Journal of Network and Computer Applications ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Review

A survey of schemes for Internet-based video delivery

Kevin J. Ma [a,b,*], Radim Bartoš [b], Swapnil Bhatia [c,b]

[a] Azuki Systems, Inc., 43 Nagog Park, Acton, MA 01720, United States
[b] Department of Computer Science, University of New Hampshire, Durham, NH 03824, United States
[c] Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, CA 94304, United States

ARTICLE INFO

Article history:
Received 18 September 2010
Received in revised form
21 December 2010
Accepted 4 February 2011

Keywords:
Video delivery

ABSTRACT

This survey looks at how traditional networking techniques (e.g., caching, traffic shaping, path diversity, and load balancing) have been adapted to address the needs of Internet-based video delivery. The stringent timing and relatively high bandwidth requirements of video traffic are taxing on best-effort networks and many video specific protocols and delivery methods have emerged over time in an attempt to mitigate network limitations. Video quality is directly tied to the underlying networks' ability to deliver data in time for playout. This paper surveys three classes of techniques which have been proposed for improving the quality of Internet delivered video: network load reduction, network interruption mitigation, and network load distribution. We discuss how each of these paradigms is applied within the different segments of the end-to-end video delivery system: by the server, in the network, or at the client, with a focus on how the underlying network conditions affect video quality optimization.

Contents

* Corresponding author at: Department of Computer Science, University of New Hampshire, Durham, NH 03824, United States. Tel.: +1 603 888 3276.
E-mail addresses: kevin.ma@azukisystems.com, kjma@cs.unh.edu (K.J. Ma), rbartos@cs.unh.edu (R. Bartoš), bhatia.swapnil@gmail.com (S. Bhatia).

1084-8045/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jnca.2011.02.001

1. Introduction

Over the past decade, the role of Internet-based video delivery has grown significantly. Consumers have embraced the convenience, flexibility, and variety available through Internet video, while content providers benefit from new vehicles for monetizing their existing content libraries. The high bandwidth requirements of video delivery, however, place significant strain on the network infrastructure. Though the proliferation of broadband access has improved bandwidth availability, network interruptions are still an issue for high quality video delivery.

The key to Internet-based video popularity is immediate, high quality, distortion-free, continuous playout. The primary metrics for gauging user experience are: initial playout latency (i.e., the amount of time between pressing the play button and seeing video rendered on the screen) and video artifacts (i.e., distortion, pixelation, or interruption in the rendered video). Minimizing wait times and minimizing distortions and stoppages is dependent upon the network's ability to deliver video data. Higher throughput allows faster initialization of the video buffer, while a minimum throughput, greater than the playout rate of the video, is necessary to prevent video artifacts from occurring.

Traditional network optimization techniques such as caching, traffic shaping, path diversity, and load balancing, have been used to address generic network scalability. Video traffic, however, differs from traditional Internet data traffic in that video is rendered at a specific aggregate rate. A minimum consistent throughput is required to prevent playout interruptions, but a delivery rate significantly greater than that minimum level does not improve the perceived quality of the rendered video. Traditional greedy network optimization techniques can therefore be further tailored for video delivery.

This survey covers optimization techniques for Internet-based video delivery systems. It focuses on application layer optimizations that improve end-to-end video delivery. The goal of each scheme is to address overall user perceived quality using standard metrics. Though many interesting challenges exist at the data-link, network, and transport layers, this survey concentrates on high level techniques which are affected by network conditions, but are not network specific. The rest of the document is organized as follows: Section 2 establishes background and terminology for the topics covered, Section 3 details the taxonomy used to classify the optimization schemes, Sections 4 and 5 discuss the schemes themselves, and Section 6 concludes by offering some thoughts on the future direction of video delivery research.

2. Network video delivery

Video sources are typically divided into two categories: *real-time* video and *non-real-time* video.

Real-time video is typically broadcast live and has more stringent timing requirements because there is no inherent buffering of the live data by the server. Video frames are generated and dispatched in real-time and clients receive and render the video frames in real-time. Any data processing for real-time video must be performed in real-time. These timing requirements typically prohibit high latency, low throughput, and CPU-intensive video transformation tasks.

Non-real-time video is typically pre-recorded and available via persistent storage, e.g., video on demand (VoD). The availability of a recorded file in non-real-time scenarios allows for pre-processing of video. Having access to the entire video also allows for digital video recorder (DVR)-like functionality (i.e., pause, fast forward, and rewind) as data may be easily retrieved from different offsets within the file.

There is also a class of *near-real-time* videos where live streams are partially buffered and redistributed from the buffered copy (Deshpande and Noh, 2008). This hybrid method is able to take advantage of many non-real-time video processing optimizations, with only minor time shifting of the real-time source. The primary difference between near-real-time and non-real-time is in the way the video is formatted. For near-real-time video, data must be stored in a format that does not require a complete set of metadata headers at the beginning of the file. HTTP Live Streaming (HLS) (Pantos, work in progress) is an example of near-real-time delivery, where video is recorded in segments and stored as individual transport stream formatted files.

2.1. Video delivery methods

The methods for delivering video are traditionally broken down into two categories: *streaming* and *download*. Streaming is typically associated with the real-time streaming protocol and real-time transport protocol (RTSP/RTP) (Schulzrinne et al., 1998, 2003), while download is typically associated with the hyper-text transfer protocol (HTTP) (Fielding et al., 1999). Download is further broken out into straight download and progressive download.

Streaming is usually characterized by having just-in-time delivery using unreliable transport and frame-based packetization. Just-in-time delivery uses less bandwidth and requires less client-side buffering than greedy, "as-fast-as-possible" approaches, while unreliable, frame-based delivery allows for graceful degradation. Graceful degradation favors degradation of video quality, by ignoring late or lost frames, over playout stoppages. Just-in-time delivery does not budget time for retransmissions, however, frame-based delivery limits the effects of single packet loss. With streaming, playout may begin as soon as the first frame is received, though a small buffer is typically employed for jitter avoidance.

Straight download is usually characterized by greedy delivery using reliable transport. Straight download typically is considered, for historical reasons, to not begin playout until the entire file has been downloaded. Reliable transport ensures zero frame loss, guaranteeing that the file is complete and of high possible quality. This quality guarantee, though, is at the expense of playout latency. Greedy download minimizes the impact of waiting for download completion, however, for large files, playout latency may still be quite high. Progressive download was introduced to deal with the high playout latency of straight download.

With progressive download, networking and playout are decoupled allowing rendering to begin before the entire file has been downloaded. This separation of functionality allows the network to more efficiently manage the rate of the download,

Please cite this article as: Ma KJ, et al. A survey of schemes for Internet-based video delivery. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.02.001

either from the client side or the server side. Progressive download is often thought of as using HTTP range GETs (i.e., requests with an HTTP "range" header that indicates a range of bytes to retrieve, rather than the entire file) to request file data in segments in a paced manner, whereas straight download is thought of as using non-range HTTP GETs to retrieve the entire file all at once. Progressive rendering enables many different progressive downloading schemes. Client-side bandwidth management may use HTTP range GETs on a single file, or may use multiple non-range HTTP GETs on smaller *segment files* (i.e., small independent files which were created by chopping up a larger file). Server-side bandwidth management may be done through protocol enhancements (MS-WMSP, 2010), or simply through direct media management (Ma et al., 2009).

While the terms straight download and progressive download are often used to describe both the delivery method as well as the playout characteristics, it is important to separate these duties. For the purposes of our discussion, we assume that progressive rendering is possible for both greedy and paced delivery. Most modern media players have progressive rendering capabilities. This includes both beginning playout before all data has been retrieved as well as discarding data which has been rendered to limit buffering requirements.

RTSP and RTP are the *de facto* standard protocols for streaming video. RTSP provides a TCP-based control channel which is used to negotiate and configure the UDP-based RTP data channels, as well as the UDP-based real-time transport control protocol (RTCP) feedback channels. RTP delivers actual audio and video frame data to the client, while RTCP is used to communicate packet loss and other information from the client back to the server and synchronization information from the server to the client. The packet loss information is used to estimate bandwidth for use in adapting the RTP data stream to the current network conditions (Fröjdh et al., 2006). Separate RTP and RTCP channels are used for audio and video data. A typical RTSP connection therefore requires one TCP connection and four additional UDP connections. From a networking perspective, RTSP imparts a great deal of overhead given the limited UDP port space per server, and the need to detect the dynamically negotiated UDP ports for firewall "fixups". However, the benefit is in the ability to perform graceful degradation.

RTP packets are composed of only one video or audio frame. While this is not the most bandwidth efficient method of packetizing data, it limits the impact of a single lost or late frame. For an uncompressed video where every frame is a key frame, the duration of a single packet loss is $1/F$ seconds, where F is the video frame rate. Most videos, however, are compressed, using differential encodings where the amount of data required to encode the video is dependent on how rapidly scenes change. Key frames provide full raster images when necessary, however, if only small portions of the image are changing, only differential updates need to be specified in subsequent frames. Though loss of a non-key frame may only affect a small portion of the viewing area, the resulting artifact may persist until the next key frame is received. Loss of a key frame, on the other hand, will result in a more significant distortion, and will also persist until the next key frame is received.

Progressive download can be used to achieve paced output, similar to streaming, but at a coarser granularity and using HTTP-based reliable transport. There is typically no graceful degradation associated with progressive download, only a potential for playout interruption. Though sender-side TCP buffer minimization schemes have been proposed to enable frame dropping at the source and provide rate adaptation through graceful degradation (Goel et al., 2008), in practice, this type of scheme is uncommon. Assuming the worst case, where a single packet loss negated an entire data burst of size s, the retransmission time r must not exceed the playout duration of the previous burst less the round-trip latency RTT required to detect the failure, i.e., $r \leq s/\rho - RTT$, where ρ is the playout rate of the video.

Though streaming and download schemes differ considerably, many of the video delivery optimization techniques surveyed below may still be applied to both. Streaming schemes, given their unreliable transport, are typically focused on preventing frame loss and/or lateness, or limiting the impact of frame loss and/or lateness. Preventing frame loss is useful for both streaming and progressive download approaches, as is minimizing frame lateness. Selective frame dropping schemes, however, were primarily created for streaming approaches, with no guaranteed delivery.

