

**Two Decades of Internet Video Streaming: A Retrospective View**

**Peer**

|  |  |  |
| --- | --- | --- |
| Journal: |  | *Transactions on Multimedia Computing Communications and Applications* |
| Manuscript ID: |  | TOMCCAP-2012-0104.R3 |
| Manuscript Type: |  | 20 years of ACM MM |
| Date Submitted by the Author: |  | n/a |
| Complete List of Authors: |  | Li, Baochun; University of Toronto, Electrical and Computer Engineering Wang, Zhi; Tsinghua University, Computer Science and Technology  Liu, Jiangchuan; Simon Fraser University, Computer Science  Zhu, Wenwu; Tsinghua University, Computer Science and Technology |
| Computing Classification  Systems: |  |  |
| Video streaming, Multicast protocols, Peer-to-peer systems, Cloud  computing, Transport protocols, Social media | |
|  |  |
|  | | |



**1**

1

2

3

4

5

6

7

8 Two Decades of Internet Video Streaming:

9

10 A Retrospective View

11

1. BAOCHUN LI, University of Toronto
2. ZHI WANG, Tsinghua University
3. JIANGCHUAN LIU, Simon Fraser University
4. WENWU ZHU, Tsinghua University

16

17

1. For over two decades, video streaming over the Internet has received a substantial amount of attention from both academia
2. and industry. Starting from the design of transport protocols for streaming video, research interests have later shifted to the
3. peer-to-peer paradigm of designing streaming protocols at the application layer. More recent research has focused on building
4. more practical and scalable systems, using Dynamic Adaptive Streaming over HTTP. In this paper, we provide a retrospective
5. view of the research results over the past two decades, with a focus on peer-to-peer streaming protocols and the effects of cloud
6. computing and social media.
7. Categories and Subject Descriptors: C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed appli-*
8. *cations*
9. General Terms: Multimedia streaming
10. Additional Key Words and Phrases: Video streaming, Multicast, P2P streaming, HTTP streaming, Cloud computing, Social
11. media
12. **ACM Reference Format:**
13. Li, B., Wang, Z., Liu, J., and Zhu, W.. 2013. Two Decades of Internet Video Streaming:
14. A Retrospective View. ACM Trans. Multimedia Comput. Commun. Appl. 2, 3, Article 1 (October 2013), 20 pages.

32 DOI = 10.1145/0000000.0000000 <http://doi.acm.org/10.1145/0000000.0000000>

33

34

35

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

42

43

1. Addresses of the authors: B. Li, Department of Electrical and Computer Engineering, University of Toronto, 10 King’s
2. College Road, Toronto, Ontario M5S 3G4, Canada; email: bli@eecg.toronto.edu; Z. Wang, W. Zhu, Department of Com-
3. puter Science and Technology, Tsinghua University, Beijing 100084, P. R. China; email: [wangzhi04@mails.tsinghua.edu.cn,](mailto:wangzhi04@mails.tsinghua.edu.cn)
4. 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.](mailto:jcliu@cs.sfu.ca)

48

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided

1. that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page
2. or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM
3. must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to
4. lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be
5. requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or [permissions@acm.org.](mailto:permissions@acm.org)

54

c 2013 ACM 1551-6857/2013/10-ART1 $15.00

*Ⓧ*

55 DOI 10.1145/0000000.0000000 <http://doi.acm.org/10.1145/0000000.0000000>

56

57 ACM Transactions on Multimedia Computing, Communications and Applications, Vol. 2, No. 3, Article 1, Publication date: October 2013.

1

2

3

4

5

6 1:2 *•* B. Li *et al.*

7

1. is more preferred and fundamentally different from having to wait for a download to complete before
2. playback. As a result of this preference, video streaming has received a substantial amount of research
3. attention in the past two decades, from both academia and industry.
4. In this paper, we seek to go back in the time machine, and present a retrospective view of the history
5. of Internet video streaming by covering its main milestones in the past two decades of research and
6. development. Following the chronological order, we present three stages of research development on
7. Internet video streaming:
8. **Client-server video streaming.** During the 1990s and early 2000s, research attention has mostly
9. focused on the design and implementation of new streaming protocols, such as the design of the Real-
10. time Transport Protocol (RTP) specifically for streaming media. The initial set of streaming protocols
11. were incorporated by the media players, which were used to receive video streams from the streaming
12. servers over the Internet.
13. **Peer-to-peer video streaming.** To scale up to an ever increasing number of users, peer-to-peer
14. (P2P) video streaming has been studied extensively in the past decade, and has widely been applied
15. to both live and on-demand video streaming. The design of P2P streaming protocols was based on the
16. philosophy that end hosts, called “peers,” were able to serve as both clients and servers, as opposed
17. to the traditional client-server design where end hosts were only able to consume video. Real-world
18. systems, such as PPLive, succeeded to serve thousands of video streams to millions of users, while
19. consuming a modest amount of bandwidth at streaming servers.
20. **HTTP video streaming over the cloud.** To use P2P streaming protocols, users were required to
21. download and install dedicated applications that implemented these protocols. However, it is much
22. more convenient for users to stream videos directly over the web using a standard Internet web
23. browser, without the need to download and install third-party applications. With HTTP video stream-
24. ing, a video stream is divided into a sequence of small chunks that can be downloaded using HTTP.
25. Due to the convenience of using the standard HTTP protocol, HTTP streaming has been widely used
26. in the industry, with streaming servers hosted by cloud computing platforms. As a result, we are now

34

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

44

45

1. 2. INTERNET VIDEO STREAMING USING THE CLIENT-SERVER ARCHITECTURE
2. The development of video compression technologies in the 1980s and the growth and popularity of
3. the Internet in the 1990s have motivated the concept of streaming videos over the Internet to a large
4. number of clients. For much of the 1990s, research in both academia and industry focused on the
5. design and implementation of new protocols for Internet video streaming from dedicated streaming
6. servers. Built on top of the Internet Protocol (IP), these new video streaming protocols were designed
7. to support Quality of Service (QoS) efficiently over the best-effort Internet. In this section, we present
8. a small number of key milestones to show how traditional client-server video streaming protocols have
9. evolved in the first decade, before the advent of peer-to-peer (P2P) video streaming. While we are not
10. able to cover all the important research results in this section, there exist excellent tutorial papers in

1

2

3

4

5

6 Two Decades of Internet Video Streaming:A Retrospective View *•* 1:3

7

8 the literature on video streaming protocols in the early stages of research [Cleary 1995; Aurrecoechea

9 et al. 1998].

10

1. 2.1 Transport Protocols for Video Streaming
2. In the early 1990s, resource reservation protocols for achieving Quality of Service (QoS) over the Inter-
3. net, such as RSVP [Zhang et al. 1993], were superseded by new and simpler transport protocols that
4. did not depend on any means to reserve bandwidth in the best-effort Internet. To detect packet loss and
5. compensate jitter during transmissions over an IP network, the Real-time Transport Protocol (RTP)
6. was proposed for end-to-end real-time transfer of stream data. RTP was designed with two fundamen-
7. tal principles: application-layer framing (*i.e.*, framing for video data should be performed properly by
8. the application layer) and integrated layer processing (*i.e.*, integrating multiple layers into one to al-
9. low efficient cooperation) [Schulzrinne et al. 1996]. Based on the User Datagram Protocol (UDP), RTP
10. defined a standardized packet format for delivering video over IP, and was designed for end-to-end
11. real-time transfer of stream data. The RTP Control Protocol (RTCP), also based on UDP, was designed
12. to monitor transmission statistics and QoS, and to achieve synchronization across multiple streams.
13. Important commercial and open-source products, such as QuickTime, have supported RTP. RTP has
14. also been adopted by 3GPP, which has been incorporated in nearly all mobile devices.
15. As new transport protocols became critically important for Internet video streaming, the Internet
16. Engineering Task Force (IETF) standardized the RTP/RTCP/RTSP protocol suite [Schulzrinne et al.
17. 1996; Schulzrinne et al. 1998], designed specifically for Internet video streaming. Beyond RTP and
18. RTCP, the Real Time Streaming Protocol (RTSP) [Schulzrinne et al. 1998] in the protocol suite was de-
19. signed to create and maintain video sessions, and to provide VCR-style control functionality, enabling
20. users to pause, resume, or seek in video streams just like local playback. RTSP is an example of session

