to its neighbors. In CoolStreaming, a video stream is divided into segments of a uniform length, and the availability of active segments in the buffer of a peer is represented by a *buffer map*. Each peer continuously exchange its buffer map with its neighbors and then uses a *scheduling algorithm* to determine which segment is to be fetched from which neighbor accordingly. An example of its neighboring relationship is shown in Figure 3.

As shown in Figure 4, a dynamic *sliding window* of segments over time needs to be distributed, unlike a fixed number of data blocks in file sharing (such as BitTorrent). As the sliding window moves forward in the stream over time, blocks are to be received in approximately the same sequence as they are played back, and out-of-order delivery can only occur within the confines of the sliding window. Each of the blocks are distributed using a gossip protocol in a random mesh topology of neighbors (an example of which has been shown in Figure 3), requiring peers to pull blocks from one another.

Since all participating peers are roughly synchronized with respect to their points of playback, they are able to periodically exchange the states of their respective buffers in the sliding window. Based on the knowledge of block availability in each other's sliding windows, a peer sends requests to its neighbors in order to pull blocks it has yet to receive.

One of the main advantages of the pull-based design is its resilience to failures. A peer can depart either gracefully or accidentally due to an unexpected failure. In either case, the departure can be easily detected after a period of idle time, and an affected peer can quickly react by requesting the missing segments from its other neighbors.

Fundamentally, setting up and maintaining trees in application-layer multicast is similar to setting up a connection in a telephone network: states of parent-child relationships are established so that data blocks can be transmitted without explicit requests taking place. In contrast, the distribution of data blocks in a pull-based protocol is similar to *gossiping* in a social setting. Following such a gossiping philosophy to distribute each of the data blocks to all the peers in the multicast session, early peer-to-peer video streaming systems, such as CoolStreaming, have motivated a large number of production-quality real-world implementations in the industry, several of which has become core technologies in start-up companies that specialized in live media streaming, including PPLive [Huang et al. 2008] and PPStream [Silverston and Fourmaux 2007].

Even though the gossiping philosophy incurs some delays in live streaming systems due to explicit requests and periodic information exchanges, its most salient advantage is its simplicity in design and implementation. If the current sliding window of the media stream to be distributed is divided into small media blocks, they will flow through the entire network wherever there exists idle upload bandwidth that can be tapped into.

3.2 P2P Video Streaming with Network Coding

Traditional pull-based P2P streaming is based on the periodic exchange of buffer availability maps, and its communication overhead cannot be overlooked. Intuitively, since the sliding window at a peer advances itself over time, buffer availability maps need to be exchanged as frequently as needed, which may lead to a substantial amount of overhead. To mitigate such an overhead, most practical live streaming systems choose to exchange buffer states less frequently. An analytical study [Feng et al. 2009], however, has attributed the performance gap between practical systems and their theoretically optimal performance to the lack of timely exchanges of buffer states.

Since 2005, the use of *network coding* has emerged as a potential remedy to these challenges in peer-to-peer video streaming systems [Liu et al. 2010; Park et al. 2006]. Network coding, first proposed in the information theory community in 2000 [Ahlswede et al. 2000], recognized the ability to code at intermediate network nodes in a communication session, in addition to the ability to forward and to replicate incoming packets. In contrast, traditional multicast only recognized the ability to forward and to replicate packets. In 2003, Ho et al. [2003] further proposed the concept of *random network coding*, where a network node transmits on each of its outgoing links a linear combination of incoming packets over a finite field, with randomly chosen coding coefficients.

It is natural to conceive a simple but new design of peer-to-peer video streaming systems in which peers, as end hosts, are able to forward, replicate, and code incoming video data blocks. But will the use of network coding improve the performance, as compared to traditional pull-based P2P streaming? In particular, would the use of random network coding be able to mitigate the problem of the communication overhead? Wang and Li [2007] have raised such a question and proposed a new protocol called R^2 that used random network coding to substantially improve the performance of live streaming systems.

R^2 divides the content of the media stream in a sliding window into *generations*, each of which is further divided into m video blocks. With the introduction of generations in R^2, we can afford to

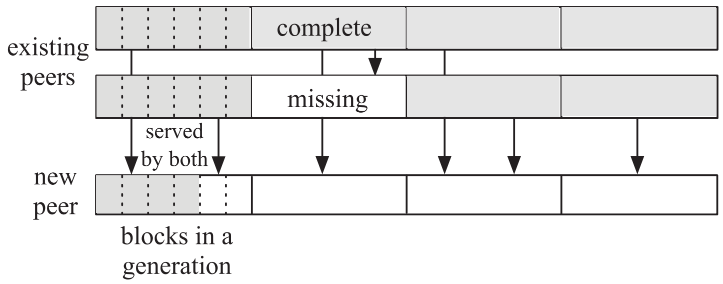

Fig. 5. Peer-to-peer live streaming with the use of random network coding: multiple existing peers are able to collaborate and serve coded blocks within the same generation to a new peer joining the session, minimizing its initial buffer delay.

design parameter settings so that a *video block* is much smaller than its counterpart in traditional live streaming systems based on random gossiping. This is due to the fact that buffer states only need to be exchanged at the granularity of a generation (one bit to represent each generation), rather than a block. With the same amount of communication overhead to exchange buffer states, the size of a single block can be much smaller in R^2. When a peer serves a generation s to its downstream target peer p, it linearly encodes all the video blocks it has received so far from s using random coefficients in GF(2^8), and then transmits the coded block to p. Since each peer buffers coded blocks it has received so far, it is able to linearly combine those from the same generation using random coefficients.

Since live streaming systems have timing requirements during playback, R^2 uses *random push* instead of pull to transmit data: each peer randomly selects a small number of downstream peers. When serving a chosen downstream peer, it then randomly selects a generation to code within, among those that the downstream peer has not yet completely received. If generations closer to the point of playback have not been completely received, they are given a higher priority, as they are more urgent.