Deterministic quality guarantees are critical for commercial video delivery systems. This is evident from the popularity of progressive download in commercial applications, e.g., with Apple iPhone® HTTP Live Streaming (Pantos, work in progress), Microsoft SilverLight™ Smooth Streaming (Microsoft, 2009), and Adobe Flash® RTMP (Systems, 2009). Though the evaluation of many of the frame loss mitigation schemes discussed below do not have direct applicability to progressive download protocols, we believe that most of the scenarios could be adapted to evaluate TCP retransmission timing as well. Streaming protocol performance has long been a focus of research and commercial deployments given the historically challenging nature of delivery networks. The modern relevance of progressive download, however, warrants additional consideration for the burstiness and retransmissions associated with reliable protocols.

2.2. Video delivery metrics

The quality of video delivery is measured using two primary metrics: *playout latency* and *video artifacts* experienced. Playout latency is the amount of time between a user pressing the play button and the start of playout on the screen. Video artifacts are any image distortions or playout interruptions which occur during rendering.

2.2.1. Playout latency

Playout latency includes the network latency between the client and the server, as well as the buffering time for the client. Clients will generally buffer data to prevent underrun due to network jitter (Dey et al., 1997; Pao and Sun, 2001). The playout latency L is thus dependent upon the initial buffer size B_{init} and the video delivery rate R, i.e., $L = B_{init}/R + RTT$. The initial buffer size B_{init} is typically less than total physical buffer size B_{phys}. All media players employ a playout buffer, though the size of the initial buffer B_{init} varies. In a degenerate case the buffer size may be set to a single frame, but typically the buffer will contain on the order of seconds of data, to absorb network jitter. The buffer size is often dynamically chosen based on network conditions, as described below.

The video delivery rate R is typically dependent upon the video delivery method. If the video is streamed, then ideally the target delivery rate R_{stream} will be close to the playout rate of the video, though may be limited by the maximum network throughput τ, i.e., $R_{stream} = \min[\rho, \tau]$. If the video is delivered via straight download, then the actual delivery rate is limited only by the maximum network throughput τ, i.e., $R_{download} = \tau$. In the case of aggregate video delivery rate, progressive download should mimic the streaming behavior.

For a given video delivery rate, the buffer size will determine the playout latency. Minimizing the buffer size will obviously minimize the latency, however, the playout buffer plays an important role in

Please cite this article as: Ma KJ, et al. A survey of schemes for Internet-based video delivery. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.02.001

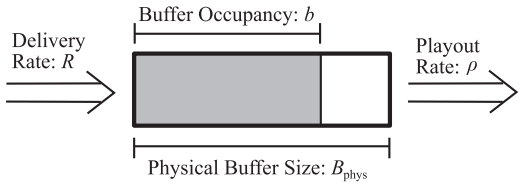

Fig. 1. Playout buffer.

preventing video artifacts by providing elasticity when faced with temporary network interruptions. Assuming the aggregate delivery rate exceeds the playout rate, i.e., $R > \rho$ over a long time scale, the maximum interruption duration must not exceed B_{phys}/ρ. Figure 1 depicts the data flow through a playout buffer.

The initial buffer size B_{init} is limited by either the physical memory size B_{phys} or the target buffer playout duration d_{target}. The duration d_{target} is typically measured in seconds and is used to calculate the buffer size as a function of the video playout rate, i.e., $B_{init} = \min[B_{phys}, d_{target} \cdot \rho]$. For a given set of network conditions, d_{target} may be optimized for the expected maximum interruption duration. As long as d_{target} is larger than the longest aggregate network interruption, the buffer should have the necessary elasticity to prevent playout underrun. This dynamic selection of buffer size B_{init} determines the playout latency L.

Using the opposite logic of the buffer size calculation, the remaining playout duration d_{remain} may be calculated given a fixed amount of buffered data b, i.e., $d_{remain} = b/\rho$.

There is an obvious trade-off between improving user experience through minimized playout latency and limiting robustness by reducing the playout buffer size. Selecting the proper buffer size, especially for resource constrained mobile devices, is a critical cost vs. performance consideration.

2.2.2. Video artifacts

Video artifacts are relatively easy to detect based on playout buffer underrun or, in the case of streaming, packet loss or packet lateness. Quantifying the impact of these failures is not as straightforward. Given the subjective nature of human tolerance to glitches in video rendering, a non-perception-based metric is required for evaluating the performance of different optimization techniques. The most common of these metrics is peak signal to noise ratio (PSNR).

For streaming methods, which employ graceful degradation schemes, packet loss or delay may manifest itself as pixelation or distortion in the rendered video. Comparing the difference in PSNR between two schemes provides a relative measure of quality. PSNR is a measure of distortion between two images. For video, PSNR is measured as an average over time. PSNR is an objective measure which provides a well defined, single unit metric for comparing different encoding schemes for a single piece of content. Absolute PSNR values for different pieces of content, however, may not be directly correlated (Huynh-Thu and Ghanbari, 2008). Also, given that PSNR is an average, it does not distinguish individual loss events over time. New metrics like the video quality metric (VQM) and the moving pictures quality metric (MPQM) have been proposed and shown to provide better correlation to subjective human perception metrics (Martinez et al., 2006), however, PSNR continues to be widely used (Wang, 2006). A full discussion of alternate quality metrics is beyond the scope of this survey, but we note the existence of alternatives for completeness. All of the papers surveyed below rely on PSNR for video artifact detection.

For download methods with reliable transport, the only artifact is playout stoppage. Packet loss or delay, or latency due to retransmission affect playout buffer underrun, however, the only perceivable impact is playout stoppage. The number of stoppages, duration of stoppages, and stoppage frequency may be used to further quantify quality.

3. Video delivery optimization schemes

The primary methods for addressing network resource limitations are:

- *Load reduction*: preventing network congestion by limiting the data transferred through different video encoding and rendering schemes.
- *Loss absorption*: preventing network loss or lateness and limiting the impact of network loss or lateness through different video encoding and rendering schemes.
- *Load distribution*: preventing network congestion by using path diversity from a single or from multiple sources.

Each of these methods may be implemented and applied by either the media player client, the media server, or in a third party server or network infrastructure. Table 1 shows a breakdown of the different network optimization techniques for video delivery. The following sections describe our methodology for classifying individual video delivery optimization techniques and describe the primary benefits of each technique.

3.1. Network optimization taxonomy

In Table 1, primary classification is made based on who is performing the optimization: client, server, or network. The delineation between client, server, and network is a natural one, and one which is heavily influenced by industry. Content providers typically control the servers which source the content, while separate Internet service providers (ISPs) control the network over which the content is delivered. Meanwhile, third-party hardware and software vendors develop client devices for use by everyday consumers to view the content. From both a financial and physical ownership point of view, each of these entities are uniquely separated.

The secondary classification performed in Table 1 involves understanding the effects of optimizations: modifying the data to be resistant to loss, limiting the data being transferred, distributing the data load across the network, or some combination of the three. What we have seen is that certain techniques are more easily implemented by servers (e.g., different media encodings), while other techniques work better when performed by clients (e.g., playout rate adaptation). In other cases, the network is the only entity with the necessary rights to apply a given technique (e.g., multicast). We also include consideration for content distribution networks (CDNs) which perform caching. Though the CDNs are not associated with ISPs, from a content provider perspective, they both have combined responsibility for network delivery.

Table 1
Network-based video delivery optimization schemes.

	Client-side	Server-side	Network
Load reduction	Adaptive Playout Rates	Adaptive Rate Encodings	IP Multicast
Loss absorption		Frame Reordering	Edge Caching
Load distribution	Peer-to-Peer	Multi-sender	

The resulting taxonomy, shown in Table 1, provides a concise list of video delivery optimization techniques, and their relationships to who provides that service, and what effect that service has on the network. This organizational structure allows us to see the many possible combinations of techniques, and helps us to better understand the relationships maintained within the video delivery ecosystem.

3.2. Network optimization techniques

The most direct approach to reducing load is to reduce the amount of data sent by the server. Given the strict timing requirements of video, indiscriminately reducing the amount of data may result in playout disruption.

Server-side schemes may employ different video encodings with different bit rates to reduce network load. Feedback from clients may be used to influence the choice of different encodings or to more directly control rate adaptation. In addition to providing feedback, the client can also adjust its playout rate, for the data it has already buffered (Girod et al., 2002). Section 4.1 discusses client playout rate adaptation schemes. While these schemes do not necessarily reduce network load, they do allow the client to tolerate short term reductions in data rate. Similarly, frame reordering schemes may be used to produce video files that are less susceptible to packet loss. These types of schemes are based on information gleaned from the network and optimized for specific network conditions. Frame reordering schemes are discussed along with other video encoding options in Section 4.2.

Another method for reducing network load is to use multicast. Multicast, in this context, implies network multicast (i.e., IP multicast), as opposed to application level multicast (e.g., peer-to-peer multicast overlay networks). While IP multicast is an important technology for improving network scalability, it is not a video specific application layer optimization and is omitted from this survey.

Load distribution is a localized form of load reduction. The goal of load distribution is not to necessarily reduce the aggregate load on the network, but rather to reduce load on individual links. Load distribution schemes tend to be the most recognizable category, as these schemes address architectural issues with delivery networks. Load reduction and loss absorption schemes tend to be focused more on the video data, and less on the network itself.

Caching schemes, discussed in Section 5.1, rely on caching data at the edge of a network, closer to the clients. This reduces load on the core of the network, while also reducing load on the video *origin servers*. The origin server is a content provider controlled server which hosts the original content being distributed. Proxy caching can also be used to provide hierarchical buffer management to mask core network latency. Partial caching and staging can be used to add *elasticity* to the video stream, where the buffering of low latency data from caches can offset the longer latency of non-cached data in an alternating fashion.

Peer-to-peer (P2P) schemes, discussed in Section 5.2, have gained considerable popularity in recent years (Liu et al., 2008). P2P schemes push the burden of sending data onto the clients; older clients are transformed into servers for newer clients. By intelligently assigning the client–server relationship between peers, load may be distributed and localized. Like caching schemes, P2P schemes also reduce load on the video origin servers (Apostolopoulos et al., 2005).

Multi-sender schemes, discussed in Section 5.3, use multiple physically dispersed servers to send data to clients (Apostolopoulos and Trott, 2004). Servers are chosen so that the paths from servers to the clients are as disjoint as possible. Path diversity can also provide redundancy benefits, though at the cost of increased aggregate network load.

Given the distributed nature of video delivery systems, no single method or scheme can solve the video delivery optimization problem. Client-side, server-side and network resident schemes must work in concert with each other to provide a holistic solution. Hybrid schemes are often the target of proposed research. Though these hybrid relationships are not explicitly depicted in Table 1, this survey covers hybrid schemes in Section 5.4.