31

1. control protocols, similar to the Session Initiation Protocol (SIP) that was later designed [Rosenberg
2. et al. 2002]. Session control protocols have since played an important role in the management of ac-
3. tive sessions, even in recent protocols in the context of 3D immersive environments [Nahrstedt et al.
4. 2011], where research progress has been made on session management protocols in 3D tele-immersive
5. systems [Baldino et al. 2011].
6. 2.2 Rate Control and Rate Shaping

38

1. To support video streaming over the Internet, it is believed that transport protocols should be designed
2. to meet real-time video delivery needs, by maximizing the video quality in the presence of packet loss.
3. Bursty packet losses have a devastating effect on the video quality, and they are typically caused
4. by network congestion. To support smooth video streaming sessions, rate control and rate shaping
5. protocols [Floyd et al. 2000; Jacobs and Eleftheriadis 1998] were proposed to minimize the possibility
6. of network congestion, by matching the rate of the video stream to the available network bandwidth.
7. *Rate control* was a technique designed to determine the sending rate of a video stream based on the
8. estimated available bandwidth in the network. Traditional rate control schemes may be classified into
9. three categories [Wu et al. 2001]: source-based, receiver-based, and hybrid rate control. The objective
10. of *rate shaping* was to match the streaming rate to the target bandwidth [Jacobs and Eleftheriadis
11. 1998], and it was required for source-based rate control, as a stored video is pre-compressed at a rate
12. that may not match the available bandwidth in the network. There were many types of rate shapers,
13. such as the codec filter, frame-dropping filter, layer-dropping filter, frequency filter, and re-quantization
14. filter [Wu et al. 2001]. In the late 1990s and early 2000s, SureStream was proposed and became part of
15. a product line, called G2, from Real Networks. SureStream created multiple encodings of each content,
16. so that a client may choose one of the available qualities, and was even capable of switching between
17. encodings during playback [Thomas 1998]. Zhang *et al.* [Zhang et al. 2001] studied the problem of rate

1

2

3

4

5

6 1:4 *•* B. Li *et al.*

7

1. adaptation according to the estimated network bandwidth, using each video’s rate distortion function
2. under various network conditions.

10

1. 2.3 Error Control
2. Due to the best effort nature of the Internet, packet loss is inevitable, thereby having a significant
3. impact on the perceptual quality of video streams. Error control [Wang and Zhu 1998] was proposed
4. to ensure a smooth video streaming experience to the users, even with the presence of packet losses.
5. A large number of error control mechanisms have been proposed in the first decade of research, which
6. can be loosely organized into the following categories:
7. *Forward Error Correction (FEC)*: The idea of FEC was to add redundant information so that the

*♦*

1. original message can be reconstructed in the presence of packet loss [Albanese et al. 1996]. Bolot *et*
2. *al.* [Bolot and Turletti 1996] proposed to use error control mechanisms based on FEC in video stream-
3. ing, and evaluated its benefits. Based on the type of redundant information to be added, FEC schemes
4. can take the forms of channel coding, source coding, and joint source/channel coding.
5. *Error-resilient encoding*: The objective of error-resilient encoding was to improve the robustness of

*♦*

1. compressed video to packet losses. The standardized error-resilient encoding schemes included resyn-
2. chronization marking, data partitioning, and data recovery [Wang et al. 2000]. However, these ap-
3. proaches were designed for error-prone environments like wireless channels, and were not applicable
4. in the context of the Internet. For video streaming over the Internet, the boundary of a packet has
5. already provided a synchronization point in the variable-length coded bit-stream at the receiver side.
6. *Error concealment*: Error concealment was a technique in which an error in video delivery was

*♦*

1. replaced by synthetic content, often interpolated from other parts of the signal. It can be executed by

30

1. the source (*i.e.*, forward error concealment), by the receiver (*i.e.*, error concealment by postprocessing),
2. and by both the source and receiver (*i.e.*, interactive error concealment) to improve the robustness of
3. the compressed video before packet loss actually happens [Wang and Zhu 1998].
4. *Delay-constrained retransmission*: Retransmission is usually dismissed as a method to recover

*♦*

1. lost packets in real-time video, since a retransmitted packet may miss its playback time. However, if
2. the one-way delay is relatively short with respect to the maximum allowable delay, a retransmission-
3. based approach (called delay-constrained retransmission) may still be a viable option for error control
4. [Podolsky et al. 1999].
5. 2.4 Proxy Caching

40

1. In the early days of video streaming, both the number of videos and the number of users were small.
2. Videos could be served by a single server, which in many cases not only stored the videos, but also
3. uploaded them to the users. Such an architecture quickly became infeasible when more and more
4. videos were made available online. An important technique for improving the scalability and reducing
5. the latency was *proxy caching*, which was based on the observation that different clients will request
6. many of the same videos.
7. While proxy caching was widely used in web content distribution, video caching had a number of
8. different focuses. On one hand, since the content of a video object was rarely updated, management
9. issues such as cache consistency and coherence were less critical in video caching. On the other hand,
10. given the high resource requirement of videos, effective management of proxy cache resources (*i.e.*, disk
11. space, disk I/O, and network I/O) became more challenging. Sen *et al.* [Sen et al. 1999] demonstrated
12. that by only caching the prefixes of videos, a large amount of videos can be served by a few megabytes
13. of buffer space at the proxy storage. To improve caching efficiency, Hua *et al.* analytically studied the
14. storage requirement in the context of on-demand streaming [Hua and Sheu 1997; Hua et al. 1998].
15. Yu *et al.* [Yu et al. 2003] proposed a QoS-adaptive proxy caching scheme for video streaming over the

1

2

3

4

5

6 Two Decades of Internet Video Streaming:A Retrospective View *•* 1:5

7

1. Internet, considering the heterogeneous network conditions and video characteristics. Other effective
2. partial caching strategies, such as sliding-interval caching, segment caching, and rate-split caching,
3. have also been proposed in the literature [Liu and Xu 2004].

11

1. 2.5 IP Multicast for Video Streaming
2. The ever increasing user population also called for a revisit to the original one-to-one Internet com-
3. munication paradigm. Maintaining unicast sessions for each user quickly became infeasible as the
4. number of users scaled up. Many emerging applications, including Internet TV and live event broad-
5. cast, required the support for video multicast, *i.e.*, simultaneously delivering a video stream to a large
6. number of clients.
7. IP multicast [Deering and Cheriton 1990] was proposed as an extension to IP unicast, with the
8. objective of providing efficient multipoint packet delivery. Given that the network topology was best
9. known in the network layer, multicast routing at this layer was also the most efficient. IP multicast
10. retained the semantics of IP and allowed users to dynamically join or leave multicast groups.
11. In the early 1990s, end-to-end adaptation schemes were mainly sender-driven, where the sender ad-
12. justed its transmission rate according to some feedback from a single receiver. This did not work well
13. for multicasting to heterogenous receivers without a common target rate. One solution was stream
14. replication, which could be viewed as a trade-off between single-rate multicast and multiple point-
15. to-point connections [Kim and Ammar 2001]. Its feasibility was well justified in a typical multicast
16. environment where the bandwidth of receivers usually followed some clustered distribution. As a re-
17. sult, a limited number of streams could be used to match these clusters to achieve a reasonably good
18. performance.

30

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

41

1. 2.6 Application-Layer Multicast for Video Streaming
2. Today, the scope and reach of IP multicast remain limited, as many Internet Service Providers (ISPs)
3. simply block or disable IP multicast due to various security and economic concerns [Diot et al. 2000].
4. The idea of using the application layer for multicast, called application-layer multicast (ALM), was pro-
5. posed around 2000 [Sheu et al. 1997; Chu et al. 2000]. Though both application-layer and IP multicast
6. require intermediate nodes in the network topology to support the replication of data packets, it was
7. much easier to implement multicast at the application layer on end hosts, as opposed to the network
8. layer at switches and routers [Hosseini et al. 2007]. Cui *et al.* [Cui et al. 2004] showed that with re-
9. spect to bandwidth consumption on the backbone network, the benefit introduced by application-layer
10. multicast overshadowed the topological inefficiency introduced.
11. Chu *et al.* [Chu et al. 2000] showed that application-layer multicast performed reasonably well
12. as compared to IP multicast, incurring a comparable delay and a reasonable amount of bandwidth
13. penalty. In the initial proposals, a single multicast tree was constructed for each multicast session. For
14. example, Overcast [Jannotti et al. ] attempted to organize a bandwidth-efficient tree by letting peers