There are three clear advantages brought forth by the use of random network coding in R^2. First, it greatly simplifies protocol design. Since coded blocks within a generation are equally useful, a peer only needs to blindly push coded blocks in the same generation till the downstream peer has obtained a sufficient number of them to decode. This eliminates the need of sending explicit requests to pull missing blocks, and saves the communication overhead associated with these requests. Second, R^2 induces much less overhead involved in buffer state exchanges due to a smaller number of generations in the sliding window. Finally, R^2 allows for "perfect collaboration" among multiple upstream peers when serving a downstream peer. Shown in Figure 5, as a new peer joins the session, multiple existing peers are able to collaborate and push fresh coded blocks in the first one or two generations after the playback point, so that the rate of accumulating blocks in these generations is only limited by the new peer's available downlink bandwidth.

4. HTTP VIDEO STREAMING OVER THE CLOUD AND SOCIAL MEDIA

Though P2P has proven to be highly efficient in video delivery, it is not convenient to regular users. In order to cache video contents being played back and to serve other peers with P2P streaming, users are usually required to install standalone applications and to keep some TCP or UDP ports open through NAT and firewalls. This is not as convenient as the turn-key solution of using a standard Web browser to watch video streams.

After WebRTC was developed [Bergkvist et al. 2012], these issues seemed to have been resolved; however, due to its limited deployment by browsers, a solution to stream videos over HTTP has seen a substantial amount of industry support. In this section, we present HTTP streaming, which was proposed even before P2P video streaming was largely deployed [Carmel et al. 2002], but has been used more widely only recently.

4.1 Dynamic Adaptive Streaming over HTTP

4.1.1 *HTTP Streaming from Content Distribution Networks.* The rapid growth of HTTP streaming is partly due to the extensive support from content distribution networks (CDN) [Peng 2004]. CDNs deploy servers in multiple geographically diverse locations, distributed over multiple ISPs [Vakali and Pallis 2003]. CDNs allow users to stream videos from a server close to them—user requests are redirected to the best available server based on either geographical proximity or server load. Since today's CDNs are mainly designed and optimized to serve web contents [Pallis and Vakali 2006], HTTP video streaming can be regarded as downloading video segments progressively from Web servers via the HTTP protocol so that clients that support HTTP can seek to arbitrary positions in the media stream by performing byte range requests to the web server [Fielding et al. 1999].

As a result, CDNs can be effectively used for high-quality TV content [Cahill and Sreenan 2004]. Adhikari et al. [2012] discovered that Netflix, the leading on-demand Internet video streaming provider, accounts for 29.7% of the peak downstream traffic in U.S. and it employs a mix of datacenters and CDNs for video content distribution. Watson [2011] has studied the *Dynamic Adaptive Streaming over HTTP* (DASH) framework used by Netflix, which is the largest DASH-based streaming provider in the world.

HTTP video streaming overcomes the challenges of convenience in P2P streaming systems, as it requires only a standard Web browser to view video streams and does not need to keep nonstandard TCP or UDP ports open though firewalls and NAT traversals. It works by breaking the overall video stream into a sequence of small HTTP-based file downloads. Users progressively download these small files, while a specific file is being played. Today, HTTP video streaming has been widely adopted by the industry, as major video streaming solutions, such as Netflix, YouTube, and Hulu, all resort to HTTP to stream their videos to the users.

4.1.2 *Research Problems with DASH.* Due to the best-effort nature of streaming videos over the Internet, Dynamic Adaptive Streaming over HTTP (DASH) has been proposed to adapt the streaming rates from Web servers. DASH was developed in 2010 [MPEG 2010] and has become a new standard in 2011 [Stockhammer 2011] to enable high-quality streaming of media content over the Internet, delivered from conventional HTTP Web servers. It works by breaking encoded content with multiple rates into small segments such that a client may continuously adjust its requests according to the estimates of local bandwidth availability. DASH can be represented by the following features.

—*Segmentation*. In DASH, a video component is encoded and divided in multiple segments, with the initialization segments containing the required information for initializing the media decoder, as well as the media segments containing the video data.

—*Media Presentation Description*. The media presentation description (MPD) describes how the segments form a video presentation [MPEG 2010]. Using the MPD, the clients request the segments for smooth playback and adjust bitrates or other attributes according to bandwidth estimates.

—*Codec Independence*. DASH is codec agnostic, and its prime container is the MP4 and MPEG-TS. It also allows seamless adoption of the upcoming improved HEVC video codec (i.e., H265).

Despite its advantages and features, the increasing popularity of deploying DASH has also led to a number of research problems.

—*Rate Adaptation Components*. DASH only defines the segmentation and the file description, and leaves rate adaptation for either the client or the server to implement. There exists a number of solutions along these lines in the recent literature. Clients can utilize multipath and multiserver approaches to receive video segments [Gouache et al. 2011]. In such receiver-driven approaches, the protocols are usually implemented at the application layer [Havey et al. 2012]. On the other hand, servers are also able to adaptively change the bitrate for its clients, based on the perception of the client download speed and server load. De Cicco et al. [2011] used a feedback mechanism to allow clients to communicate with a server, which performs the rate adaptation.

—*Rate Adaptation Strategies*. Rate adaptation strategies determine how different versions of segments are received by users to achieve the objectives, including streaming stability, fairness, and high quality. Akhshabi et al. [2011] have compared the bitrate adaptation of some popular DASH client implementations and observed that these implementations were either too aggressive or too conservative, causing unfairness and inefficiency. Jiang et al. [2012] have proposed to jointly take efficiency, fairness, and stability into consideration when adjusting the video bitrate.

—*User Quality Experience*. Though rate adaptation is highly correlated with the users' video quality experience in DASH streaming, such a correlation may require a more detailed study. Cranley et al. [2006] demonstrated the dynamic nature of user perception with adapting video streaming. In the context of DASH, Mok et al. [2012] have studied the user experience and observe that users prefer a gradual quality change between the best and worst quality levels, instead of abrupt switching. To better guide the design of rate adaptation strategies, a good metric for evaluating the user experience is still an open area of research [Song et al. 2011].

4.2 Video Streaming from the Cloud

As HTTP streaming is increasingly employed by major content providers, more and more video streaming systems have been built and deployed with HTTP streaming. As a result, the sharing of user generated content (UGC) (e.g., YouTube) has fundamentally changed the video streaming landscape, where regular users dynamically generate the video contents to be uploaded to the servers.