4. Load reduction schemes

In this section we cover two main network load reduction schemes: *adaptive playout rate* schemes and *video encoding* schemes. With adaptive playout rate schemes, client players support rendering video content at a rate other than the intended video bit rate. With video encoding schemes, servers are modified to support encoding and delivering video content at multiple bit rates.

4.1. Adaptive playout rates

Video clients typically buffer some portion of the video file to minimize the impact of network instability. Assuming that sufficient network bandwidth exists, a typical client will wait until the buffer is full before beginning playout. If a network interruption occurs, the client will continue to play from the buffer. If the network interruption is shorter than the buffer duration d, underruns are averted. There are two limitations to the client buffering paradigm:

- Filling the buffer increases playout latency by L seconds.
- Underrun protection is limited by the buffer size B.

Because L is directly proportional to B, any attempts to increase underrun protection by increasing B will negatively impact L. Adaptive playout rates may be used to address both limitations simultaneously, without the need for trade-offs. For playout latency, the video playout rate may be temporarily reduced to reduce the buffering requirements, while for underrun protection, the video playout rate may be temporarily reduced to lengthen the protection interval (Girod et al., 2002) or to increase the buffer refill rate (Kalman et al., 2004). Though there is a limit to how much the playout rate can be reduced before the quality of the video becomes too low, this technique can be used to augment the physical capabilities of the client player.

We first make the assumption that the playout rate of the video is less than the video delivery rate: $\rho < \tau$, i.e., the network is not the limiting factor. We also make the assumption that the desired playout buffer size is less than the maximum physical buffer size: $b < B_{phys}$, i.e., the buffer hardware is not the limiting factor. Assuming a reduced playout rate $\rho' < \rho$, playout latency and buffer duration may then be evaluated.

4.1.1. Reducing playout latency

We know that buffer size is directly proportional to video playout rate. Substituting our reduced playout rate ρ' into the buffer size calculation, we can see that: $B' = d_{target} \cdot \rho'$. We also know that playout latency is directly proportional to buffer size. Substituting our reduced playout rate buffer size B' into the latency calculation, we can see that $L' = B'/R + RTT$. Therefore, given the directly proportional relationships in these equations, we can see that reducing the playout rate also reduces the playout latency, i.e., $\rho' < \rho \Rightarrow B' < B \Rightarrow L' < L$.

Please cite this article as: Ma KJ, et al. A survey of schemes for Internet-based video delivery. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.02.001

4.1.2. Increasing underrun protection

We know that the playout duration d of a fixed amount of buffered data b is inversely proportional to the video playout rate ρ. Substituting our reduced playout rate ρ' into the playout duration calculation, we can see that: $d'_{remain} = b/\rho'$. Given this is an inverse relationship, we can see that reducing the playout rate increases the perceived buffer duration, i.e., $\rho' < \rho \Rightarrow d'_{remain} > d_{remain}$.

4.1.3. Playout rate reduction

Videos are typically encoded at a constant frame rate, e.g., 24 or 30 frames per second (fps). Similarly, audio is encoded at a constant sample rate, e.g., 22.05, 44.1, or 48 kHz. Playout rate reduction can be implemented by feeding the fixed rate data at a slower rate than intended to stretch out the wall clock duration of the rendered video. For streaming video, each video frame or group of audio samples is typically associated with a timestamp relative to the start of the video. Playout rate reduction can be achieved by modifying these timestamps with a cumulative offset.

Li et al. have focused on client-side playout rate adaptation for wireless networks (Li et al., 2004, 2005, 2008a). They examine playout rate reduction when combined with power consumption (Li et al., 2004). Mobile clients typically rely on wireless communications and battery life is of great importance; reduced power consumption for wireless radios is highly desirable. However, limiting the power used for wireless communications can negatively impact the quality of the wireless channel and cause network interruptions. Li et al. propose a quality measurement scheme whereby a cost model based on a constant video bit rate and fixed packet size is used. A playout slowdown cost captures user perception while a separate playout variation cost captures the variation in playout rate over time. The trade-off in quality and power consumption is measured to evaluate the ability of playout rate adaptation to compensate for deficiencies in the wireless channel.

Li et al. (2004) evaluate high definition video delivered over wireless LANs using reliable transport. They use a probabilistic model of packet loss to estimate future power consumption, based on the video packets waiting to be transmitted. They then reduce the video delivery rate to lower power consumption. Packet loss due to interference is inevitable in wireless links, however, the link layer protocols are designed to retransmit when collisions occur. Li et al. seek to minimize collisions and retransmissions by varying the network delivery rate. In this case, the network quality is intentionally degraded by reducing wireless signal strength to save power.

Li et al. (2005, 2008a) also include content awareness in their playout rate adaptation schemes. Given the differential compression schemes commonly used for video, low motion scenes require fewer bits to encode. Changes in rate during low motion scenes are also less perceptible to viewers. Li et al. propose a metric for measuring *motion intensity*, and expand their previous playout slowdown and playout variation cost models to take into account motion intensity using a sliding window rate history. They define a scheme for taking advantage of low motion scenes (as defined by the motion intensity) to more optimally reduce playout rate using a preset progression to prevent abrupt changes.

Li et al. also investigate discarding packets which are late and cannot be salvaged even by slowing the client playout rate (Li et al., 2008a; Kalman et al., 2004). This scheme removes the power component of the model and presumably the reliable transport to allow for packet discard. Though they acknowledge the transparency of losses over the wireless link (given retransmissions), they assume that excessive packet loss will result in late frames. They adapt the network delivery rate in a range of $R = \rho \pm 33\%$. Rates as low as 74 kbps mimic challenging 2G or poor quality 3G cellular environments, though are less applicable to WiFi. And though frame dropping is more stream oriented, the concept of motion intensity and the principles of playout rate reduction is still applicable to download schemes.

Kalman et al. (2004) have also done work with adaptive media playout (AMP). Their scheme addresses both playout latency and network interruption. They focus on a network condition where the video delivery rate on average is just greater than the video bit rate, i.e., $\rho \approx \tau$. Given some assumed jitter, network underruns are expected to occur. The playout rate is adapted in three scenarios:

1. *AMP-Initial*: The playout rate is reduced at the initial startup. Playout is started prior to full buffer occupancy and the reduced playout rate allows for an increased buffer fill rate to help reach full buffer occupancy.
2. *AMP-Robust*: The playout rate is reduced by a fixed amount when buffer occupancy falls below a certain threshold. This allows for an increased buffer fill rate when buffer occupancy gets too low.
3. *AMP-Live*: The playout rate is reduced by a fixed amount when buffer occupancy falls below a certain threshold or increased by a fixed amount when buffer occupancy rises above a certain threshold. This allows for adjusting buffer fill and drain rates to maintain a target buffer fullness.

The scheme proposed by Kalman et al. allows for both a playout rate which is lower and a playout rate which is higher than the actual video bit rate, in the AMP-Live case. In cases where network latency causes a burst of data to occur, if the buffer cannot handle the accumulation of latency, the playout rate must be increased to catch up to the packets that are still being streamed in. It also prevents severe distortions in the total video duration.

Though they give no specific working environment examples, Kalman et al. assume "error-prone channels". They assume a 1% probability of packet loss, where packet losses always result in bursts. The burst loss durations were exponentially distributed between 0 and 1.5 s, for AMP-Live, and between 0 and 2 s for AMP-Robust and AMP-Initial. In addition, like Li et al, they assume a barely sufficient network delivery rate.

Hui and Lee (2006) have used a similar scheme, but have included a multi-sender dimension in their efforts. Argyriou (2006) also proposed a scheme similar to the AMP-Live scheme proposed by Kalman et al. Like Kalman et al., they use a single target buffer fullness threshold and allows for both a playout rate which is lower and a playout rate which is higher than the actual video bit rate. The distinguishing feature in this scheme is the predictive rate adaptation. The scheme predicts the expected future video delivery rate based on previous frame latencies. As with Li et al. (2004), they assume a wireless connection and a reliable transport protocol. A separate loss probability for handoff between wireless access points is also introduced. The network throughput is assumed to be fixed at about twice the video playout rate, i.e., $\tau \approx 2 \cdot \rho$, and retransmission latencies are simulated for lost packets. In this case, the relative network capacity $\tau/\rho \approx 2$ is more realistic than the cases where $\tau/\rho \approx 1$.

4.2. Video encoding schemes

Video is typically encoded at a single bit rate. In cases where network bandwidth is expected to be an issue, multiple encodings may be created, e.g., high, medium, and low bit rate encodings, as

Please cite this article as: Ma KJ, et al. A survey of schemes for Internet-based video delivery. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.02.001

shown in Fig. 2(a). Often, however, these independent encodings cannot be easily interchanged by the player during playout due to differences in resolution or frame rate. Assuming the necessary encoding parameters match, there is still an issue of synchronizing frame offsets within each file. Given the differential encoding and predictive compression schemes used by video codecs, switching should only be done on key frame boundaries to prevent video artifacts from occurring. Implementing rate adaptation requires additional metadata mapping the key frames and byte offsets for each of the encoded files. The different encodings are generated by the server and the rate adaptation is also implemented on the server during streaming delivery. An example of this is the packet-switched streaming service (PSS) (Fröjdh et al., 2006) used with RTSP. The RTCP feedback channel is used by the server to monitor client bandwidth and initiate rate changes when client bandwidth changes.

An alternative to synchronizing file offsets for different encoded files is to use file segments, as shown in Fig. 2(b). The video is still transcoded into multiple bit rates, but each individual bit rate is written into multiple independently playable files. When played in succession, the segments combine to form the complete video. In this case, the granularity at which rate adaptation may occur is lower, as switching will only occur on segment boundaries. The different encodings are generated by the server, however, the rate adaptation is handled by the client using progressive download. An example of this is the HTTP Live Streaming (Pantos, work in progress) protocol used by the Apple iPhone®. The client monitors its own download rate and requests segments whose playout rate does not exceed the current download rate.

The alternative to using multiple independent encodings is to use a layered encoding. Layered encodings come in two forms (Kim and Ammar, 2005): *cumulative* and *non-cumulative*, as shown in Fig. 2(c) and (d), respectively. Both cumulative and non-cumulative schemes are suitable for use with either streaming or download.