1

2

3

4

5

6 1:6 *•* B. Li *et al.*

7

8



Source

S

C

A

B

D

F

E

9

10

11

12

13

14

15

16

17

18

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

20



Source

S

C

A

B

D

F

E

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

1. join near the root, and then migrating them down the tree to a position according to bandwidth avail-
2. ability. Tran *et al.* [Tran et al. 2003] proposed ZIGZAG, in which the video server distributes chunks to
3. many clients by organizing them into an appropriate tree rooted at the server. The advantage of such
4. a design, shown in Fig. 1, was its simplicity.
5. The design of a single multicast tree suffered from a number of problems. First, such a design may
6. not be *fair* to all the peers. As we can easily see from Fig. 1, when the tree was formed, some peers
7. were chosen to be interior nodes that must contribute their upload bandwidth to support others, while
8. others were leaf nodes that were not required to contribute any upload bandwidth. Even if fairness was
9. not a concern, the multicast rate that a child node can enjoy was restricted by the upload bandwidth
10. available at its parent. In case the parent node left the multicast session, the children would be left in
11. the cold, waiting for a new parent.
12. To improve fairness in application-layer multicast, it has been proposed since 2003 that streams can
13. be split to multiple “slices,” and distributed across a “forest” of multiple interior-node-disjoint multicast
14. trees [Castro et al. 2003], or a mesh overlay on top of a tree [Kostic´ et al. 2003]. Fig. 2 illustrates an
15. example of multicasting with two trees, constructed with the intention that a majority of nodes that
16. are interior nodes in one tree will be leaf nodes in the other tree. By distributing the responsibility of
17. contributing uploading bandwidth to most of the peers in the multicast session, the fairness problem
18. is mitigated, yet the robustness problem remains to be solved.

39

40

1. 3. PEER-TO-PEER VIDEO STREAMING
2. In contrast to the client-server streaming model where the consumption and supply of resources (*e.g.*,
3. bandwidth) is always decoupled, peer-to-peer streaming evolves in which *peers* are both suppliers and
4. consumers of resources. The peer-to-peer design philosophy takes advantage of the ability of partici-
5. pating end hosts, or peers, in a multicast group to contribute their uplink bandwidth. The peer-to-peer
6. design has two major advantages. *First*, it does not require the support from the underlying network
7. infrastructure, and as a result, it is cost-effective and easy to be deployed. *Second*, in the peer-to-peer
8. design, a peer is not only downloading a video stream, but also uploading it to other peers watching the
9. same program. As a result, it has the potential to scale up with the group size, as a stronger demand
10. also generates more suppliers.
11. In retrospect, application-layer multicast and peer-to-peer share the same design philosophy. Application-
12. layer multicast is largely *push-based*, in that video streams are being “pushed” to the receivers along
13. one or more trees. As the tree topologies have been constructed before the streaming session starts,
14. each receiver in the trees only needs to forward the video stream to its downstream receivers, with
15. very minimal delays due to such forwarding.

1

2

3

4

5

6 Two Decades of Internet Video Streaming:A Retrospective View *•* 1:7

7

8 Whole File

9



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

10

11 (a)

12

Sliding Window

13

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |

14

15

16 Playback point

17 (b)

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

20

21

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

1. The design philosophy in BitTorrent has converged with academic solutions in application-layer mul-
2. ticast, and a new generation of *pull-based* peer-to-peer streaming protocols on random mesh topologies
3. has emerged, independently proposed in Chainsaw [Pai et al. 2005] and CoolStreaming [Zhang et al.
4. 2005b]. In such protocols, peers exchange information with their neighbors periodically about what
5. video segments each of them has in their buffers, and a missing video segment must be explicitly
6. requested and transferred from one of the neighbors who has it. In comparison, application-layer mul-
7. ticast based on multicast trees adopts a more rigid design [Venkataraman et al. 2006; Magharei et al.
8. 2007], in that the structure of each tree needs to be actively managed as peers join and leave the ses-
9. sion. As a result, pull-based protocols are much simpler to design and more amenable to real-world
10. implementations [Zhang et al. 2007].
11. On the flip side of the coin, the main challenge of designing pull-based protocols is the timing con-
12. straints that new streaming protocols must observe: if video segments do not arrive in time, they are
13. not useful when it comes to the time of playing them back. In this section, we first present an example
14. of pull-based streaming protocols, and then illustrate how network coding is able to help address the
15. challenges of timing constraints in pull-based streaming.

37

38 3.1 Traditional Pull-based P2P Video Streaming

39

1. CoolStreaming [Zhang et al. 2005b] was one of the first real-world implementations of pull-based peer-
2. to-peer streaming protocols. In CoolStreaming, a peer maintains a partial view of other peers as its
3. *neighbors*, and schedules the transmission of video segments by sending outgoing requests to its neigh-
4. bors. In CoolStreaming, a video stream is divided into segments of a uniform length, and the availabil-
5. ity of active segments in the buffer of a peer is represented by a *buffer map*. Each peer continuously
6. exchange its buffer map with its neighbors, and then uses a *scheduling algorithm* to determine which
7. segment is to be fetched from which neighbor accordingly. An example of its neighboring relationship
8. is shown in Fig. 3.
9. As shown in Fig. 4, a dynamic *sliding window* of segments over time needs to be distributed, unlike
10. a fixed number of data blocks in file sharing (such as BitTorrent). As the sliding window moves for-
11. ward in the stream over time, blocks are to be received in approximately the same sequence as they
12. are played back, and out-of-order delivery can only occur within the confines of the sliding window.
13. Each of the blocks are distributed using a gossip protocol in a random mesh topology of neighbors (an
14. example of which has been shown in Fig. 3), requiring peers to “pull” blocks from one another. Since all
15. participating peers are roughly synchronized with respect to their points of playback, they are able to
16. periodically exchange the states of their respective buffers in the sliding window. Based on the knowl-

1

2

3

4

5

6 1:8 *•* B. Li *et al.*

7

1. edge of block availability in each other’s sliding windows, a peer sends requests to its neighbors in
2. order to “pull” blocks it has yet to receive.
3. One of the main advantages of the pull-based design is its resilience to failures. A peer can depart
4. either gracefully or accidentally due to an unexpected failure. In either case, the departure can be
5. easily detected after a period of idle time, and an affected peer can quickly react by requesting the
6. missing segments from its other neighbors.
7. Fundamentally, setting up and maintaining trees in application-layer multicast is similar to setting
8. up a connection in a telephone network: states of parent-child relationships are established so that
9. data blocks can be transmitted without explicit requests taking place. In contrast, the distribution
10. of data blocks in a pull-based protocol is similar to *gossiping* in a social setting. Following such a
11. gossiping philosophy to distribute each of the data blocks to all the peers in the multicast session,
12. early peer-to-peer video streaming systems, such as CoolStreaming, have motivated a large number
13. of production-quality real-world implementations in the industry, several of which has become core
14. technologies in start-up companies that specialized in live media streaming, including PPLive [Huang
15. et al. 2008] and PPStream [Silverston and Fourmaux 2007].
16. Even though the gossiping philosophy incurs some delays in live streaming systems due to explicit
17. requests and periodic information exchanges, its most salient advantage is its simplicity in design
18. and implementation. If the current sliding window of the media stream to be distributed is divided
19. into small media blocks, they will flow through the entire network wherever there exists idle upload
20. bandwidth that can be tapped into.

28

29 3.2 P2P Video Streaming with Network Coding

30

1. Traditional pull-based P2P streaming is based on the periodic exchange of buffer availability maps,
2. and its communication overhead cannot be overlooked. Intuitively, since the sliding window at a peer
3. advances itself over time, buffer availability maps need to be exchanged as frequently as needed,
4. which may lead to a substantial amount of overhead. To mitigate such an overhead, most practical live
5. streaming systems choose to exchange buffer states less frequently. An analytical study [Feng et al.
6. 2009], however, has attributed the performance gap between practical systems and their theoretically
7. optimal performance to the lack of timely exchanges of buffer states.
8. Since 2005, the use of *network coding* has emerged as a potential remedy to these challenges in
9. peer-to-peer video streaming systems [Liu et al. 2010; Park et al. 2006]. Network coding, first proposed
10. in the information theory community in 2000 [Ahlswede et al. 2000], recognized the ability to code
11. at intermediate network nodes in a communication session, in addition to the ability to forward and
12. to replicate incoming packets. In contrast, traditional multicast only recognized the ability to forward
13. and to replicate packets. In 2003, Ho *et al.* [Ho et al. 2003] have further proposed the concept of *random*
14. *network coding*, where a network node transmits on each of its outgoing links a linear combination of
15. incoming packets over a finite field, with randomly chosen coding coefficients.
16. It is natural to conceive a simple but new design of peer-to-peer video streaming systems, in which
17. peers, as end hosts, are able to forward, replicate, and *code* incoming video data blocks. But will the
18. use of network coding improve the performance, as compared to traditional pull-based P2P streaming?
19. In particular, would the use of random network coding be able to mitigate the problem of the commu-
20. nication overhead? Wang *et al.* [Wang and Li 2007] have raised such a question, and proposed a new
21. protocol called *R*2 that used random network coding to substantially improve the performance of live
22. streaming systems.
23. *R*2 divides the content of the media stream in a sliding window into *generations*, each of which
24. is further divided into *m* video blocks. With the introduction of generations in *R*2, we can afford to
25. design parameter settings so that a *video block* is much smaller than its counterpart in traditional live