The highly centralized design of datacenters had catapulted the popularity of *cloud computing* since 2008, and had embraced a design philosophy that is exactly the opposite to that used by peer-to-peer systems. Attracted by the abundant networking resources in the cloud and the on-demand "pay-as-you-go" pricing model, an increasing number of video streaming services have been hosted by the cloud computing platforms. For example, Netflix, one of the leading video streaming service providers, has been reported to resort to the cloud service [Cockcroft 2011]. As a result, the design of network topologies in datacenters has quickly become an active area of research, drawing significant research attention since 2008. Zhu et al. [2011] introduced the main concepts of multimedia cloud computing and presented a cloud-based video streaming framework. Such a paradigm shift to cloud computing, in the industry and academia alike, has superseded the research interests in peer-to-peer streaming.

Benefits of Cloud-Based Streaming. A cloud computing platform offers reliable, elastic, and cost-effective resource provisioning, and has changed the way of enabling scalable and dynamic network services. There have been pioneering studies on demand-driven resource provisioning, as well as initial attempts that leverage cloud services to support media streaming applications, from both industry (e.g., Netflix) and academia [Wu et al. 2011; Huang et al. 2011]. Cloud providers have made it feasible for video-on-demand (VoD) providers, such as Netflix, to pay by GB for bandwidth resources, leading to long-term cost savings. Over the short term, a cloud-based video streaming service is able to handle bursty demands very well.

Limitations of Cloud-Based Streaming. There are, however, a number of critical theoretical and practical challenges to be addressed when migrating video streaming services to the cloud. First,

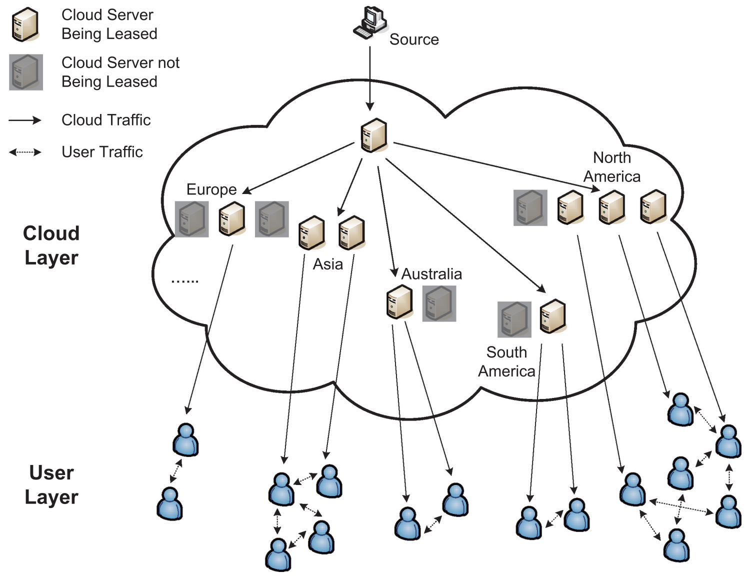

Fig. 6. A framework for cloud-based video streaming.

servers in the cloud have diverse capacities and lease prices; and the billing cycle for the lease cannot be arbitrarily short either—Amazon EC2 leases its virtual machines by the hour. As such, when being leased, the virtual machines and their corresponding charges cannot be simply terminated at any time. For a newly leased virtual machine, it takes a substantial amount of time (a few minutes) to instantiate it.[1] Therefore, a VoD provider needs to accurately predict its future demand so that an adequate number of new virtual machines can be leased to meet such demands with minimized costs [Niu et al. 2012].

Figure 6 shows a generic framework that facilitates the migration of existing live media streaming services to a cloud-based solution [Wang et al. 2012a]. The framework, consisting of a *cloud layer* and *user layer*, adaptively leases and adjusts cloud servers in a fine granularity to accommodate temporal and spatial dynamics of user demands. Upon receiving a user's subscription request, the cloud layer redirects this user to a selected cloud server, with the re-direction being transparent to the user. Given time-varying user demands, more server resources will be leased upon an increase in demand during peak times, or otherwise terminated. The cloud layer can serve a storage- and bandwidth-buffer for the user layer and can mitigate the impact of demand dynamics.

These challenges are further complicated given the global heterogeneous distributions of the cloud servers and that of the user demands. Today's live streaming applications have become highly globalized, with subscribers from all over the world. Such globalization makes user behavior and demands even more diverse and dynamic [Wang et al. 2012a]. The impact of such globalized demand patterns cannot be addressed for a single-datacenter-based cloud, and a geographically distributed cloud service is naturally a better solution.

[1]http://aws.amazon.com/ec2/.

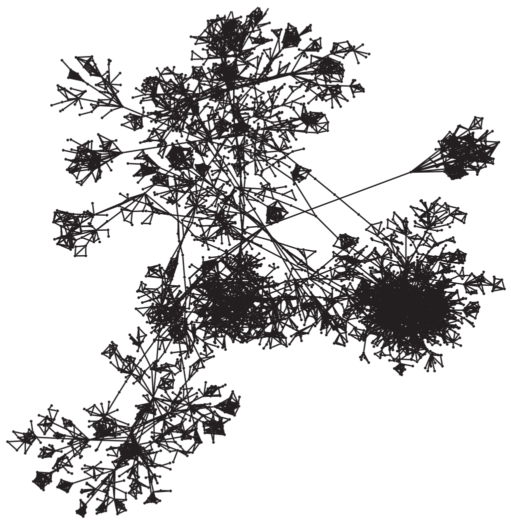

Fig. 7. A sample graph of YouTube videos and their links.

4.3 The Effects of User-Generated and Social Media

Compared to the Internet a decade ago, networked services in the Web 2.0 era focus more on the user experience, user participation, and interaction with rich media. The users are now actively engaged to be part of a social ecosystem, rather than passively receiving video streams.