Cumulative encodings rely on a base encoding to which additional layered encodings are applied. Each additional layer improves upon the base encoding, but the additional layers are useless without the base encoding. The MPEG2, MPEG4, and H263 are popular examples of codecs which support cumulative encoding. Non-cumulative encodings create multiple independently decodable encodings. When combined, the multiple non-cumulative encodings provided better quality than any single encoding. Multiple description coding (MDC) (Goyal, 2001) is the primary example for non-cumulative encoding. Non-cumulative encodings require more data than cumulative encodings to encode a video of equal quality, as each description contains redundant information necessary to support independent decoding. Though non-cumulative encodings provide some additional error protection through its redundancy, the resource constrained environments for video delivery often prefer the lower bandwidth requirement of cumulative encodings (Kim and Ammar, 2005). Non-cumulative encodings, however, are simpler to use with multi-sender schemes which seek path diversity for load distribution. MDC will be discussed further in Section 5.

Another method for improving the robustness of a video encoding is to rearrange the frames of a non-layered single description encoding to reduce correlation and add resilience

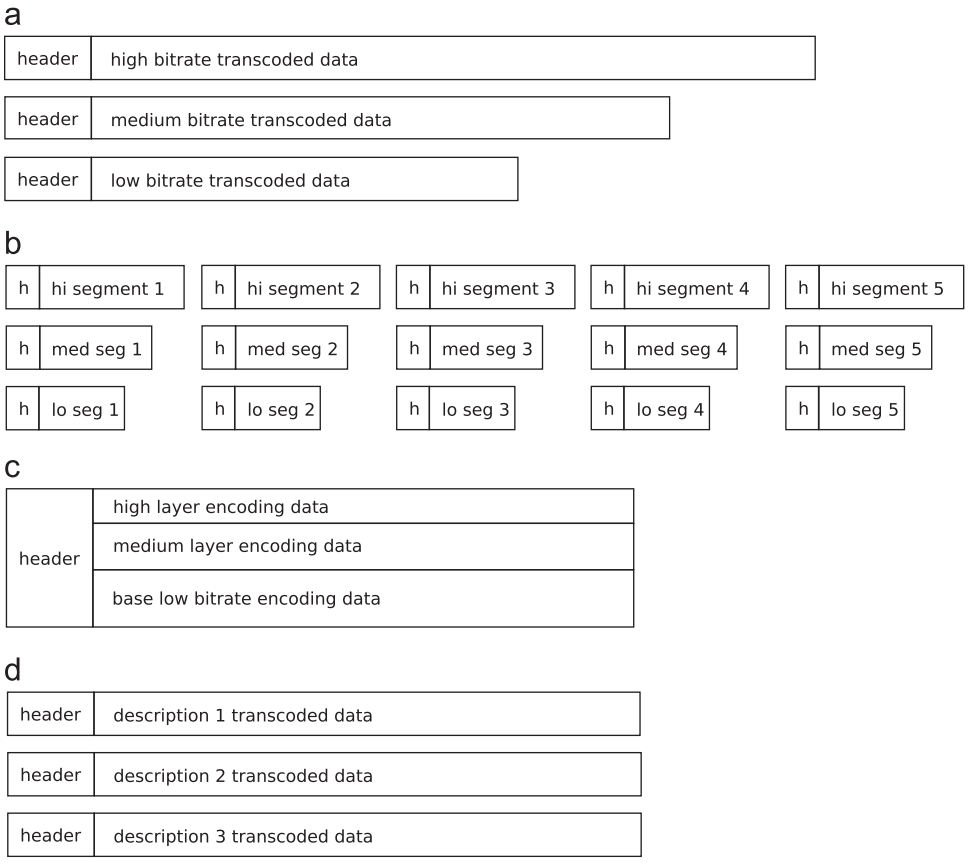

Fig. 2. Multiple bit rate encoding schemes: (a) independent transcoded files, (b) independent transcoded and segmented files, (c) cumulative layered encoding, and (d) non-cumulative multiple description encodings.

Please cite this article as: Ma KJ, et al. A survey of schemes for Internet-based video delivery. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.02.001

to burst losses. Video is processed and stored sequentially. The resulting temporal order is logical from an encoding and decoding perspective, however, because of the compression schemes used in modern video codecs, not all frames are of equal size or importance. Though the reordering of frames does not reduce the bandwidth required to deliver the video, nor does it improve the quality of the video (assuming the data is received), it can increase the probability that only non-key frames are lost when network interruptions occur. For streaming delivery with graceful degradation, key frame loss results in higher distortion than non-key frame losses, so reducing key frame loss can increase the quality of rendered video in lossy networks.

4.2.1. Encoded frame reordering

In this section, we focus our discussion on frame reordering schemes for single encoding videos. Wee et al. (2002) propose an analytical model for measuring the distortion for a group of pictures (GOP), where the GOP includes a keyframe and all related non-key frames. The distortion is predicted using the probability that any frame within the GOP will arrive late. They propose optimizing the frame ordering to minimize overall distortion and investigate the trade-off between advancing frames to help future playout and the disruption to current playout caused by advanced frames. Sending a frame sooner increases the chance that it will arrive on time, however, it decreases the chance that other frames will arrive on time. Figure 3 shows how advancing a frame from position f to position $f-k$ causes the k frames in between to be shifted outward and increases their possibility of being delayed. Their scheme can be performed offline with the frame reordered versions of the video stored for use in VoD.

Liang and Girod (2006) also looked at the interdependencies of frames in predictive encodings but proposed a different solution which relies on live encoding. They consider the scenario of live video where real-time adaptation is needed. Their approach relies on detecting frame loss through a client feedback mechanism. When a key frame loss is detected, future frames are encoded to a different key frame which was not lost and is likely to still be in the playout buffer. They propose a scheme for choosing the optimal alternate reference key frame using a distortion cost function. The scheme relies on a low latency feedback channel, as well as storage and maintenance of the distortion prediction tables. They built upon their original work in Liang et al. (2003) where they investigate the effects of burst losses in addition to single packet loss.

Liang et al. (2008) have also investigated non-linear increases in distortion caused by burst losses, compared with single packet losses. Intuitively, this non-linearity is understandable given the higher probability of key frame loss in a burst loss scenario. They propose the use of a packet interleaving scheme for networks that exhibit large burst losses. Blocks of frames are interleaved, as depicted in Fig. 4, to reduce the probability of correlated loss. The limitation of this scheme is that all frames associated with a block must be received before the block can be rendered. Given an interleaving of n blocks of m frames each, the interleaving introduces a delay of $n \cdot (m-1)+1$, for each block.

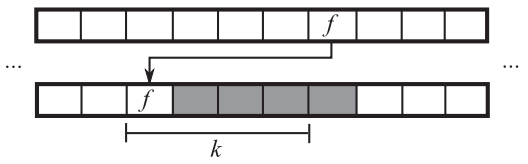

Fig. 3. Frame reordering can be used to minimize packet loss distortion, though the arrival of the k shifted frames will be delayed by a frame.

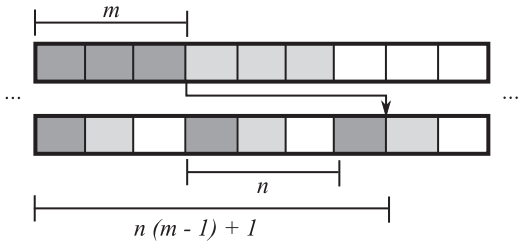

Fig. 4. Frame interleaving can be used to minimize burst loss distortion, though block arrival will be delayed by $n \cdot (m-1)+1$ frames.

5. Load distribution schemes

Access networks and the network core have traditionally been a significant bandwidth bottleneck. Local area network (LAN) bandwidth has typically exceeded access network bandwidth by three orders of magnitude. While core network bandwidth has often exceeded LAN bandwidths, the load on core networks is many orders of magnitude larger than that on a typical LAN.

In this section we cover three network load distribution schemes: *caching*, *P2P*, and *multiple sender* schemes. Caching schemes distribute data to the network edge so that data does not need to cross the network core reducing congestion in the core. P2P schemes push network load away from centralized data centers toward localized communities at the network edge to not only limit congestion in the core, but to also limit requests serviced by the origin data center. Multiple sender schemes distribute load over multiple diverse paths so that no single core network path is overloaded. The schemes described below represent different approaches to solving the common problems of improving core network utilization and offloading origin servers.

5.1. Caching schemes

Large scale video distribution is typically handled by large centralized data centers. Data centers have high bandwidth Internet connections with dedicated resources and service level agreements (SLAs) to guarantee availability. Data centers, while providing high quality network access, are expensive and consequently few in number. Because of their relative sparsity, it is unlikely that a data center will be either physically or temporally proximate to any given client. As such, a typical connection from client to data center must cross the network core. When the core network is congested, interruptions or delays in the data stream may occur. One standard solution for avoiding core network congestion is to use edge caches. Caches are placed at the network edge to minimize the need for core network traversal. Data caches are typically geographically dispersed, as depicted in Fig. 5, with the assumption that any client that is far away from the origin data center will be able to connect to a cache that is closer.

Caching schemes are one of the key features employed by CDNs. While distributing caching hardware may be cheaper than building data centers, it is still expensive and requires management resources. CDNs provide an alternative to content providers whereby content providers may lease a portion of the CDN's large caching and distribution infrastructure, rather than build their own. This makes CDN based cache optimizations attractive to commercial content providers. The following section focuses on video delivery enhancements using network caching.

5.1.1. Prefix caching

Caching schemes have been proposed to address playout latency whereby caching data close to the client reduces the impact of wide

Please cite this article as: Ma KJ, et al. A survey of schemes for Internet-based video delivery. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.02.001

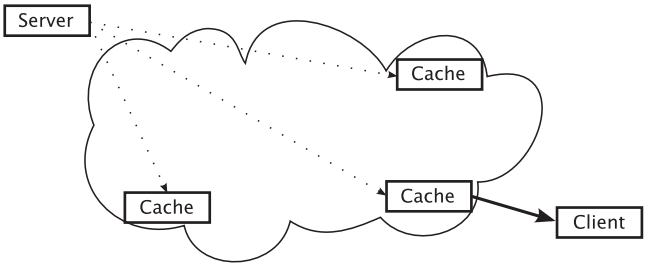

Fig. 5. Cached video delivery: the server replicates the video to edge caches; the closest edge cache sends the video to client.

area network (WAN) traversal latency. Sen et al. (1999) propose the concept of *prefix caching* where only the initial portion of the video, the *prefix*, is cached. This limits the storage requirement since video files can be large, while still reducing the initial playout latency since the initial portion is immediately available. The playout duration of the prefix is used to mask the retrieval latency of the rest of the uncached video. A separate staging buffer is used as a sliding window for storing the remainder of the file. Sen et al. also propose a rate shaping scheme for delivery of the data and discuss the trade-offs between allocating cache resources to the prefix buffer and the staging buffer.