1

2

3

4

5

6 Two Decades of Internet Video Streaming:A Retrospective View *•* 1:9

7

8

existing peers

new peer

by both

blocks in a generation

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | complete | |  | | |  | |
|  |  | | | | | |  | |  | | | |
|  |  |  |  |  |  | missing | |  | | |  | |
|  | served | | | |  | |  | |  |  | |  |

9

10

11

12

13

14

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

15

16

17

18

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

21

1. streaming systems based on random gossiping. This is due to the fact that buffer states only need to
2. be exchanged at the granularity of a generation (one bit to represent each generation), rather than a
3. block. With the same amount of communication overhead to exchange buffer states, the size of a single
4. block can be much smaller in *R*2. When a peer serves a generation *s* to its downstream target peer *p*, it
5. linearly encodes all the video blocks it has received so far from *s* using random coefficients in GF(28),
6. and then transmits the coded block to *p*. Since each peer buffers coded blocks it has received so far, it
7. is able to linearly combine those from the same generation using random coefficients.
8. Since live streaming systems have timing requirements during playback, *R*2 uses *random push*
9. instead of “pull” to transmit data: each peer randomly selects a small number of downstream peers.
10. When serving a chosen downstream peer, it then randomly selects a generation to code within, among
11. those that the downstream peer has not yet completely received. If generations closer to the point of
12. playback have not been completely received, they are given a higher priority as they are more urgent.
13. There are three clear advantages brought forth by the use of random network coding in *R*2. *First*,
14. it greatly simplifies protocol design. Since coded blocks within a generation are equally useful, a peer
15. only needs to blindly push coded blocks in the same generation till the downstream peer has obtained
16. a sufficient number of them to decode. This eliminates the need of sending explicit requests to “pull”
17. missing blocks, and saves the communication overhead associated with these requests. *Second*, *R*2
18. induces much less overhead involved in buffer state exchanges, due to a smaller number of generations
19. in the sliding window. *Finally*, *R*2 allows for “perfect collaboration” among multiple upstream peers
20. when serving a downstream peer. Shown in Fig. 5, as a new peer joins the session, multiple existing
21. peers are able to collaborate and push fresh coded blocks in the first one or two generations after the
22. playback point, so that the rate of accumulating blocks in these generations is only limited by the new
23. peer’s available downlink bandwidth.

45

46

1. 4. HTTP VIDEO STREAMING OVER THE CLOUD AND SOCIAL MEDIA
2. Though P2P has proven to be highly efficient in video delivery, it is not convenient to regular users. In
3. order to cache video contents being played back and to serve other peers with P2P streaming, users are
4. usually required to install standalone applications and to keep some TCP or UDP ports open through
5. NAT and firewalls. This is not as convenient as the turn-key solution of using a standard web browser
6. to watch video streams.
7. After WebRTC was developed [Bergkvist et al. 2012], these issues seemed to be resolved; however,
8. due to its limited deployment by browsers, a solution to stream videos over HTTP has seen a substan-
9. tial amount of industry support. In this section, we present HTTP streaming, which was proposed even

1

2

3

4

5

6 1:10 *•* B. Li *et al.*

7

1. before P2P video streaming was largely deployed [Carmel et al. 2002], but has been used more widely
2. only recently.

10

1. 4.1 Dynamic Adaptive Streaming over HTTP
2. 4.1.1 *HTTP Streaming from Content Distribution Networks.* The rapid growth of HTTP streaming
3. is partly due to the extensive support from content distribution networks (CDN) [Peng 2004]. CDNs
4. deploy servers in multiple geographically diverse locations, distributed over multiple ISPs [Vakali
5. and Pallis 2003]. CDNs allow users to stream videos from a server close to them — user requests

16

1. are redirected to the best available server based on either geographical proximity or server load. Since
2. today’s CDNs are mainly designed and optimized to serve web contents [Pallis and Vakali 2006], HTTP
3. video streaming can be regarded as downloading video segments progressively from web servers via
4. the HTTP protocol, so that clients that support HTTP can seek to arbitrary positions in the media
5. stream by performing byte range requests to the web server [Fielding et al. 1999].
6. As a result, CDNs can be effectively used for high-quality TV content [Cahill and Sreenan 2004].
7. Adhikari *et al.* [Adhikari et al. 2012] have discovered that, Netflix, the leading on-demand Internet
8. video streaming provider, accounts for 29*.*7% of the peak downstream traffic in US, and it employs a
9. mix of datacenters and CDNs for video content distribution. Watson [Watson 2011] has studied the
10. *Dynamic Adaptive Streaming over HTTP* (DASH) framework used by Netflix, which is the largest
11. DASH-based streaming provider in the world.
12. HTTP video streaming overcomes the challenges of convenience in P2P streaming systems, as it
13. requires only a standard web browser to view video streams, and does not need to keep non-standard
14. TCP or UDP ports open though firewalls and NAT traversals. It works by breaking the overall video
15. stream into a sequence of small HTTP-based file downloads. Users progressively download these small
16. files, while a specific file is being played. Today, HTTP video streaming has been widely adopted by the
17. industry, as major video streaming solutions, such as Netflix, YouTube, and Hulu, all resort to HTTP
18. to stream their videos to the users.

35

1. 4.1.2 *Research Problems with DASH.* Due to the best-effort nature of streaming videos over the
2. Internet, *Dynamic Adaptive Streaming over HTTP* (DASH) has been proposed to adapt the streaming
3. rates from web servers. DASH was developed in 2010 [(MPEG) 2010], and has become a new standard
4. in 2011 [Stockhammer 2011] to enable high-quality streaming of media content over the Internet,
5. delivered from conventional HTTP web servers. It works by breaking encoded content with multiple
6. rates into small segments, such that a client may continuously adjust its requests according to the
7. estimates of local bandwidth availability. DASH can be represented by the following features:
8. *Segmentation*. In DASH, a video component is encoded and divided in multiple segments, with the
9. initialization segments containing the required information for initializing the media decoder, as well
10. as the media segments containing the video data.
11. *Media Presentation Description*. The media presentation description (MPD) describes how the seg-
12. ments form a video presentation [(MPEG) 2010]. Using the MPD, the clients request the segments for
13. smooth playback, and adjust bitrates or other attributes according to bandwidth estimates.
14. *Codec Independence*. DASH is codec agnostic, and its prime container is the MP4 and MPEG-TS. It
15. also allows seamless adoption of the upcoming improved HEVC video codec (*i.e.*, H265).
16. Despite its advantages and features, the increasing popularity of deploying DASH has also led to a
17. number of research problems:
18. *d Rate adaptation components*. DASH only defines the segmentation and the file description, and
19. leaves rate adaptation for either the client or the server to implement. There exists a number of so-
20. lutions along these lines in the recent literature. Clients can utilize multi-path and multi-server ap-

1

2

3

4

5

6 Two Decades of Internet Video Streaming:A Retrospective View *•* 1:11

7

1. proaches to receive video segments [Gouache et al. 2011]. In such receiver-driven approaches, the pro-
2. tocols are usually implemented at the application layer [Havey et al. 2012]. On the other hand, servers
3. are also able to adaptively change the bitrate for its clients, based on the perception of the client down-
4. load speed and server load. De Cicco *et al.* [De Cicco et al. 2011] have used a feedback mechanism to
5. allow clients to communicate with a server, which performs the rate adaptation.
6. *d Rate adaptation strategies*. Rate adaptation strategies determine how different versions of seg-
7. ments are received by users, to achieve the objectives including streaming stability, fairness, and high
8. quality. Akhshabi *et al.* [Akhshabi et al. 2011] have compared the bitrate adaptation of some popular
9. DASH client implementations, and observed that these implementations were either too aggressive or
10. too conservative, causing unfairness and inefficiency. Jiang *et al.* [Jiang et al. 2012] have proposed to
11. jointly take the efficiency, fairness and stability into consideration when adjusting the video bitrate.
12. *d User quality experience*. Though rate adaptation is highly correlated with the users’ video qual-
13. ity experience in DASH streaming, such a correlation may require a more detailed study. Cranley
14. *et al.* [Cranley et al. 2006] demonstrated the dynamic nature of user perception with adapting video
15. streaming. In the context of DASH, Mok *et al.* [Mok et al. 2012] have studied the user experience and
16. observe that users prefer a gradual quality change between the best and worst quality levels, instead
17. of abrupt switching. To better guide the design of rate adaptation strategies, a good metric to evaluate
18. the user experience is still an open area of research [Song et al. 2011].