When it comes to streaming services, one prominent example reflecting this change is YouTube. Established in 2005, it is now serving well over 4 billion views a day, with most of the contents being generated by users. The sheer number of UGC objects is orders of magnitude higher than that of traditional movies or TV programs and evolves rapidly. As of March 2013 [Statistics 2013], every second, 1.2 hours worth of video is uploaded on YouTube by users around the world, attracting almost 140 views for every person in the world on average. Earlier industry insiders estimated that YouTube spent over $1 million a day to pay for its server bandwidth. This has defeated any effort towards increasing server capacity and improving the user experience. Saxena et al. have revealed that the average service delay of YouTube is nearly 6.5 seconds, which is much longer than the other measured sites [2008].

While peer-to-peer mechanisms would be a candidate for scaling the video streaming system, the huge number of videos with a highly skewed popularity implies that many of the peer-to-peer overlays will be too small to function well. Moreover, user-generated videos are generally short (70% are shorter than 1 minute, even though YouTube has relaxed the length limit [Cheng et al. 2012]), implying that an overlay will suffer from an extremely high churn rate. These factors together make existing per-video based overlay design suboptimal, if not entirely inapplicable.

On the other hand, the UGC nature introduces new social relations and interactions among videos and users. In particular, there are interesting relations among the YouTube videos: the links to related videos generated by uploader's choices form a small-world network, as illustrated in Figure 7 [Cheng et al. 2012]. This suggests that the videos have strong correlations with each other. A user often quickly loads another related video when finishing the previous one. An earlier work that explores such relationships is NetTube [Cheng and Liu 2009], which introduces an upper-layer overlay on top of the swarms of individual videos. In the upper-layer overlay, given a peer, neighborhood relations are established among all the swarms that contain this peer. This conceptual relation facilitates a peer to quickly locate the potential suppliers for the next video and enable a smooth transition. The relations also enable effective pre-fetching of videos. While the repository of short videos is large, the next video in YouTube is most likely confined by the related list of the current video. This list in general includes 20 videos at most. Therefore, even if the number of pre-fetched videos is small, the hit ratio of the next video could still be quite high. After multiple rounds, the rate can easily reach over 90% given the small world of the videos [Cheng and Liu 2009].

Migrating YouTube-like UGC services to the cloud is nontrivial, either. A unique and critical step here is to partition the contents and assign them to a number of cloud servers. Not only do the loads of cloud servers have to be balanced, as in traditional standalone videos, but also the relationships among the videos and users have to be preserved so as to promote access locality. There are some existing works trying to solve this problem [Pujol et al. 2010; Cheng and Liu 2011], yet an all-around optimal solution remains to be developed.

The latest development of general social network applications, in particular, Facebook and Twitter, have further changed the information distribution landscape and even people's daily life. These online Social Networking Services (SNS) directly connect people through cascaded relations, and information thus spreads much faster and more extensively than through conventional Web portals or newsgroup services.

Such word-of-mouth spreading [Rodrigues et al. 2011] has also drastically expanded the ways of discovering the videos beyond traditional Web browsing and searching. There have been pioneering studies on information propagation over generic networks and, more recently, over social networks [Budak et al. 2011]. Yet their focuses have been largely on the conventional text or image objects and on their stationary coverage among users. The sheer and ever increasing data volume, the broad coverage, and the long access durations of video objects have presented more significant challenges than other types of objects to the SNS management, and to that of video sharing sites. A measurement study [Broxton et al. 2010] based on YouTube data has shown that, between April 2009 and March 2010, 25% of views on YouTube come from social sharing. Despite recent work towards this direction [Wang et al. 2012b], the characteristics of such requests from OSNs have yet to be comprehensively measured at large scales.

5. LESSONS LEARNED AND CONCLUDING REMARKS

In retrospect, the rise of research interests in video streaming systems was largely fueled by user demand and the increasingly abundant availability of bandwidth in the Internet, reflected both in the uplink bandwidth at end users (motivating the peer-to-peer philosophy), and in the downlink bandwidth at the servers in the cloud datacenters and in Content Distribution Networks (CDNs).

The popularity of the peer-to-peer design paradigm originated from its advantage of a *clean slate* starting point, in that it was not confined by any legacy protocols in the Internet. Due to the freedom of designing overlay topologies, the peer-to-peer paradigm was amenable to both theoretical and practical treatments, from modeling, analyses, to implementation and deployment. There have been technical challenges that have received heated discussions from researchers in this context. One example is on the tree-based structured versus pull-based structureless overlays. Both have shown their success in practical deployment, yet neither completely overcomes the challenges from a dynamic peer-to-peer environment. The selling point for pull-based systems is their simplicity, but they suffer from a latency-overhead trade-off with the exchange of buffer maps [Venkataraman et al. 2006; Zhang et al. 2005a]. A tree-based system does not suffer from this trade-off but has to face the inherent instability, maintenance overhead, and bandwidth underutilization. There have been attempts toward a hybrid design [Venkataraman et al. 2006; Wang et al. 2007]. It has been shown that most of the data blocks delivered through a pull-based mesh overlay essentially follow a specific tree structure or a small set of trees, mostly of stable peers [Wang et al. 2007]. As such, while maintaining a priori topology for all the peers is costly, optimizing the organization for a relatively stable core subset—with others being organized through a mesh—could simultaneously achieve high efficiency with low overhead and delay.

The hybrid design is seemingly a compromise; yet we believe that the root cause is the intrinsic heterogeneity of the Internet and its end users, which inevitably renders any one-fits-all solution to be suboptimal. For peer-to-peer streaming, its high scalability and low cost for server cluster deployment have made it feasible and promising. On the other hand, it faces significant challenges from a wide spectrum of issues, including incentives, fairness, availability, and stability. Consider an extreme *flash crowd* scenario of a popular concert broadcast that attracts one 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 enough for the initial period, a user is more likely to leave the system. This not only represents a failure of the system to provide service to this particular user but also generates a peer departure event, thus introducing more churn in the system [Sripanidkulchai et al. 2004]. This problem can hardly be addressed by the peer-to-peer design paradigm alone, and external assistance, such as Content Distribution Networks or cloud datacenters, is a must.