Shen et al. have built upon this idea of prefix caching, by extending the definition of prefixes. They consider videos as being naturally segmented (e.g., chapterized or annotated video), with each segment being a possible starting point. Chapterization of content is typically performed by the content provider and marks logical video starting points, however, user generated annotations (either implicitly based on access requests, or explicitly based on user comments) are also a good indicator of high probability starting points. Seeing the value in having semi-random access to different portions of the video, they propose a fixed segmentation of the video with caching of each segment's prefix (Shen et al., 2007). Their cache eviction algorithm uses segment popularity for ranking segments for eviction. It may be the case that only the middle or end portion of a video is interesting or relevant, therefore, there is no reason to store the beginning of the video, only the beginnings of the popular sections.

Li et al. (2008b) extended the popularity-based caching scheme with variable sized segments and presented an algorithm for coalescing adjacent segments. Segment coalescence reduces the number of segments, which reduces the number of prefixes that must be cached. This frees space for other videos, however, it also reduces the starting point flexibility. A generalized cost function is provided to manage the trade-off between decreased user satisfaction and increased storage space, all weighted by the popularity of the segment. Tu et al. (2009) expand upon the cost function of Li et al. by more accurately modeling user satisfaction on a per-GOP basis, rather than just a per-segment basis. They also introduce two-level logical cache partitioning based on popularity duration. The level 2 (L2) cache uses their cost function for segment evaluation and prefix determination, however, the cost analyses is a non-real-time function that is performed on a daily or weekly basis. The level 1 (L1) cache stores any retrieved segments that are not in the L2 cache (i.e., less popular segments) as they are requested by the clients. L1 eviction relies on a least recently used (LRU) scheme. If the L1 content proves to be more popular over time, it can be designated to replace existing L2 content.

5.1.2. Segment caching

Moving beyond prefix caching, more generic caching schemes focus on continuous delivery of data and prevention of video artifacts. Ma and Du (2002) propose a scheme where half of the data is cached in the proxy and half remains at the origin server based on alternating chunks, i.e., the video is divided into N chunks $0,\ldots,N-1$, with even numbered chunks residing at the proxy and odd numbered chunks residing at the origin server. This scheme requires the client to request data from both the proxy and the origin server in parallel, however, the WAN latency for each odd chunk is offset by the playout duration of the alternately cached even chunk. While this could be considered a multiple sender scheme, the authors propose the proxy cache as being a staging server. They suggest that having multiple connections synchronized at the client can be made mostly transparent by implementing it in the video player.

Chen et al. (2004, 2006, 2007) provide a more dynamic approach to segment-based caching through cache prefetching and proxy jitter avoidance techniques. The obvious advantage of prefetching is the avoidance of cache miss latency. Chen et al. refer to cache miss latency as *proxy jitter*. Aggressive prefetching can lead to inefficient storage utilization and excess bandwidth consumption due to unnecessary prefetch requests. The trade-off between proxy jitter and wasted prefetches is typically measured in byte hit ratio. Using the playout rate of the video and the available proxy bandwidth, Chen et al. provide an analytical model for calculating the prefetch deadlines required to avoid proxy jitter and the minimum caching required to guarantee artifact free playout. For caches with limited storage, however, minimum caching for all videos may not be feasible. In such cases, the byte hit ratio trade-off is complicated by the fact that aggressive cache prefetching also requires aggressive cache eviction. Eviction of prefix segments may adversely affect playout latency for future requests, whereas over dedication of resources to prefixes may result in playout disruptions if the cache is unable to buffer enough data to service all of the clients. Their scheme attempts to balance these trade-offs to increase the scalability of the proxy caches while maintaining quality.

Chen et al. (2004, 2006) propose a Hyper proxy system which combines segmentation, intelligent eviction, and intelligent prefetching based on video popularity. Initially, the system fully caches each video. When an eviction occurs, fully cached videos are selected first for partial eviction. The fully cached videos are segmented based on their current popularity and segments are evicted based on their expected future popularity. Chen et al. use previous request frequency and previous average request duration to predict future request probability and duration. Readmission into the cache is based on similar calculations with a goal of minimizing over-caching and minimizing buffer underrun, as determined by a dynamic duration threshold. The final component of their scheme is active prefetching. Prefetching is initiated based on the prefetch deadlines that were calculated for proxy jitter avoidance. Their SProxy implementation provides a more complete solution with real world deployment statistics (Chen et al., 2007).

Others have also proposed caching optimizations in conjunction with server-side encoding schemes. Kao and Lee (2007) propose a cache replacement algorithm for use in a proxy which does transcoding. The transcoding proxy generates different bitrate encodings of the same video and takes into account the byte size and popularity of each encoding when caching (e.g., videos more often access from slower networks cache lower bitrate encodings). Li and Ong (2007) looked at prefix caching, but extended their discussion to include multi-layer video encodings. Chattopadhyay et al. (2007) propose a scheme for caching their own multi-layered encoding which is optimized for progressive download.

The majority of video caching schemes are targeted at reducing playout latency by minimizing the distance that the video has to travel to the client. In general, these schemes do not make absolute assumptions about network latency or loss, only that

Please cite this article as: Ma KJ, et al. A survey of schemes for Internet-based video delivery. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.02.001

proximity caching has relative performance advantages over WAN-based transactions. While fully caching videos would also have the effect of eliminating the packet loss associated with WAN congestion, fully caching large video files is an expensive proposition. Optimizing caching schemes to limit storage requirements, while still maintaining the reduced playout latency and improved user experience, would be of great advantage to CDNs and large corporate enterprise networks.

5.2. Peer-to-peer (P2P) schemes

In non-P2P networks, nodes are exclusively partitioned into clients and servers, with servers distributing content and clients consuming content. In P2P networks, however, the line between clients and servers is blurred to distribute the burden of the servers onto the clients. The clients act as servers for other clients so that those later clients do not have to go directly to the origin server.

P2P networks are typically organized as application overlay networks. Figure 6 depicts a basic tree-based P2P network where the origin server and client are connected through a network of *peer* nodes. Though other configurations are also used, trees (i.e., acyclic directed graphs) are popular due to their simplicity and the ability to distribute unidirectional video traffic evenly over nodes and links (Jurca et al., 2007). Mesh network overlays are also sometimes used for their flexibility, however, they require additional routing complexity. In many cases, P2P networks will employ multiple overlays to provide overlay path redundancy, better leaf node utilization, and path diversity (Liu et al., 2008). Multiple overlays, however, incur higher network construction costs, require more bandwidth per peer to connect to more peers, and increase management complexity.

In P2P schemes, peer routing is done at the application layer, and must still rely on IP networking for their underlying connectivity. Direct peers are considered one hop away in the overlay network, though the peer may be many hops away in the underlying physical network. In order for P2P networks to function effectively, peers should be temporally and physically proximate to minimize network latency between peers. Peers must also be mindful of their upstream and downstream bandwidth, as well as their processing, storage, and memory resources. Unlike the uniform server characteristics and bandwidth availability in most professional data centers, peer capabilities may vary widely. P2P networks may also exhibit high rates of change, as peers may join or leave the network at any time. The overlays must be dynamically adapted to support continuous connectivity in the event of peer churn. Fast reconvergence of overlay network connections between peers is critical for P2P reliability. This is especially critical for video streaming, which has stringent delay requirements.

P2P schemes, in general, are much more complex than centralized server or distributed caching schemes. They are also very susceptible to churn-based denial of service attacks. Their advantage, however, is in their theoretical ability to fairly distribute server and network load. While much effort has been devoted to the study of generic P2P networks, the unique characteristics of video traffic allow for additional enhancements to P2P-based delivery. Though overlay convergence is important, the schemes themselves are not video specific. The primary video specific contributions are found in P2P routing and packet scheduling, to minimize jitter in delivery.

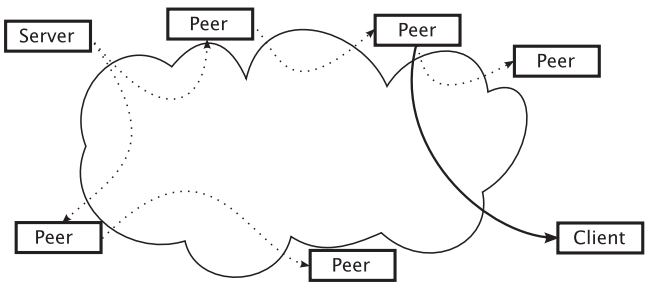

Fig. 6. Peer-to-peer video delivery: peers replicate the video to their neighbors until the video makes it to the end client.

5.2.1. P2P packet scheduling

In this section, we compare a number of P2P scheduling schemes which have been proposed (Setton et al., 2006; Baccichet et al., 2007; Li et al., 2007). We first, however, define a common nomenclature for concepts common to all the schemes. We define the packet importance function $I(n,m)$ as the importance of the nth packet on the mth overlay network. For single overlay networks, the simplified $I(n)$ notation is used. The packet importance is used for prioritization of packets, in each of the schemes. When network congestion occurs, the packet importance is used to select packets to be discarded. Lower priority packets are discarded when packet delivery starts falling behind and the transmit queues begin to overflow. Also common to many of the schemes is a distortion factor $D(n)$, which measures the impact to the video of losing the nth packet. While different methods are used to assess distortion, the specifics of video signal processing are beyond the scope of this document; our primary concern is how these content-aware concepts are applied to network packet scheduling. Other scheme specific functions are introduced below, but are generalized to facilitate the comparison of schemes.

Setton et al. (2006) propose combining a P2P architecture with packet scheduling that prioritizes packets based on the impact of losing a given frame and the number of clients impacted by the given stream. They propose calculating the packet importance $I(n,m)$ based on the distortion $D(n)$, weighted by the number of clients $N(m)$ downstream from the given peer:

$$I(n,m) = D(n) * N(m)$$

In this scheme $D(n)$ is proposed simply as the number of frames affected by the loss of a given packet. Specifically, $D(n)$ in this case, gives higher weight to key frames, as the loss of a key frame will affect the proper decoding of subsequent non-key frames. Simulations were performed to compare the quality of a video streamed with no frame prioritization versus one using the proposed $I(n,m)$. In their simulations, they assume core network links have sufficient bandwidth and only vary the local access links of the peers. They do not introduce any artificial packet loss; they only simulate the dropping of packets due to congestion and peer churn. Their peer upstream and downstream link bandwidths are chosen from a realistic set of home networking alternatives, between 512 kbps and 20 Mbps for downstream and between 256 kbps and 5 Mbps for upstream. They then vary the acceptable playout latency and measure the video quality.