26

27 4.2 Video Streaming from the Cloud

28

1. As HTTP streaming is increasingly employed by major content providers, more and more video stream-
2. ing systems have been built and deployed with HTTP streaming. As a result, the sharing of user gener-
3. ated content (UGC) (*e.g.*, YouTube) has fundamentally changed the video streaming landscape, where
4. 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.* [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 at-

tempts 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 prac-

tical challenges to be addressed when migrating video streaming services to the cloud. First, servers in

the cloud have diverse capacities and lease prices; and the billing cycle for the lease cannot be arbitrar-

ily short either — Amazon EC2 leases its virtual machines by the hour. As such, when being leased,

1

2

3

4

5

6 1:12 *•* B. Li *et al.*

7

8

9

10

11

12

13

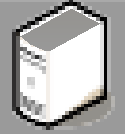
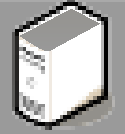
14

15

16

Cloud Server Being Leased

Cloud Server not Being Leased



Cloud Traffic User Traffic

**Cloud Layer**

Europe

Source

North America

17

18

19

20

21

22

**User**

**Layer**

25

26

27

…...

Australia

South America

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

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

[EC2 mec2]. Therefore, a VoD provider need to accurately predict its future demand, so that an ade-

32

33 quate number of new virtual machines can be leased to meet such demands with minimized costs [Niu

34 et al. 2012].

Fig. 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 global-

ized, 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.

49

1. 4.3 The Effects of User-Generated and Social Media
2. Compared to the Internet a decade ago, networked services in the web 2.0 era focus more on the user
3. experience, user participation and interaction with rich media. The users are now actively engaged to
4. be part of a social ecosystem, rather than passively receiving video streams.
5. When it comes to streaming services, one prominent example reflecting this change is YouTube.
6. Established in 2005, it is now serving well over 4 billion views a day, with most of the contents being

1

2

3

4

5

*•* 1:13

7

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 [Saxena et al. 2008].

While peer-to-peer mechanisms would be a candidate to scale 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.

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

3839

40

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

42

43

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 Fig. 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

1

2

3

4

5

6 1:14 *•* B. Li *et al.*

7

1. video could still be quite high. After multiple rounds, the rate can easily reach over 90% given the
2. small world of the videos [Cheng and Liu 2009].

1. The latest development of general social network applications, in particular, Facebook and Twitter,
2. have further changed the information distribution landscape and even people’s daily life. These online
3. Social Networking Services (SNS) directly connect people through cascaded relations, and information
4. thus spread much faster and more extensively than through conventional web portals or newsgroup
5. services.
6. Such word-of-mouth spreading [Rodrigues et al. 2011] has also drastically expanded the ways of
7. discovering the videos, beyond traditional web browsing and searching. There have been pioneering
8. studies on information propagation over generic networks and, more recently, over social networks
9. [Budak et al. 2011]. Yet their focuses have been largely on the conventional text or image objects
10. and on their stationary coverage among users. The sheer and ever increasing data volume, the broad
11. coverage, and the long access durations of video objects have presented more significant challenges
12. than other types of objects to the SNS management, and to that of video sharing sites. A measurement
13. study [Broxton et al. 2010] based on YouTube data has shown that, between April 2009 and March
14. 2010, 25% of views on YouTube come from social sharing. Despite recent work towards this direction
15. [Wang et al. 2012b], the characteristics of such requests from OSNs have yet to be comprehensively
16. measured at large scales.

32

33 5. LESSONS LEARNED AND CONCLUDING REMARKS

34

1. In retrospect, the rise of research interests in video streaming systems was largely fueled by user de-
2. mand and the increasingly abundant availability of bandwidth in the Internet, reflected both in the
3. uplink bandwidth at end users (motivating the peer-to-peer philosophy), and in the downlink band-
4. width at the servers in the cloud datacenters and in Content Distribution Networks (CDNs).
5. The popularity of the peer-to-peer design paradigm originated from its advantage of a *clean slate*
6. starting point, in that it was not confined by any legacy protocols in the Internet. Due to the freedom of
7. designing overlay topologies, the peer-to-peer paradigm was amenable to both theoretical and practical
8. treatments, from modeling, analyses, to implementation and deployment. There have been technical
9. challenges that have received heated discussions from researchers in this context. One example is
10. on the tree-based structured versus pull-based structureless overlays. Both have shown their success
11. in practical deployment, yet neither completely overcomes the challenges from a dynamic peer-to-
12. peer environment. The selling point for pull-based systems is their simplicity, but they suffer from a
13. latency-overhead trade-off with the exchange of buffer maps [Venkataraman et al. 2006; Zhang et al.
14. 2005a]. A tree-based system does not suffer from this trade-off, but has to face the inherent instability,
15. maintenance overhead, and bandwidth under-utilization. There have been attempts toward a hybrid
16. design [Venkataraman et al. 2006; Wang et al. 2007]. It has been shown that most of the data blocks
17. delivered through a pull-based mesh overlay essentially follow a specific tree structure or a small set
18. of trees, mostly of stable peers [Wang et al. 2007]. As such, while maintaining *a priori* topology for
19. all the peers is costly, optimizing the organization for a relatively stable core subset — with others
20. being organized through a mesh — could simultaneously achieve high efficiency with low overhead
21. and delay.

1

2

3

4

5

6 Two Decades of Internet Video Streaming:A Retrospective View *•* 1:15

7

1. The hybrid design is seemingly a compromise; yet we believe that the root cause is the intrinsic
2. heterogeneity of the Internet and its end users, which inevitably renders any one-fit-all solution to be
3. suboptimal. For peer-to-peer streaming, its high scalability and low cost for server cluster deployment
4. have made it feasible and promising. On the other hand, it faces significant challenges from a wide
5. spectrum of issues, including incentives, fairness, availability, and stability. Consider an extreme *flash*
6. *crowd* scenario of a popular concert broadcast that attracts one million users; if these users arrive
7. within the the first 100 seconds of the concert, the peak arrival rate will be 10,000 peers per second. If
8. the video quality is not good enough for the initial period, a user is more likely to leave the system. This
9. not only represents a failure of the system to provide service to this particular user, but also generates
10. a peer departure event, thus introducing more churn in the system [Sripanidkulchai et al. 2004]. This
11. problem can hardly be addressed by the peer-to-peer design paradigm alone, and external assistance
12. such as Content Distribution Networks or cloud datacenters is a must.
13. Whether the peer-to-peer design philosophy will thrive or survive hinges upon the crucial judg-
14. ment of whether its advantages outweigh its drawbacks, being so vulnerable to flash crowds and high
15. turnover rates. The potential copyright infringement as well as the lack of a clear business model for
16. both the content providers and the users are also critical concerns. Measurement studies have shown
17. that, in some peer-to-peer video streaming systems, a small set of peers are requested to contribute 10
18. to 35 times more uploading bandwidth than downloading bandwidth, while many others are *free-riders*
19. that contribute nearly zero. Such incentive mechanisms as *tit-for-tat* for peer-to-peer file download do
20. not seem to work well for real-time streaming data. With recent downward trends in storage and
21. bandwidth resource pricing, the momentum of the pendulum is swinging to the opposite side for peer-
22. to-peer. According to [Stoica 2010], bandwidth pricing of Content Distribution Networks was observed
23. to be dropping quickly every year, from 40 cents per GB in 2006 to less than 5 cents per GB in 2010.
24. As a result, the streaming cost per hour had decreased 15%–35%, to less than 3 cents per hour in
25. 2010. The consequence of these observations was dramatically reduced distribution costs for content
26. providers. For example, for paid content that is usually priced at $0.99 per episode, the distribution