Whether the peer-to-peer design philosophy will thrive or survive hinges upon the crucial judgment of whether its advantages outweigh its drawbacks, being so vulnerable to flash crowds and high turnover rates. The potential copyright infringement as well as the lack of a clear business model for both the content providers and the users are also critical concerns. Measurement studies have shown that in some peer-to-peer video streaming systems, a small set of peers are requested to contribute 10 to 35 times more uploading bandwidth than downloading bandwidth, while many others are *free-riders* that contribute nearly zero. Such incentive mechanisms as *tit-for-tat* for peer-to-peer file download do not seem to work well for real-time streaming data. With recent downward trends in storage and bandwidth resource pricing, the momentum of the pendulum is swinging to the opposite side for peer-to-peer. According to Stoica [2010], bandwidth pricing of Content Distribution Networks was observed to be dropping quickly every year, from 40 cents per GB in 2006 to less than 5 cents per GB in 2010. As a result, the streaming cost per hour had decreased 15%–35%, to less than 3 cents per hour in 2010. The consequence of these observations was dramatically reduced distribution costs for content providers. For example, for paid content that is usually priced at $0.99 per episode, the distribution cost is less than 3% of the provider's total cost; for subscription-based premium content, it only costs $1.60 per month to stream content to a user watching two hours per day! On the other hand, the emergence of such ad-based business models have boosted the importance of video quality. There is a crucial interplay between video quality and user engagement [Dobrian et al. 2011], and what contributes to the majority of a content provider's revenue is the income from ad-supported premium content. The cost per thousand of ad impressions (CPM) for premium content has reached $20–$40, with a single ad covering the cost of an entire hour of streaming. As a consequence, content providers, with an objective of generating more revenue, start caring more about the visual quality and stability of their streaming service to maximize user engagement. Half of the Internet video providers are offering or planning to offer targeted or interactive advertising, and they continue to adopt social networking, growing to 41% by last year [Infonetics 2011b].

By leasing storage and bandwidth resources to leading VoD providers, such as Netflix, cloud computing has emerged as the potential winner by purchasing resources at wholesale and selling them to cloud users at retail. HTTP/DASH streaming powered by cloud computing and datacenters, coupled with the social media and social effects due to the highly-available online social network services, have recently enjoyed a meteoric rise in popularity, attracting an enthusiastic level of research attention just like peer-to-peer systems did a decade ago.

Looking back over the past two decades, Internet video has greatly changed the landscape of content distribution. It has largely swept out optical discs as the storage and distribution media in the movie industry. It is currently reshaping the TV broadcast industry with an ever-accelerating speed. Over the next five years, SNL Kagan [SNL 2011] has predicted that the global IPTV subscriber base will grow at a 9.1% compounded annual growth rate, reaching 100.5 million homes by 2017, equivalent to an 11.5% pay TV subscriber share. IPTV service revenues are expected to reach $41.2 billion by 2017, accounting for 13.9% of global total pay TV revenues. Today, most IPTV service providers use dedicated transport networks, providing quality access of digital TV programs from a head-end device to end users' dedicated set-top boxes (STBs). The Hybrid Broadcast Broadband TV (HbbTV) consortium of industrial pioneers, including SES, Humax, Philips, and etc, is currently promoting and establishing an open standard for hybrid set-top boxes that receive broadcast and broadband digital television and multimedia applications with a single-user interface. Our belief is that network convergence is inevitable and the global Internet will be the vehicle for the further expansion of IPTV services. As a matter of fact, in the U.K. BBC's iPlayer has been successfully broadcasting high-quality TV programs to both TV subscribers with STBs and public Internet users with Adobe Flashplayer since 2007; in the U.S. CNBC, Bloomberg Television and Showtime use live-streaming services from BitGravity's CDN to stream live television to paid subscribers using a standard http protocol. China, the largest IPTV market by subscribers (12.6 million) to date, is probably the most vigorous market seeing an entire range of technologies from peer-to-peer, to CDN, and to cloud computing, and these technologies are competing with each other and with dedicated IPTV networks.

Our argument also follows the changing viewing habits. In 2006, an Accenture's survey on IPTV [Accenture 2006] showed that compelling TV content is the core foundation of any IPTV proposition, which remains true today; yet it also indicated that consumers overwhelmingly preferred to watch television on a television set, and the least favored devices on which to watch television are mobile phones. The latest survey by Infonetics [2011a], however, has shown that IPTV services are getting highly personalized, integrated, portable, and on-demand. The results have confirmed an important observation that is fairly obvious now: most service providers are moving beyond basic video offerings toward richer user experiences. To date, 63% of IPTV service providers plan to support multi-screen viewing across TVs, PCs, tablets, and smartphones [Infonetics 2011b]. Meanwhile, multiview and multistreaming are being developed, in which multiple video streams from the same event are delivered to a user, who will be able to switch between camera views [Arefin et al. 2012]. This is a real recognition by service providers of what is happening in homes across the planet—that families are voraciously and simultaneously consuming streamed high-definition video on devices other than the traditional set-top box/TV couple.

The young adults and teenagers, otherwise known as the *Internet generation*, have long been hooked to their desktops or laptops, interacting, socializing, and generating their own video content. The new generation of smart mobile devices, mostly notably based on iOS, Android, and Windows Phone platforms, all of which emerged only in the recent five years, are driving the revolution further. The role of the Internet itself has evolved from the original use as a communication tool to provide easier and faster access to an infinite supply of information. Despite its rapidly expanding bandwidth, its narrow waist of the TCP/IP protocol suite, which used to be the key to its success, has fundamental design mismatches when providing real-time multimedia streaming services with a massive volume of data and a massive audience. Research efforts in the past two decades have largely focused on accommodating and remedying these mismatches. While a discussion of the future Internet architecture is beyond the scope of this article, we have seen exciting future development towards data- or information-centric networks [Ghodsi et al. 2011] that would turn to reality in the next two decades, with the hope of making truly scalable, robust, personalized, and social video streaming to networked users a reality.

REFERENCES

ACCENTURE. 2006. International IPTV Consumer Readiness Study. http://www.accenture.com/SiteCollectionDocuments/PDF/1PTV-ConsumerStudy.pdf.