Baccichet et al. propose a scheme similar to Setton et al., using the same distortion function $D(n)$, however, they omit the consideration for number of downstream nodes impacted by the loss (Baccichet et al., 2007) as they are using multicast rather than unicast delivery to downstream peers:

$$I(n) = D(n)$$

They also include retransmission in their scheme, to complement frame prioritization. They perform their tests using a live system, rather relying on simulations, but still comparing the video quality with and without frame prioritization and retransmission. They also introduce peer churn and measure the average playout latency experienced and the number of playout interruptions experienced by peers.

Please cite this article as: Ma KJ, et al. A survey of schemes for Internet-based video delivery. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.02.001

Li et al. (2007) also propose a scheme similar to Setton et al. using a distortion factor $D(n)$, weighted by the number of clients $N(m)$ downstream from the given peer, but optimized for packet latency. Their distortion function $D(n)$ is just a general weighting function for the nth packet; they provide no specific definition for distortion. They also introduce a term $T(n)$ to account for the processing time required to send the nth packet:

$$I(n,m) = D(n) * N(m)/T(n)$$

In the degenerate case of all packets having equal processing cost, this scheme is identical to the scheme of Setton et al. In this scheme, larger packets with slightly higher importance could be passed over for smaller packets with slightly less importance. This scheme is not video specific; the authors generalize their scheme to any real-time data stream. They simulate a static network configuration with no peer churn and randomly vary network bandwidth and measure the average packet latency. Impact of latency on video quality is not addressed.

Chakareski and Frossard (2006a) propose a rate-distortion optimized packet scheduling scheme. They do not specifically address P2P systems, but rather define a more generalized routing problem for prioritizing packets at a single node. They do not take into account the number of clients affected, but do use a distortion function $D(n)$ and the packet size. Since there is typically a direct correlation between packet size and processing time, we substitute the term $T(n)$ for the purposes of comparison. They also include in their calculation the probability of loss $P(n)$, given different physical layer media:

$$I(n) = P(n) * D(n)/T(n)$$

Though this scheme is not P2P specific, we mention it here given its similarity to the other P2P scheduling schemes proposed.

Li et al. (2009) have also proposed a non-P2P specific frame dropping scheme which uses a distortion function $D(n)$ based on preventing key frame loss. They rely on cooperation between "links" and "users" to calculate a global solution to prevent network congestion for a given set of client/server pairs. Frame dropping is performed by the server, rather than by the network. In this way, it is very similar to the P2P overlay network frame scheduling problem for a mesh overlay configuration.

The commercial efficacy of P2P schemes for video delivery is not clear at this time, however, the research into video packet routing sparked by the nascent popularity of P2P is valuable for non-P2P schemes as well. It is difficult to relate the issues of peer churn with the Internet errors, but generic enhancements video stream resilience and aggregate underrun prevention may be widely applicable. Though these schemes focus mainly on packet discard for streaming approaches, they could be expanded to investigate retransmission timing for progressive download schemes as well.

5.3. Multiple sender schemes

Most traditional video distribution schemes are either one-to-one, or one-to-many. One-to-many schemes (e.g., broadcast television) require a specialized multicast infrastructure for proper implementation. The Internet today typically relies on one-to-one connections. While less efficient than one-to-many communications, one-to-one communications are simpler to implement given the lack of support for multicast in the current Internet. Unicast also provides more flexibility for media personalization (e.g., start times can be chosen at will for VoD, whereas broadcast television requires you to watch what is currently on). A third option also exists, as depicted in Fig. 7, namely many-to-one communications.

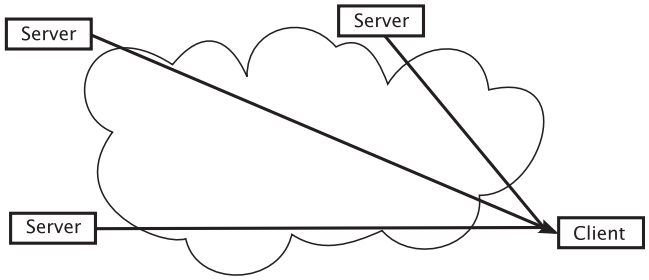

Fig. 7. Multi-sender video delivery: multiple servers send different encodings of the video to the client.

While having multiple senders will typically result in increased aggregate bandwidth due to network encapsulation overhead and information redundancy, using multiple senders can have a significant impact on reducing load for specific individual network links. Highly loaded links may benefit from the load distribution offered by multi-sender schemes. Additionally, clients may benefit from redundant data being sent through disjoint paths, when non-cumulative multiple encodings (e.g., MDC, as outlined in Section 4) are used. Even in the case where path diversity is not achieved, Tullimas et al. (2008) suggest that multiple connections to a single server may be used to provide for congestion avoidance. They propose the initiation of additional TCP connections to combat exponential back-off and slow start, while using window size management to limit throughput. This degenerate case of multiple TCP connections to a single server could be expanded to multiple senders.

Multi-sender schemes depend on client modifications to not only select and connect to multiple servers, but also to be able to reconstruct data which has been spread across multiple servers. For basic data files, there are numerous trivial data reconstruction schemes, e.g., simple interleaving where the first χ bytes are retrieved from one source and the next χ bytes are retrieved from a different source. The previously discussed caching scheme by Ma and Du (2002) is an example of this. In general, segment-based progressive download schemes, like HTTP Live Streaming (Pantos (work in progress), are well suited for retrieving from multiple senders. The granularity of the segments can vary between a single frame of data and many seconds of data. The retrieval can also be for an arbitrary number of bytes assuming the source files are identical. For these rudimentary schemes, data may be retrieved in parallel but it must still be rendered sequentially. With layered encoding or multiple description coding schemes, however, data is not interleaved, but rather, multiple layers or descriptions are sent in parallel and may be combined to achieve higher quality rendered video. While the multi-sender paradigm is often combined with other load distribution schemes (i.e., caching and P2P), this section focuses on video delivery enhancements using just multi-sender networks.

5.3.1. Multiple sender bandwidth distribution

Chakareski and Girod (2003) propose a model for tracking packet performance from multiple servers and using that information to predict future performance for each server. The scheme then calculates cost functions for all possible server allocation policies, choosing the policy that minimizes errors. Chakareski and Frossard (2006b, 2008) follow up with a scheme that combines multiple senders, bandwidth estimation at each sender, and multiple encodings of the same video at different bit rates. Each sender is able to know every other senders' bandwidth and, without communicating, knows who will send which packets at any given time, based on that knowledge. Bandwidth estimates are piggybacked on acknowledgement messages from the client,

Please cite this article as: Ma KJ, et al. A survey of schemes for Internet-based video delivery. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.02.001

however, the client must send the acknowledgement message to every sender in order for each sender to know every other senders' bandwidth. The impact to the client of sending these acknowledgement messages is not addressed. The estimates are averaged over a specified trailing time interval. While they acknowledge that the complexity of global optimizations make this scheme impractical, they suggest that a priori packet classifications and presumably a priori agreed upon transmission algorithms provide a lower complexity alternative.

Chakareski and Girod (2003) model a burst loss scenario with a 3% initial loss probability and an 85% continued burst loss probability. They also more systematically consider the effects of different levels of bandwidth and different levels of independent loss (Chakareski and Frossard, 2006b, 2008). Different levels of bandwidth are simulated in 20 kbps increments, up to the point where sufficient bandwidth exists for error free playout (80 kbps for no loss and 200 kbps under loss); packet loss was simulated at 0%, 5%, and 10%. They also measured actual Internet network delay and loss, however, the results are not specified. A reference is provided to a 2001 study (Markopoulou et al., 2003) that showed aggregate loss rates of less than 0.3%, though with high tendency for burst losses (as is common in the Internet). The use of higher resolution, higher bit rate, higher frame rate video, with modeling for burst losses would provide for an interesting comparison with the existing results.

Nguyen and Zakhor (2002, 2004) propose a slightly different approach, where the clients have more control over server allocation, which they refer to as receiver driven packet scheduling. Similar to Chakareski and Frossard, clients send control messages containing the estimated sending rate and delay of every sender to every sender. These additional control channels are required for unreliable streaming protocols, however, TCP-based approaches may rely on implicit knowledge based on the TCP window and acknowledgements received (Wang et al., 2009). The messages, however, are only sent when the receiver wishes to change the server load distribution. They estimated bandwidth using RTT and loss estimates for a given path Nguyen and Zakhor (2002). They also propose a more complex algorithm which takes into account combined loss across all paths Nguyen and Zakhor (2004). In both cases congestion and loss are assumed to be isolated to a single path. The client uses these estimates to rank the paths and change the distribution of load accordingly, in an attempt to minimize packet loss, packet duplication, and out of order packets, using its predicted loss and delay for each server. Each server uses an a priori agreed upon packet scheduling algorithm to maintain coherency and prevent duplicate transmissions or transmit omissions.

Nguyen and Zakhor provide both simulated and empirical results. Their simulations assume 800 kbps aggregate throughput, with burst losses contributing an average packet loss of 2%. In their live Internet tests, no artificial loss was introduced and their results showed an average packet loss of less than 0.1%.

5.4. Hybrid schemes

The benefits of each of the schemes described above are not mutually exclusive. In many cases, the advantages of caching, P2P, and multi-sender schemes have been combined to form new hybrid schemes. In this section, we cover research relating to hybrid approaches to video load distribution.

5.4.1. Caching + P2P

Kusmierek et al. (2006) have looked at combining P2P concepts with caching in their Loopback scheme. Rather than having a self forming P2P network, the Loopback scheme configures clients into a chain, based on the temporal locality of their requests. A caching architecture is employed as a first level for video distribution. A proxy cache at the edge of the caching infrastructure acts as the root of a P2P network. The proxy cache acts as a centralized agent for creating the P2P networks, but the complexity of managing overlays is removed by only creating single descendent chains. Figure 8 depicts a P2P network using single descendent loopback chains.

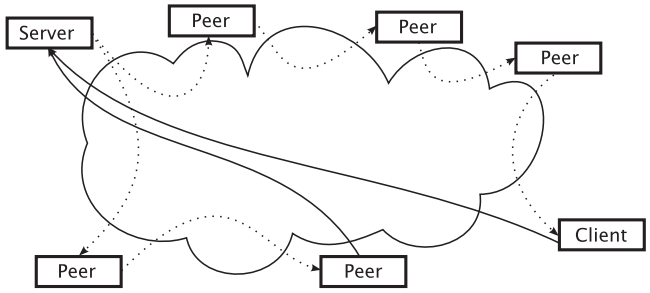

Fig. 8. Loopback cache: peers are organized into chains, which loopback to the server.