34

1. cost is less than 3% of the provider’s total cost; for subscription-based premium content, it only costs
2. $1.60 per month to stream content to a user watching 2 hours per day! On the other hand, the emer-
3. gence of such ad-based business models have boosted the importance of *video quality*. There is a crucial
4. interplay between video quality and user engagement [Dobrian et al. 2011], and what contributes to
5. the majority of a content provider’s revenue is the income from ad-supported premium content. The
6. cost per thousand of ad impressions (CPM) for premium content has reached $20–$40, with a single ad
7. covering the cost of an entire hour of streaming. As a consequence, content providers, with an objective
8. of generating more revenue, start caring more about the visual quality and stability of their streaming
9. service to maximize user engagement. Half of the Internet video providers are offering or planning to
10. offer targeted or interactive advertising, and they continue to adopt social networking, growing to 41%
11. by last year [Infonetics 2011b].
12. By leasing storage and bandwidth resources to leading VoD providers such as Netflix, cloud com-
13. puting has emerged as the potential winner by purchasing resources at wholesale and selling them to
14. cloud users at retail. HTTP/DASH streaming powered by cloud computing and datacenters, coupled
15. with the social media and social effects due to the highly-available online social network services, have
16. recently enjoyed a meteoric rise in popularity, attracting an enthusiastic level of research attention
17. just like peer-to-peer systems did a decade ago.
18. Looking back over the past two decades, Internet video has greatly changed the landscape of content
19. distribution. It has largely swept out optical discs as the storage and distribution media in the movie
20. industry. It is currently re-shaping the TV broadcast industry with an ever-accelerating speed. Over
21. the next five years, SNL Kagan [SNL 2011] has predicted that the global IPTV subscriber base will

1

2

3

4

5

6 1:16 *•* B. Li *et al.*

7

1. grow at a 9.1% compounded annual growth rate, reaching 100.5 million homes by 2017, equivalent
2. to an 11.5% pay TV subscriber share. IPTV service revenues are expected to reach $41.2 billion by
3. 2017, accounting for 13.9% of global total pay TV revenues. Today, most IPTV service providers use
4. dedicated transport networks, providing quality access of digital TV programs from a head-end device
5. to end users’ dedicated set-top boxes (STBs). The Hybrid Broadcast Broadband TV (HbbTV) consortium
6. of industrial pioneers including SES, Humax, Philips, and etc. is currently promoting and establishing
7. an open standard for hybrid set-top boxes that receive broadcast and broadband digital television
8. and multimedia applications with a single-user interface. Our belief is that network convergence is
9. inevitable and the global Internet will be the vehicle for the further expansion of IPTV services. As
10. a matter of fact, in UK, BBC’s iPlayer has been successfully broadcasting high-quality TV programs
11. to both TV subscribers with STBs and public Internet users with Adobe Flashplayer since 2007; in
12. US, CNBC, Bloomberg Television and Showtime use live-streaming services from BitGravity’s CDN
13. to stream live television to paid subscribers using a standard http protocol. China, the largest IPTV
14. market by subscribers (12.6 million) to date, is probably the most vigorous market seeing an entire
15. range of technologies from peer-to-peer, to CDN, and to cloud computing, and these technologies are
16. competing with each other and with dedicated IPTV networks.
17. Our argument also follows the changing viewing habits. In 2006, an Accenture’s survey on IPTV
18. [Accenture 2006] showed that compelling TV content is the core foundation of any IPTV proposition,
19. which remains true today; yet it also indicated that consumers overwhelmingly preferred to watch
20. television on a television set, and the least favored devices on which to watch television are mobile
21. phones. The latest survey by Infonetics [Infonetics 2011a] however have shown that IPTV services
22. are getting highly personalized, integrated, portable, and on-demand. The results have confirmed an
23. important observation that is fairly obvious now: most service providers are moving beyond basic video
24. offerings toward richer user experiences. To date, 63% of IPTV service providers plan to support multi-
25. screen viewing across TVs, PCs, tablets, and smartphones [Infonetics 2011b]. Meanwhile, multi-view
26. and multi-streaming are being developed, in which multiple video streams from the same event are

34

1. delivered to a user, who will be able to switch between camera views [Arefin et al. 2012]. This is a real
2. recognition by service providers of what is happening in homes across the planet — that families are
3. voraciously and simultaneously consuming streamed high-definition video on devices other than the
4. traditional set-top box/TV couple.
5. The young adults and teenagers, otherwise known as the *Internet generation*, have long been hooked
6. to their desktops or laptops, interacting, socializing, and generating their own video content. The new
7. generation of smart mobile devices, mostly notably based on iOS, Android and Windows Phone plat-
8. forms, all of which emerged only in the recent five years, are driving the revolution further. The role
9. of the Internet itself has evolved from the original use as a communication tool to provide easier and
10. faster access to an infinite supply of information. Despite its rapidly expanding bandwidth, its narrow
11. waist of the TCP/IP protocol suite, which used to be the key to its success, has fundamental design
12. mismatches when providing real-time multimedia streaming services with a massive volume of data
13. and a massive audience. Research efforts in the past two decades have largely focused on accommodat-
14. ing and remedying these mismatches. While a discussion of the future Internet architecture is beyond
15. the scope of this paper, we have seen exciting future development towards data- or information-centric
16. networks [Ghodsi et al. 2011] that would turn to reality in the next two decades, with the hope of
17. making truly scalable, robust, personalized, and social video streaming to networked users a reality.

52

53

1. REFERENCES
2. ACCENTURE. 2006. International IPTV Consumer Readiness Study.

1

2

3

4

5

6 Two Decades of Internet Video Streaming:A Retrospective View *•* 1:17

7

1. ADHIKARI, V. K., GUO, Y., HAO, F., VARVELLO, M., HILT, V., STEINER, M., AND ZHANG, Z.-L. 2012. Unreeling Netflix:
2. Understanding and Improving Multi-CDN Movie Delivery. In *Proc. IEEE INFOCOM*.
3. AHLSWEDE, R., CAI, N., LI, S. R., AND YEUNG, R. W. 2000. Network Information Flow. *IEEE Trans. Inform. Theory 46,* 4,

11 1204–1216.

AKHSHABI, S., BEGEN, A. C., AND DOVROLIS, C. 2011. An Experimental Evaluation of Rate-Adaptation Algorithms in Adaptive

12

Streaming over HTTP. In *Proc. of ACM MMSys*.

1. ALBANESE, A., BLOMER, J., EDMONDS, J., LUBY, M., AND SUDAN, M. 1996. Priority Encoding Transmission. *IEEE Transac-*
2. *tions on Information Theory 42,* 6, 1737–1744.
3. AMIR, E., MCCANNE, S., AND ZHANG, H. 1995. An Application Level Video Gateway. In *Proc. of ACM Multimedia*.
4. AREFIN, A., HUANG, Z., NAHRSTEDT, K., AND AGARWAL, P. 2012. 4D TeleCast: Towards Large Scale Multi-Site and Multi-View
5. Dissemination Of 3DTI Contents. In *Proc. of IEEE ICDCS*.
6. AURRECOECHEA, C., CAMPBELL, A. T., AND HAUW, L. 1998. A Survey of QoS Architectures. *Multimedia systems 6,* 3, 138–151.
7. BALDINO, B., DUCKWORTH, M., ROMANOW, A., AND PEPPERELL, A. 2011. Framework for telepresence multi-streams. *IETF,*
8. *draft-ietf-clue-framework-10*.
9. BERGKVIST, A., BURNETT, D. C., JENNINGS, C., AND NARAYANAN, A. 2012. Webrtc 1.0: Real-time communication between
10. browsers. *Working draft, W3C*.

BOLOT, J.-C. AND TURLETTI, T. 1996. Adaptive Error Control for Packet Video in the Internet. In *Proc. of IEEE ICIP*.