ADHIKARI, V. K., GUO, Y., HAO, F., VARVELLO, M., HILT, V., STEINER, M., AND ZHANG, Z.-L. 2012. Unreeling Netflix: Understanding and improving multi-CDN movie delivery. In *Proceedings of the IEEE INFOCOM*.

AHLSWEDE, R., CAI, N., LI, S. R., AND YEUNG, R. W. 2000. Network information flow. *IEEE Trans. Inform. Theory 46,* 4, 1204–1216.

AKHSHABI, S., BEGEN, A. C., AND DOVROLIS, C. 2011. An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP. In *Proceedings of the ACM MMSys*.

ALBANESE, A., BLOMER, J., EDMONDS, J., LUBY, M., AND SUDAN, M. 1996. Priority encoding transmission. *IEEE Trans. Inform. Theory 42,* 6, 1737–1744.

AMIR, E., MCCANNE, S., AND ZHANG, H. 1995. An application level video gateway. In *Proceedings of the ACM Multimedia*.

AREFIN, A., HUANG, Z., NAHRSTEDT, K., AND AGARWAL, P. 2012. 4D TeleCast: Towards large scale multi-site and multi-view dissemination of 3DTI contents. In *Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS)*.

AURRECOECHEA, C., CAMPBELL, A. T., AND HAUW, L. 1998. A survey of QoS architectures. *Multimedia Syst. 6,* 3, 138–151.

BALDINO, B., DUCKWORTH, M., ROMANOW, A., AND PEPPERELL, A. 2011. Framework for telepresence multi-streams. http://tools.ietf.org/html/draft-ietf-clue-framework-10.

BERGKVIST, A., BURNETT, D. C., JENNINGS, C., AND NARAYANAN, A. 2012. Webrtc 1.0: Real-time communication between browsers. *Working draft, W3C*.

BOLOT, J.-C. AND TURLETTI, T. 1996. Adaptive error control for packet video in the internet. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*.

BROXTON, T., INTERIAN, Y., VAVER, J., AND WATTENHOFER, M. 2010. Catching a Viral Video. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*.

BUDAK, C., AGRAWAL, D., AND ABBADI, A. E. 2011. Limiting the spread of misinformation in social networks. In *Proceedings of the ACM WWW*.

CAHILL, A. J. AND SREENAN, C. J. 2004. An efficient CDN placement algorithm for the delivery of high-quality TV content. In *Proceedings of the ACM Multimedia*.

CARMEL, S., DABOOSH, T., REIFMAN, E., SHANI, N., ELIRAZ, Z., GINSBERG, D., AND AYAL, E. 2002. Network media streaming. U.S. Patent 6,389,473, filed March 24, 1998; issued March 14, 2002.

CASTRO, M., DRUSCHEL, P., KERMARREC, A.-M., NANDI, A., ROWSTRON, A., AND SINGH, A. 2003. SplitStream: High-bandwidth multicast in cooperative environments. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*.

CHENG, X. AND LIU, J. 2009. NetTube: Exploring social networks for peer-to-peer short video sharing. In *Proceedings of the IEEE INFOCOM*.

CHENG, X. AND LIU, J. 2011. Load-balanced migration of social media to content clouds. In *Proceedings of the ACM NOSSDAV*.

CHENG, X., LIU, J., AND DALE, C. 2012. Understanding the characteristics of internet short video sharing: A YouTube-based measurement study. *IEEE Trans. Multimedia 15*, 5.

CHU, Y.-H., RAO, S. G., AND ZHANG, H. 2000. A case for end system multicast. In *Proceedings of the ACM SIGMETRICS*.

CLEARY, K. 1995. Video on demand—competing technologies and services. In *Proceedings of the Broadcasting Convention*.

COCKCROFT, A. 2011. Netflix in the Cloud. http://www.Slideshere.net/adianco/netflix-in-the-cloud-2011.

CRANLEY, N., PERRY, P., AND MURPHY, L. 2006. User perception of adapting video quality. *Int. J. Human-Comput. Stud. 64,* 8, 637–647.

CUI, Y., LI, B., AND NAHRSTEDT, K. 2004. oStream: Asynchronous streaming multicast in application-layer overlay networks. *IEEE J. Select. Areas Commun. 22,* 1, 91–106.

DE CICCO, L., MASCOLO, S., AND PALMISANO, V. 2011. Feedback control for adaptive live video streaming. In *Proceedings of the ACM MMSys*.

DEERING, S. AND CHERITON, D. 1990. Multicast routing in datagram internetworks and extended LANs. *ACM Trans. Comput. Syst. 8,* 2, 85–110.

DIOT, C., LEVINE, B., LYLES, B., KASSEM, H., AND BALENSIEFEN, D. 2000. Deployment issues for the IP multicast service and architecture. *IEEE Netw. 14,* 1, 78–88.

DOBRIAN, F., AWAN, A., JOSEPH, D., GANJAM, A., ZHAN, J., SEKAR, V., STOICA, I., AND ZHANG, H. 2011. Understanding the impact of video quality on user engagement. In *Proceedings of the ACM SIGCOMM*.

FENG, C., LI, B., AND LI, B. 2009. Understanding the performance gap between pull-based mesh streaming protocols and fundamental limits. In *Proceedings of the IEEE INFOCOM*.

FIELDING, R., GETTYS, J., MOGUL, J., FRYSTYK, H., MASINTER, L., LEACH, P., AND BERNERS-LEE, T. 1999. Internet engineering task force. RFC 2616.

FLOYD, S., HANDLEY, M., PADHYE, J., AND WIDMER, J. 2000. Equation-based congestion control for unicast applications. In *Proceedings of the SIGCOMM*.

FORGIE, J. 1979. ST-A Proposed Internet Stream Protocol. http://www.rfc-editor.org/ien/ien119.txt.

GHODSI, A., SHENKER, S., KOPONEN, T., SINGLA, A., RAGHAVAN, B., AND WILCOX, J. 2011. Information-centric networking: Seeing the forest for the trees. In *Proceedings of the ACM Workshop on Hot Topics in Networks*.