Fig. 9. Loopback cache: the chain buffers an extra $s_{N-1}-s_0+\delta$ seconds of data.

The start time for each member of the chain is offset by the number of clients ahead of it in the chain. Each new client added to a chain initially buffers the last δ seconds of video data that they viewed. If another client makes a request within δ seconds of the previous request, that client is added to the end of the chain and receives data from the previous client. The previous client limits its buffering at that point, as described below.

Given a chain of N clients, with start times $S=\{s_0,\ldots,s_{N-1}\}$, Loopback ensures that: $s_{i+1}-s_i \le \delta$, for $0 \le i < N-1$. The last client in the chain always buffers δ seconds worth of data; the other clients in the chain buffer $s_{i+1}-s_i$ seconds worth of data (i.e., they buffer only enough to service the next client in the chain). In the case of a failure at the ith node, a patch of $s_{i+1}-s_i$ seconds of data is sent to s_{i-1}, so that s_{i-1} can service s_{i+1}. The buffer size limits any clients buffering requirement to δ seconds of data and also allows chain length to be managed based on popularity (i.e., more popular videos will have higher request frequency and need shorter δ s while less popular videos with lower request frequencies may need longer δ s to prevent large numbers of single entry chains).

Given that the clients limit their buffering to: $s_{i+1}-s_i$ seconds, the clients do not have to manage buffering beyond the servicing of the next peer in the chain. Kusmierek et al. also consider this client buffering as cached data that the proxy cache would normally have to buffer, but no longer does. The cache only needs to keep the data required to service the first client in the chain. The last client in the chain returns its data back to the cache, hence the name *Loopback*. In this way, the data is available for the next request that comes along, which by definition, must have occurred more than δ seconds in the future, otherwise the new client would have just been added to the existing chain. The loopback chain caches $s_{N-1}-s_0+\delta$ seconds worth of data for the proxy, as depicted in Fig. 9. This scheme also has the obvious P2P benefit of reducing network load on the proxy cache. Ignoring physical network latency between adjacent peers in the chain, there is no additional playout latency injected by the scheme. If all clients are hitting the same cache, it is assumed that they must

Please cite this article as: Ma KJ, et al. A survey of schemes for Internet-based video delivery. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.02.001

all share some type of locality, and thus P2P latency should be minimized.

For Kusmierek et al., the goal was to address the scalability of the proxy cache by limiting the number of direct requests. They do not make any assumptions about absolute bandwidth, nor do they assume infinite bandwidth. They instead choose a more scalable methodology of comparing schemes using relative bandwidth measures. They assume that the server bandwidth is always limited to 30% of the total bandwidth required to service all requests. They then measure the bandwidth load reduction and excess storage availability afforded to the server by their scheme.

Kalapriya and Nandy (2005) propose a similar approach of using peers for extended caching, but do not consider the simplification of using a single chain of peers. Consequently, their unrestricted tree architecture requires greater focus on error recovery. They model error probability in a node as directly proportional to the relative caching load of the node within the system. Presumably, the higher the load, the more likely the node is overloaded. Their results focus on the number of streams served while varying cache sizes. They model *flash crowds* using request rates of 9, 15, and 30 request per minute. Though these rates would not be considered high in large scale data center environments, in consumer P2P environments it is possible that 30 requests could saturate the network connection. They measure server scalability in terms of number of streams. They do not address server scalability in terms of network or CPU utilization.

5.4.2. P2P + multiple senders

One of the key issues with having multiple senders is the selection of which senders to use. De Mauro et al. (2006) propose a receiver driven scheme using pings to try to determine network path correlations. The path correlations are used to select peer routes for P2P overlay networks. Once paths have been established to each server, different MDC encodings are distributed from each server to the client, as depicted in Fig. 10. De Mauro et al. pose the intuitive argument that path diversity is desirable, as disjoint paths minimize the probability that packets from multiple descriptions will be lost. They go on to propose that through a series of pings, they can estimate path correlation and select near optimal pairs of paths, where optimality is based on least number of shared links. For each path, they measure the difference in RTT for some number of consecutive pings and calculate a statistical correlation. Using a loss probability and a "network area" calculation based on RTTs (as proposed by Guo et al., 2003), they predict the servers with the least shared paths.

Lu and Lyu (2006) propose another receiver driven scheme where clients request data from their peers. They enforce a constraint of having only a single outstanding request to each peer. In this way, they implicitly measure and adapt to network load, though pipelining advantages are lost. The most capable peers will respond and become available for another request soonest. Separately they prioritize between streams based on the time left until an underrun occurs.

De Mauro et al. provide simulation results for their scheme, having modeled realistic wireless bandwidths and latency. Lu and Lyu performed their tests on a live system, also with realistic parameters.

5.4.3. Caching + multiple senders

In a different pairing of CDNs and caching, Apostolopoulos et al. have proposed a scheme which takes advantage of existing CDN caching and distribution infrastructures. CDNs typically provide many edge caches and an automated infrastructure for replicating files to each cache. Apostolopoulos et al. propose using

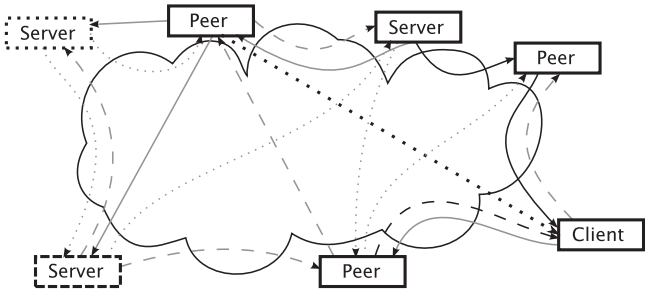

Fig. 10. Peer-to-peer multi-sender video delivery: multiple servers send different encodings, over different paths, through the P2P network, to the client.

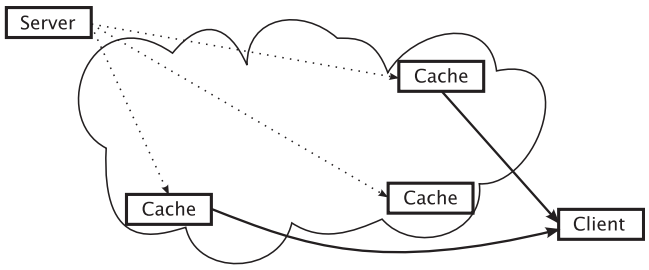

Fig. 11. Cached multi-sender video delivery: multiple caches send different encodings, over different paths, to the client.

multiple caches to distribute MDC coded videos. They have modeled and examined the performance gains of using multiple description (MD) versus single description (SD) coding. They address different network scenarios (i.e., varying degrees of path diversity and magnitudes of packet loss, including burst losses). Specifically, they focus on video with two descriptions (Apostolopoulos et al., 2002a, 2002b), as depicted in Fig. 11. Their research focuses on addressing two questions:

- How many descriptions should be replicated to each cache?
- How should clients select *surrogates* (caches)?

Apostolopoulos et al. (2002c) provide analysis of three replication schemes, and three surrogate selection schemes. The replication schemes use different random distributions:

- *SD*: random selection of half the servers, placing an SD encoded video at each of those servers.
- *MD-half*: uses the same random servers as SD, placing both MD video encodings at each of the servers.
- *MD-all*: randomly assigns one of the MD video encodings to each of the servers.

The SD algorithm provides the control for comparisons. The MD-half and MD-all algorithms were chosen specifically to enforce the restriction that they have approximately the same storage and bandwidth footprint as the SD algorithm.

The surrogate selection schemes use hop count as a measure of distance and progressively increase in complexity:

- *Shortest path*: select the closest server for each MD encoding.
- *Heuristic*: select the shortest paths with weighting for least shared links.
- *Distortion*: select the servers predicted to have least distortion, based on probability of packet loss.

The shortest path algorithm provides the control for comparisons. It is a direct adaptation of existing DNS proximity based

Please cite this article as: Ma KJ, et al. A survey of schemes for Internet-based video delivery. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.02.001

server selection used in CDNs today. The heuristic algorithm takes the average path length for groups of servers and adds a penalty for the number of shared links in the path. It then selects servers based on the minimum adjusted path length. The distortion algorithm uses a Gilbert (1960) model to predict packet loss, and selects servers based on a minimum distortion calculation.

Apostolopoulos et al. model an average packet loss of 5% based on studies performed in 1997 (Yajnik et al., 1999). It is not clear if the 5% loss is still applicable to the modern Internet as network capacity and throughput has advanced considerably since then. As mentioned previously, Nguyen and Zakhor (2004) observed sub-0.1% loss rates in their work. Also, the trans-Atlantic test scenarios would be highly unlikely in a CDN-based delivery scheme. Cache proximity would not typically favor inter-continental link traversal.

6. Conclusion

The network infrastructure plays a crucial role in the delivery of video over the Internet. Video is currently one of the most demanding applications in terms of bandwidth, timing, and buffering requirements. Failure to provide appropriate quality of service is very noticeable to users. To address these issues, many classic networking schemes have been adapted and applied to video delivery. This survey organizes a variety of recent video delivery optimization schemes and evaluates the underlying network assumptions made by these schemes.

Our classification methodology takes a network-centric view that focuses on where optimizations are applied (i.e., the client, the server, or the network elements themselves) and how optimizations affect network traffic (i.e., load reduction, loss absorption, or load distribution). Understanding the role of each component in this ecosystem, client, server, and network, allows us to optimize the entire video delivery path in a more holistic way. Looking at it from each component's point of view, we can then identify different optimization techniques which may be combined during different stages of video delivery. As video delivery solutions become more complex, our taxonomy is useful in recognizing these interactions in the hybrid methods presented above.

We would like to highlight two key observations from the schemes surveyed: most analyses focus on extreme network conditions and have very limited consideration to progressive download schemes. Network reliability has continually improved over time, and more investigation into the characteristics of modern data traffic is needed. The non-linear effects of scaling packet loss and throughput may limit the applicability of high loss, high jitter simulations on real-world low loss, low jitter networks. Improvements to network throughput and latency have also made reliable communications more practical for real-time applications. TCP-based delivery has become popular for guaranteeing high quality video delivery, and further research into retransmission avoidance and mitigation for reliable video delivery is needed.