1. BROXTON, T., INTERIAN, Y., VAVER, J., AND WATTENHOFER, M. 2010. Catching a Viral Video. In *Proc. of IEEE ICDM*.
2. BUDAK, C., AGRAWAL, D., AND ABBADI, A. E. 2011. Limiting the Spread of Misinformation in Social Networks. In *Proc. of*
3. *ACM* [*WWW*.](http://WWW/)
4. CAHILL, A. J. AND SREENAN, C. J. 2004. An Efficient CDN Placement Algorithm for the Delivery of High-Quality TV Content.
5. In *Proc. of ACM Multimedia*.
6. CARMEL, S., DABOOSH, T., REIFMAN, E., SHANI, N., ELIRAZ, Z., GINSBERG, D., AND AYAL, E. 2002. Network Media Stream-
7. ing. US Patent 6,389,473.
8. CASTRO, M., DRUSCHEL, P., KERMARREC, A.-M., NANDI, A., ROWSTRON, A., AND SINGH, A. 2003. SplitStream: High-
9. Bandwidth Multicast in Cooperative Environments. In *Proc. of ACM SOSP*.
10. CHENG, X. AND LIU, J. 2009. NetTube: Exploring Social Networks for Peer-To-Peer Short Video Sharing. In *Proc. of IEEE INFOCOM*.
11. CHENG, X. AND LIU, J. 2011. Load-Balanced Migration of Social Media to Content Clouds. In *Proc. of ACM NOSSDAV*.
12. CHENG, X., LIU, J., AND DALE, C. 2012. Understanding the characteristics of internet short video sharing: A youtube-based
13. measurement study.
14. CHU, Y.-H., RAO, S. G., AND ZHANG, H. 2000. A Case for End System Multicast. In *Proc. of ACM SIGMETRICS*.
15. CLEARY, K. 1995. Video on Demand-Competing Technologies and Services.
16. COCKCROFT, A. 2011. Netflix in the Cloud.
17. CRANLEY, N., PERRY, P., AND MURPHY, L. 2006. User Perception of Adapting Video Quality. *International Journal of Human-*
18. *Computer Studies 64,* 8, 637–647.
19. CUI, Y., LI, B., AND NAHRSTEDT, K. 2004. oStream: Asynchronous Streaming Multicast in Application-Layer Overlay Net-
20. works. *IEEE Journal on Selected Areas in Communications 22,* 1, 91–106.
21. DE CICCO, L., MASCOLO, S., AND PALMISANO, V. 2011. Feedback Control for Adaptive Live Video Streaming. In *Proc. of the ACM MMSys*.

44

DEERING, S. AND CHERITON, D. 1990. Multicast Routing in Datagram Internetworks and Extended LANs. *ACM Transaction*

1. *on Computer Systems 8,* 2, 85–110.
2. DIOT, C., LEVINE, B., LYLES, B., KASSEM, H., AND BALENSIEFEN, D. 2000. Deployment Issues for the IP Multicast Service
3. and Architecture. *IEEE Network 14,* 1, 78–88.
4. DOBRIAN, F., AWAN, A., JOSEPH, D., GANJAM, A., ZHAN, J., SEKAR, V., STOICA, I., AND ZHANG, H. 2011. Understanding the
5. Impact of Video Quality on User Engagement. In *Proc. of ACM SIGCOMM*.
6. EC2. [http://aws.amazon.com/ec2/.](http://aws.amazon.com/ec2/)
7. FENG, C., LI, B., AND LI, B. 2009. Understanding the Performance Gap between Pull-based Mesh Streaming Protocols and
8. Fundamental Limits. In *Proc. of IEEE INFOCOM*.
9. FIELDING, R., GETTYS, J., MOGUL, J., FRYSTYK, H., MASINTER, L., LEACH, P., AND BERNERS-LEE, T 1999. RFC 2616.

FLOYD, S., HANDLEY, M., PADHYE, J., AND WIDMER, J. 2000. *Equation-Based Congestion Control for Unicast Applications*.

54

Vol. 30. ACM.

55 FORGIE, J. 1979. ST-A Proposed Internet Stream Protocol.

1

2

3

4

5

6 1:18 *•* B. Li *et al.*

7

1. GHODSI, A., SHENKER, S., KOPONEN, T., SINGLA, A., RAGHAVAN, B., AND WILCOX, J. 2011. Information-Centric Networking:
2. Seeing the Forest for the Trees. In *Proc. of ACM Workshop on Hot Topics in Networks*.
3. GOUACHE, S., BICHOT, G., BSILA, A., AND HOWSON, C. 2011. Distributed & Adaptive HTTP Streaming. In *Proc. of IEEE*
4. *ICME*.

HAVEY, D., CHERTOV, R., AND ALMEROTH, K. 2012. Receiver Driven Rate Adaptation for Wireless Multimedia Applications.

12

In *Proc. of ACM MMSys*.

1. HO, T., KOETTER, R., MEDARD, M., KARGER, D., AND EFFROS, M. 2003. The Benefits of Coding over Routing in a Randomized
2. Setting. In *Proc. Int’l Symposium on Information Theory (ISIT)*.
3. HOSSEINI, M., AHMED, D., SHIRMOHAMMADI, S., AND GEORGANAS, N. 2007. A Survey of Application-Layer Multicast Proto-
4. cols. *IEEE Communications Surveys Tutorials 9,* 3, 58 –74.
5. HUA, K. A., CAI, Y., AND SHEU, S. 1998. Patching: a Multicast Technique for True Video-On-Demand Services. In *Proc. of ACM*
6. *International Conference on Multimedia*.
7. HUA, K. A. AND SHEU, S. 1997. Skyscraper Broadcasting: a New Broadcasting Scheme for Metropolitan Video-On-Demand
8. Systems. In *ACM SIGCOMM Computer Communication Review*. Vol. 27. ACM, 89–100.
9. 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
10. System. In *Proc. of ACM SIGCOMM*.

HUANG, Z., MEI, C., LI, L.-E., AND WOO, T. 2011. CloudStream: Delivering High-Quality Streaming Videos Through a Cloud-

23

Based SVC Proxy. In *Proc. of IEEE INFOCOM*.

1. INFONETICS. 2011a. [http://www.infonetics.com/pr/2011/lte-deployment-strategies-service-provider-survey-highlights.asp.](http://www.infonetics.com/pr/2011/lte-deployment-strategies-service-provider-survey-highlights.asp)
2. INFONETICS. 2011b. IPTV Services Getting Highly Personalized, Highly Integrated, Portable, On-Demand.
3. JACOBS, S. AND ELEFTHERIADIS, A. 1998. Streaming Video Using Dynamic Rate Shaping and TCP Congestion Control.
4. *Journal of Visual Communication and Image Representation 9,* 3, 211–222.
5. JANNOTTI, J., GIFFORD, D. K., JOHNSON, K. L., KAASHOEK, M. F., AND O’TOOLE, JR., J. W. Overcast: Reliable Multicasting
6. with an Overlay Network. In *Proc. of 4th Symposium on Operating System Design and Implementation (OSDI)*. Vol. 4.
7. JIANG, J., SEKAR, V., AND ZHANG, H. 2012. Improving Fairness, Efficiency, and Stability in Http-Based Adaptive Video Stream-
8. ing With Festive. In *Proc of ACM CoNEXT*.
9. KIM, T. AND AMMAR, M. H. 2001. A Comparison of Layering and Stream Replication Video Multicast Schemes. In *Proc. of*
10. *ACM NOSSDAV*.

KOSTIC´ , D., RODRIGUEZ, A., ALBRECHT, J., AND VAHDAT, A. 2003. Bullet: High Bandwidth Data Dissemination Using an

1. Overlay Mesh. In *Proc. of ACM SOSP*.
2. LI, B. AND NAHRSTEDT, K. 1999. A Control-Based Middleware Framework for Quality of Service Adaptations. *IEEE Journal*
3. *on Selected Areas in Communications 17,* 9, 1632–1650.
4. LIU, J. AND XU, J. 2004. Proxy Caching for Media Streaming over the Internet. *IEEE Communications Magazine 42,* 8, 88–94.
5. LIU, Z., WU, C., LI, B., AND ZHAO, S. 2010. UUSee: Large-Scale Operational On-Demand Streaming with Random Network
6. Coding. In *Proc. of IEEE INFOCOM*.
7. MAGHAREI, N., REJAIE, R., AND GUO, Y. 2007. Mesh or Multiple-Tree: a Comparative Study of Live P2P Streaming Ap-
8. proaaches. In *Proc. of IEEE INFOCOM*.
9. MCCANNE, S., JACOBSON, V., AND VETTERLI, M. 1996. Receiver-Driven Layered Multicast. In *ACM SIGCOMM Computer*
10. *Communication Review*. Vol. 26. ACM, 117–130.

MOK, R. K., LUO, X., CHAN, E. W., AND CHANG, R. K. 2012. QDASH: a QoE-Aware Dash System. In *Proc. of ACM MMSys*.

44

11–22.

1. (MPEG), I. J. S. W. . 2010. Dynamic adaptive streaming over HTTP.
2. NAHRSTEDT, K., YANG, Z., WU, W., AREFIN, A., AND RIVAS, R. 2011. *Session Management in 3D Tele-immersive Systems*. Hot
3. Topics in Multimedia Series, vol. 51. Springer-Verlag, Chapter International Journal of Multimedia Tools and Applications

48 (MTAP), 23–43.