GOUACHE, S., BICHOT, G., BSILA, A., AND HOWSON, C. 2011. Distributed & adaptive HTTP streaming. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*.

HAVEY, D., CHERTOV, R., AND ALMEROTH, K. 2012. Receiver driven rate adaptation for wireless multimedia applications. In *Proceedings of the ACM MMSys*.

HO, T., KOETTER, R., MEDARD, M., KARGER, D., AND EFFROS, M. 2003. The benefits of coding over routing in a randomized setting. In *Proceedings of the International Symposium on Information Theory (ISIT)*.

HOSSEINI, M., AHMED, D., SHIRMOHAMMADI, S., AND GEORGANAS, N. 2007. A survey of application-layer multicast protocols. *IEEE Comm. Surv. Tutor. 9,* 3, 58–74.

HUA, K. A., CAI, Y., AND SHEU, S. 1998. Patching: A multicast technique for true video-on-demand services. In *Proceedings of the ACM International Conference on Multimedia*.

HUA, K. A. AND SHEU, S. 1997. Skyscraper broadcasting: A new broadcasting scheme for metropolitan video-on-demand systems. *ACM SIGCOMM Comput. Commun. Rev. 27*, 89–100.

HUANG, Y., FU, T. Z., CHIU, D.-M., LUI, J. C., AND HUANG, C. 2008. Challenges, design and analysis of a large-scale P2P-VoD system. In *Proceedings of the ACM SIGCOMM*.

HUANG, Z., MEI, C., LI, L.-E., AND WOO, T. 2011. CloudStream: Delivering high-quality streaming videos through a cloud-based SVC proxy. In *Proceedings of the IEEE INFOCOM*.

INFONETICS. 2011a. http://www.infonetics.com/pr/2011/lte-deployment-strategies-service-provider-survey-highlights.asp.

INFONETICS. 2011b. IPTV Services Getting Highly Personalized, Highly Integrated, Portable, On-Demand. http://www.infonetics.com/.

JACOBS, S. AND ELEFTHERIADIS, A. 1998. Streaming video using dynamic rate shaping and TCP congestion control. *J. Visual Commun. Image Rep. 9,* 3, 211–222.

JANNOTTI, J., GIFFORD, D. K., JOHNSON, K. L., KAASHOEK, M. F., AND O'TOOLE, JR., J. W. 2000. Overcast: Reliable multicasting with an overlay network. In *Proceedings of the 4th Symposium on Operating System Design and Implementation (OSDI)*. Vol. 4.

JIANG, J., SEKAR, V., AND ZHANG, H. 2012. Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with festive. In *Proceedings of the ACM CoNEXT*.

KIM, T. AND AMMAR, M. H. 2001. A comparison of layering and stream replication video multicast schemes. In *Proceedings of the ACM NOSSDAV*.

KOSTIĆ, D., RODRIGUEZ, A., ALBRECHT, J., AND VAHDAT, A. 2003. Bullet: High width data dissemination using an overlay mesh. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*.

LI, B. AND NAHRSTEDT, K. 1999. A control-based middleware framework for quality of service adaptations. *IEEE J. Select. Areas Commun. 17,* 9, 1632–1650.

LIU, J. AND XU, J. 2004. Proxy caching for media streaming over the internet. *IEEE Commun. Mag. 42,* 8, 88–94.

LIU, Z., WU, C., LI, B., AND ZHAO, S. 2010. UUSee: Large-scale operational on-demand streaming with random network coding. In *Proceedings of the IEEE INFOCOM*.

MAGHAREI, N., REJAIE, R., AND GUO, Y. 2007. Mesh or multiple-tree: A comparative study of live P2P streaming approaches. In *Proceedings of the IEEE INFOCOM*.

MCCANNE, S., JACOBSON, V., AND VETTERLI, M. 1996. Receiver-driven layered multicast. *ACM SIGCOMM Comput. Commun. Rev. 26*, 117–130.

MOK, R. K., LUO, X., CHAN, E. W., AND CHANG, R. K. 2012. QDASH: A QoE-aware dash system. In *Proceedings of the ACM MMSys*. 11–22.

MPEG. 2010. Dynamic adaptive streaming over HTTP. http://mpeg.chiariglione.org/.

NAHRSTEDT, K., YANG, Z., WU, W., AREFIN, A., AND RIVAS, R. 2011. Session management in 3D tele-immersive systems. In *Hot Topics in Multimedia Series*, vol. 51, Springer-Verlag, Berlin, Chapter: International Journal of Multimedia Tools and Applications (MTAP), 23–43.

NIU, D., XU, H., LI, B., AND ZHAO, S. 2012. Quality-assured cloud bandwidth auto-scaling for video-on-demand applications. In *Proceedings of the IEEE INFOCOM*.

Ooi, W. T., Van Renesse, R., and Smith, B. 2000. The design and implementation of programmable media gateways. In *Proceedings of the ACM NOSSDAV*.

Pai, V., Kumar, K., Tamilmani, K., Sambamurthy, V., and Mohr, A. 2005. Chainsaw: Eliminating trees from overlay multicast. In *Proceedings of the 4th International Workshop on Peer-to-Peer Systems (IPTPS)*. 127–140.

Pallis, G. and Vakali, A. 2006. Insight and perspectives for content delivery networks. *Commun. ACM 49,* 1, 101–106.

Park, J.-S., Gerla, M., Lun, D. S., Yi, Y., and Medard, M. 2006. Codecast: A network-coding-based ad hoc multicast protocol. *IEEE Wirel. Commun. 13,* 5, 76–81.

Peng, G. 2004. CDN: Content Distribution Network. arXiv preprint cs/0411069.

Podolsky, M., Yano, K., and McCanne, S. 1999. A RTCP-based retransmission protocol for unicast RTP streaming multimedia. IETF, draft-podolsky-avt-rtprx-00.txt.

Pujol, J. M., Erramilli, V., Siganos, G., Yang, X., Laoutaris, N., Chhabra, P., and Rodriguez, P. 2010. The little engine(s) that could: Scaling online social networks. In *Proceedings of the ACM SIGCOMM*.