1. NIU, D., XU, H., LI, B., AND ZHAO, S. 2012. Quality-Assured Cloud Bandwidth Auto-Scaling for Video-On-Demand Applica-
2. tions. In *Proc. of IEEE INFOCOM*.
3. OOI, W. T., VAN RENESSE, R., AND SMITH, B. 2000. The Design and Implementation of Programmable Media Gateways. In
4. *Proc. of ACM NOSSDAV*.
5. PAI, V., KUMAR, K., TAMILMANI, K., SAMBAMURTHY, V., AND MOHR, A. 2005. Chainsaw: Eliminating Trees from Overlay
6. Multicast. In *Proc. of 4th International Workshop on Peer-to-Peer Systems (IPTPS)*. 127–140.

PALLIS, G. AND VAKALI, A. 2006. Insight and Perspectives for Content Delivery Networks. *Communications of the ACM 49,* 1,

55

101–106.

1

2

3

4

5

6 Two Decades of Internet Video Streaming:A Retrospective View *•* 1:19

7

1. PARK, J.-S., GERLA, M., LUN, D. S., YI, Y., AND MEDARD, M. 2006. Codecast: a Network-Coding-Based Ad Hoc Multicast
2. Protocol. *Wireless Communications, IEEE 13,* 5, 76–81.
3. PENG, G. 2004. CDN: Content Distribution Network. *arXiv preprint cs/0411069*.
4. PODOLSKY, M., YANO, K., AND MCCANNE, S. 1999. A RTCP-Based Retransmission Protocol for Unicast RTP Streaming
5. 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

13

Engine(S) That Could: Scaling Online Social Networks. In *Proc. of ACM SIGCOMM*.

1. RODRIGUES, T., BENEVENUTO, F., CHA, M., GUMMADI, K.-P., AND ALMEIDA, V. 2011. On Word-Of-Mouth Based Discovery of
2. the Web. In *Proc. of ACM IMC*.
3. ROSENBERG, J., SCHULZRINNE, H., CAMARILLO, G., JOHNSTON, A., PETERSON, J., SPARKS, R., HANDLEY, M., SCHOOLER,
4. E., ET AL. 2002. RFC 3261. *SIP: Session Initiation Protocol*.
5. SAXENA, M., SHARAN, U., AND FAHMY, S. 2008. Analyzing Video Services in Web 2.0: a Global Perspective. In *Proc. of ACM*
6. *NOSSDAV*.
7. SCHULZRINNE, H., CASNER, S., FREDERICK, R., AND JACOBSON, V. 1996. RFC 1889. *RTP: A Transport Protocol for Real-Time*
8. *Applications*.
9. 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 *Proc. of IEEE INFOCOM*.

1. SHEU, S., HUA, K. A., AND TAVANAPONG, W. 1997. Chaining: a Generalized Batching Technique for Video-On-Demand Sys-
2. tems. In *Proc. of IEEE International Conference on Multimedia Computing and Systems*.
3. SILVERSTON, T. AND FOURMAUX, O. 2007. Measuring P2P IPTV Systems. In *Proc. of ACM NOSSDAV*.
4. SNL. 2011. Global Multichannel Markets Special Report: The State of Global IPTV.
5. SONG, H., MAHIMKAR, A., GE, Z., WANG, J., YATES, J., AND ZHANG, Y. 2011. Q-Score: Proactive Service Quality Assessment
6. in a Large IPTV System. In *Prof. of ACM IMC*.
7. SRIPANIDKULCHAI, K., GANJAM, A., MAGGS, B., AND ZHANG, H. 2004. The Feasibility of Supporting Large-Scale Live Stream-
8. ing Applications with Dynamic Application End-Points. In *Proc. of ACM SIGCOMM*.
9. STATISTICS, Y. Last accessed, March 24, 2013. [http://www.youtube.com/yt/press/statistics.html.](http://www.youtube.com/yt/press/statistics.html)
10. STOCKHAMMER, T. 2011. Dynamic Adaptive Streaming over HTTP: Standards and Design Principles. In *Proc. of ACM Confer-*
11. *ence on Multimedia Systems*.
12. STOICA, I. 2010. It’s Not the Cost, It’s the Quality! In *Proc. of 9th International Workshop on Peer-to-Peer Systems (IPTPS)*.

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

1. TRAN, D. A., HUA, K. A., AND DO, T. 2003. Zigzag: an Efficient Peer-To-Peer Scheme for Media Streaming. In *Proc. of IEEE*
2. *INFOCOM*.
3. VAKALI, A. AND PALLIS, G. 2003. Content Delivery Networks: Status and Trends. *IEEE Internet Computing 7,* 6, 68–74.
4. VENKATARAMAN, V., YOSHIDA, K., AND FRANCIS, P. 2006. Chunkyspread: Heterogeneous Unstructured Tree-Based Peer-to-
5. Peer Multicast. In *Proc. 5th International Workshop on Peer-to-Peer Systems (IPTPS)*. 2–11.
6. WANG, F., LIU, J., AND CHEN, M. 2012a. CALMS: Cloud-Assisted Live Media Streaming for Globalized Demands With
7. Time/Region Diversities. In *Proc. of IEEE INFOCOM*.
8. WANG, F., XIONG, Y., AND LIU, J. 2007. mTreebone: A Hybrid Tree/Mesh Overlay for Application-Layer Live Video Multicast.
9. In *Proc. of IEEE ICDCS*.
10. WANG, M. AND LI, B. 2007. *R*2: Random Push with Random Network Coding in Live Peer-to-Peer Streaming. *IEEE Journal*

.

*on Selected Areas in Communications*

1. WANG, Y., WENGER, S., WEN, J., AND KATSAGGELOS, A. K. 2000. Error Resilient Video Coding Techniques. *IEEE Signal*
2. *Processing Magazine 17,* 4, 61–82.
3. WANG, Y. AND ZHU, Q.-F. 1998. Error Control and Concealment for Video Communication: a Review. *Proceedings of the*

48 *IEEE 86,* 5, 974–997.

1. WANG, Z., SUN, L., CHEN, X., ZHU, W., LIU, J., CHEN, M., AND YANG, S. 2012b. Propagation-based Social-aware Replication
2. for Social Video Contents. In *Proc. of ACM Multimedia*.
3. WATSON, M. 2011. HTTP Adaptive Streaming in Practice. In *Proc. of ACM MMSys*.
4. WU, D., HOU, Y., ZHU, W., ZHANG, Y.-Q., AND PEHA, J. M. 2001. Streaming Video over the Internet: Approaches and Directions.
5. WU, Y., WU, C., LI, B., QIU, X., AND LAU, F.-C. 2011. CloudMedia: when Cloud on Demand Meets Video on Demand. In
6. *Proc. IEEE ICDCS*.

YU, F., ZHANG, Q., ZHU, W., AND ZHANG, Y.-Q. 2003. QoS-Adaptive Proxy Caching for Multimedia Streaming over the Internet.

55

56 *IEEE Transactions on Circuits and Systems for Video Technology 13,* 3, 257–269.

1

2

3

4

5

6 1:20 *•* B. Li *et al.*

7

1. ZHANG, L., DEERING, S., ESTRIN, D., SHENKER, S., AND ZAPPALA, D. 1993. RSVP: a New Resource ReSerVation Protocol.
2. *IEEE Network 7,* 5, 8–18.
3. ZHANG, M., LUO, J.-G., ZHAO, L., , AND YANG, S.-Q. 2005a. A Peer-to-Peer Network for Live Media Streaming - Using a
4. Push-Pull Approach. In *Proc. of ACM Multimedia*.

ZHANG, M., ZHANG, Q., SUN, L., AND YANG, S. 2007. Understanding the Power of Pull-Based Streaming Protocol: Can We Do

12

Better? *IEEE Journal on Selected Areas in Communications 25,* 9, 1678–1694.

1. ZHANG, Q., ZHU, W., AND ZHANG, Y.-Q. 2001. Resource Allocation for Multimedia Streaming over the Internet. *IEEE Trans-*
2. *actions on Multimedia 3,* 3, 339–355.
3. ZHANG, X., LIU, J., LI, B., AND YUM, Y.-S. 2005b. CoolStreaming/DONet: A Data-Driven Overlay Network for Peer-To-Peer
4. Live Media Streaming. In *Proc. of IEEE INFOCOM*.
5. ZHU, W., LUO, C., WANG, J., AND LI, S. 2011. Multimedia Cloud Computing. *IEEE Signal Processing Magazine 28,* 3, 59–69.

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55