Rodrigues, T., Benevenuto, F., Cha, M., Gummadi, K.-P., and Almeida, V. 2011. On word-of-mouth based discovery of the Web. In *Proceedings of the ACM IMC*.

Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., Schooler, E., et al. 2002. RFC 3261, SIP: Session Initiation Protocol.

Saxena, M., Sharan, U., and Fahmy, S. 2008. Analyzing video services in Web 2.0: A global perspective. In *Proceedings of the ACM NOSSDAV*.

Schulzrinne, H., Casner, S., Frederick, R., and Jacobson, V. 1996. RFC 1889, RTP: A Transport Protocol for Real-Time Applications.

Schulzrinne, H., Rao, A., and Lanphier, R. 1998. RFC 2326, Real Time Streaming Protocol (RTSP).

Sen, S., Rexford, J., and Towsley, D. 1999. Proxy prefix caching for multimedia streams. In *Proceedings of the IEEE INFOCOM*.

Sheu, S., Hua, K. A., and Tavanapong, W. 1997. Chaining: A generalized batching technique for video-on-demand systems. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*.

Silverston, T. and Fourmaux, O. 2007. Measuring P2P IPTV systems. In *Proceedings of the ACM NOSSDAV*.

SNL. 2011. Global multichannel markets special report: The state of global IPTV. http://www.sml.com/.

Song, H., Mahimkar, A., Ge, Z., Wang, J., Yates, J., and Zhang, Y. 2011. Q-Score: Proactive service quality assessment in a large IPTV system. In *Proceedings of the ACM IMC*.

Sripanidkulchai, K., Ganjam, A., Maggs, B., and Zhang, H. 2004. The feasibility of supporting large-scale live streaming applications with dynamic application end-points. In *Proceedings of the ACM SIGCOMM*.

Statistics. 2013. Last accessed March 24, 2013 from http://www.youtube.com/yt/press/statistics.html.

Stockhammer, T. 2011. Dynamic adaptive streaming over HTTP: Standards and design principles. In *Proceedings of the ACM Conference on Multimedia Systems*.

Stoica, I. 2010. It's not the cost, it's the quality! In *Proceedings of the 9th International Workshop on Peer-to-Peer Systems (IPTPS)*.

Thomas, V. 1998. White paper: Ip multicast in realsystem g2. *RealNetworks, Inc*, 1–14.

Tran, D. A., Hua, K. A., and Do, T. 2003. Zigzag: An efficient peer-to-peer scheme for media streaming. In *Proceedings of the IEEE INFOCOM*.

Vakali, A. and Pallis, G. 2003. Content delivery networks: Status and trends. *IEEE Internet Comput. 7,* 6, 68–74.

Venkataraman, V., Yoshida, K., and Francis, P. 2006. Chunkyspread: Heterogeneous unstructured tree-based peer-to-peer multicast. In *Proceedings of the 5th International Workshop on Peer-to-Peer Systems (IPTPS)*. 2–11.

Wang, F., Liu, J., and Chen, M. 2012a. CALMS: Cloud-assisted live media streaming for globalized demands with time/region diversities. In *Proceedings of the IEEE INFOCOM*.

Wang, F., Xiong, Y., and Liu, J. 2007. mTreebone: A hybrid tree/mesh overlay for application-layer live video multicast. In *Proceedings of the IEEE ICDCS*.

Wang, M. and Li, B. 2007. R^2: Random push with random network coding in live peer-to-peer streaming. *IEEE J. Select. Areas Commun.*

Wang, Y., Wenger, S., Wen, J., and Katsaggelos, A. K. 2000. Error resilient video coding techniques. *IEEE Signal Process. Mag. 17,* 4, 61–82.

Wang, Y. and Zhu, Q.-F. 1998. Error control and concealment for video communication: A review. *Proc. IEEE 86,* 5, 974–997.

Wang, Z., Sun, L., Chen, X., Zhu, W., Liu, J., Chen, M., and Yang, S. 2012b. Propagation-based social-aware replication for social video contents. In *Proceedings of the ACM Multimedia*.

Watson, M. 2011. HTTP adaptive streaming in practice. In *Proceedings of the ACM MMSys*.

Wu, D., Hou, Y., Zhu, W., Zhang, Y.-Q., and Peha, J. M. 2001. Streaming video over the Internet: Approaches and directions. *IEEE Trans. Circuits Syst. Video Technol. 11,* 3, 282–300.

Wu, Y., Wu, C., Li, B., Qiu, X., and Lau, F.-C. 2011. CloudMedia: When cloud on demand meets video on demand. In *Proceedings of the IEEE ICDCS*.

Yu, F., Zhang, Q., Zhu, W., and Zhang, Y.-Q. 2003. QoS-adaptive proxy caching for multimedia streaming over the Internet. *IEEE Trans. Circuits Syst. Video Technol. 13,* 3, 257–269.

Zhang, L., Deering, S., Estrin, D., Shenker, S., and Zappala, D. 1993. RSVP: A new resource ReSerVation Protocol. *IEEE Netw. 7,* 5, 8–18.

Zhang, M., Luo, J.-G., Zhao, L., and Yang, S.-Q. 2005a. A peer-to-peer network for live media streaming—Using a push-pull approach. In *Proceedings of the ACM Multimedia*.

Zhang, M., Zhang, Q., Sun, L., and Yang, S. 2007. Understanding the power of pull-based streaming protocol: Can we do better? *IEEE J. Select. Areas Commun. 25,* 9, 1678–1694.

Zhang, Q., Zhu, W., and Zhang, Y.-Q. 2001. Resource allocation for multimedia streaming over the Internet. *IEEE Trans. Multimedia 3,* 3, 339–355.

Zhang, X., Liu, J., Li, B., and Yum, Y.-S. 2005b. CoolStreaming/DONet: A data-driven overlay network for peer-to-peer live media streaming. In *Proceedings of the IEEE INFOCOM*.

Zhu, W., Luo, C., Wang, J., and Li, S. 2011. Multimedia cloud computing. *IEEE Signal Process. Mag. 28,* 3, 59–69.

Received October 2012; revised March, June, July 2013; accepted July 2013