We believe that the future of video delivery research needs to consider the two primary delivery methods: streaming and segmented delivery. For streaming approaches, error recovery and error prevention continue to be the focus. Network coding and forward error correction (FEC) for limiting quality degradation are areas for further study. For segmented delivery approaches, the coordination of multiple sources needs to be further investigated, both within a single CDN and across multiple CDNs. The need for further characterization of both rate adaptation and reliable multicast distribution schemes applies to both streaming and segmented delivery. For all of these areas, a firm understanding and justification of the network conditions is critical.

High quality video delivery is critical for ensuring user satisfaction, protecting content provider brand value, and providing a vehicle for advertising sponsors. Video delivery schemes which rely on unreliable transport are unable to provide the quality guarantees required by the video delivery ecosystem. Content providers use high quality video to attract users who in turn provide the financial incentive for content providers, by either paying for content or attracting sponsors.

Looking at the modern landscape of Internet-based video delivery, we find the major media players (i.e., Adobe Flash Player®, Microsoft Windows Media Player®, and Apple QuickTime®) using progressive download schemes by default. The standard RTSP players (i.e., RealNetworks Real Player® and Apple QuickTime®) will also attempt to tunnel RTP through TCP connections since firewall blocking of RTP is common.

As video data becomes an ever larger portion of network traffic and Internet delivered video becomes an ever larger part of modern culture, the importance of high quality user experience becomes even more important. Moving forward, quality guarantees, rather than quality degradation, will be the focus of both users and should be the focus for researchers as well. The techniques described herein provide a firm basis for network video delivery research, and hopefully offers inspiration for enhancing existing video delivery protocols and infrastructure.

References

Apostolopoulos JG, Trott MD. Path diversity for enhanced media streaming. IEEE Communications Magazine 2004:80–7.

Apostolopoulos J, Tan W, Wee S, Wornell GW. Modeling path diversity for multiple description video communication. In: Proceedings of the 2002 IEEE international conference on acoustics speech and signal processing (ICASSP 2002); 2002a.

Apostolopoulos JG, Tan W, Wee SJ. Performance of a multiple description streaming media content delivery network. In: Proceedings of the 2002 IEEE international conference on image processing (ICIP 2002); 2002b. p. II-189–III-192.

Apostolopoulos JG, Tan W, Wee SJ. On multiple description streaming with content delivery networks. In: Proceedings of the 2002 IEEE International Conference on Computer Communications (InfoCom 2002); 2002c. p. 1736–45.

Apostolopoulos J, Trott M, Kalker T, Tan W. Enterprise streaming: different challenges from internet streaming. In: Proceedings of the 2005 IEEE international conference on multimedia & expo (ICME 2005); 2005. p. 1386–91.

Argyriou A. Improving the performance of TCP wireless video streaming with a novel playback adaptation algorithm. In: Proceedings of the 2006 IEEE international conference on multimedia & expo (ICME 2006); 2006. p. 1169–72.

Baccichet P, Noh J, Setton E, Girod B. Content-aware P2P video streaming with low latency. In: Proceedings of the 2007 IEEE international conference on multimedia & expo (ICME 2007); 2007. p. 400–3.

Chakareski J, Frossard P. Rate-distortion optimized distributed packet scheduling of multiple video streams over shared communication resources. IEEE Transactions on Multimedia 2006a;8:207–18.

Chakareski J, Frossard P. Distributed streaming via packet partitioning. In: Proceedings of the 2006 IEEE international conference on multimedia & expo (ICME 2006); 2006b. p. 1529–32.

Chakareski J, Frossard P. Distributed collaboration for enhanced sender-driven video streaming. IEEE Transactions on Multimedia 2008;10:858–70.

Chakareski J, Girod B. Server diversity in rate-distorted optimized media streaming. In: Proceedings of the 2003 IEEE international conference on image processing (ICIP 2003); 2003. p. 645–8.

Chattopadhyay S, Ramaswamy L, Bhandarkar SM. A framework for encoding and caching of video for quality adaptive progressive download. In: Proceedings of the 2007 ACM international conference on multimedia; 2007. p. 775–8.

Chen S, Shen B, Wee S, Zhang X. Designs of high quality streaming proxy systems. In: Proceedings of the 2004 IEEE international conference on computer communications (InfoCom 2004); 2004. p. 1512–21.

Chen S, Shen B, Wee S, Zhang X. Segment-based streaming media proxy: modeling and optimization. IEEE Transactions on Multimedia 2006;8:243–56.

Chen S, Shen B, Wee S, Zhang X. Sproxy: a caching infrastructure to support Internet streaming. IEEE Transactions on Multimedia 2007;9:1064–74.

De Mauro AD, Schonfeld D, Casetti C. A peer-to-peer overlay network for real time video communication using multiple paths. In: Proceedings of the 2006 IEEE international conference on multimedia & expo (ICME 2006); 2006. p. 921–4.

Deshpande S, Noh J. P2TSS: time-shifted and live streaming of video in peer-to-peer systems. In: Proceedings of the 2008 IEEE international conference on multimedia & expo (ICME 2008); 2008. p. 649–52.

Please cite this article as: Ma KJ, et al. A survey of schemes for Internet-based video delivery. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.02.001

Dey JK, Sen S, Kurose JF, Towsley D, Salehi JD. Playback restart in interactive streaming video applications. In: Proceedings of the 1997 IEEE international conference on multimedia computing and systems; 1997. p. 458–65.

Fielding R, Gettys J, Mogul J, Frystyk H, Masinter L, Leach P, et al. Hypertext transfer protocol—HTTP/1.1, RFC 2616, Internet Engineering Task Force (IETF); 1999.

Fröjdh P, Horn U, Kampmann M, Nohlgren A, Westerlund M. Adaptive streaming within the 3GPP packet-switched streaming service. IEEE Network Magazine 2006;20:34–40.

Gilbert EN. Capacity of a burst-noise channel. Bell System Technical Journal 1960;39:1253–65.

Girod B, Kalman M, Liang YJ, Zhang R. Advances in channel-adaptive video streaming. In: Proceedings of the 2002 IEEE international conference on image processing (ICIP 2002); 2002. p. I-9–12.

Goel A, Krasic C, Walpole J. Low-latency adaptive streaming over TCP. ACM Transactions on Multimedia Computing Communications and Applications 2008;4(20).

Goyal VK. Multiple description coding: compression meets the network. IEEE Signal Processing Magazine 2001;18:74–93.

Guo M, Zhang Q, Zhu W. Selecting path-diversified servers in content distribution networks. In: Proceedings of the 2003 IEEE global telecommunications conference (GlobeCom 2003); 2003. p. 3181–5.

Hui SC, Lee JYB. Playback-adaptive multi-source video streaming. In: Proceedings of the 2006 IEEE global telecommunications conference (GlobeCom 2006); 2006. p. 819–922.

Huynh-Thu Q, Ghanbari M. Scope of validity of PSNR in image/video quality assessment. IEEE/IET Electronics Letters 2008;44:800–1.

Jurca D, Chakareski J, Wagner J, Frossard P. Enabling adaptive video streaming in P2P systems. IEEE Communications Magazine 2007:108–14.

Kalapriya K, Nandy SK. Throughput driven, highly available streaming stored playback video service over a peer-to-peer network. In: Proceedings of the 2005 IEEE international conference on advanced information networking and applications (AINA 2005); 2005. p. 229–34.

Kalman M, Steinbach E, Girod B. Adaptive media playout for low-delay video streaming over error-prone channels. IEEE Transactions on Circuits and Systems for Video Technology 2004;14:841–51.

Kao C, Lee C. Aggregate profit-based caching replacement algorithms for streaming media transcoding proxy systems. IEEE Transactions on Multimedia 2007;9: 221–30.

Kim T, Ammar MH. A comparison of heterogeneous video multicast schemes: layered encoding or stream replication. IEEE Transactions on Multimedia 2005;7:1123–30.

Kusmierek E, Dong Y, Du DHC. Loopback: exploiting collaborative caches for large scale streaming. IEEE Transactions on Multimedia 2006;8:233–42.

Li Y, Ong K. Optimized cache management for scalable video streaming. In: Proceedings of the 2007 ACM international conference on multimedia; 2007. p. 799–802.

Liang YJ, Girod B. Network-adaptive low-latency video communication over best-effort networks. IEEE Transactions on Circuits and Systems for Video Technology 2006;16:72–81.

Liang YJ, Apostolopoulos JG, Girod B. Analysis of packet loss for compressed video: does burst-length matter? In: Proceedings of the 2003 IEEE international conference on acoustics, speech, and signal processing (ICASSP 2003); 2003. p. 684–7.

Liang YJ, Apostolopoulos J, Girod B. Analysis of packet loss for compressed video: effect of burst losses and correlation between error frames. IEEE Transactions on Circuits and Systems for Video Technology 2008;18:861–74.

Li Y, Markopoulou A, Bambos N, Apostolopoulos J. Joint power/playout control schemes for media streaming over wireless links. In: Proceedings of the 2004 IEEE packet video workshop; 2004.

Li Y, Markopoulou A, Bambos N, Apostolopoulos J. Joint packet scheduling and content-aware playout control for video sreaming over wireless links. In: Proceedings of the 2005 international workshop on multimedia signal processing; 2005.

Li J, Yeo CK, Lee BS. Peer-to-peer streaming scheduling to improve real-time latency. In: Proceedings of the 2007 IEEE international conference on multimedia & expo (ICME 2007); 2007. p. 36–9.

Li Y, Markopoulou A, Apostolopoulos J, Bambos N. Content-aware playout and packet scheduling for video streaming over wireless links. IEEE Transactions on Multimedia 2008a;10:885–95.

Li X, Tu W, Steinbach E. Dynamic segment based proxy caching for video on demand. In: Proceedings of the 2008 IEEE international conference on multimedia & expo (ICME 2008); 2008b. p. 1181–4.

Li Y, Li Z, Chiang M, Calderbank AR. Content-aware distortion-fair video streaming in congested networks. IEEE Transactions on Multimedia 2009;11:1182–93.

Liu Y, Guo Y, Liang C. A survey on peer-to-peer video streaming systems. Peer-to-Peer Networking and Applications 2008;1:18–28.

Lu S, Lyu MR. Constructing robust and resilient framework for cooperative video streaming. In: Proceedings of the 2006 IEEE international conference on multimedia & expo (ICME 2006); 2006. p. 917–20.

Ma W, Du DHC. Reducing bandwidth requirement for delivering video over wide area networks with proxy server. IEEE Transactions on Multimedia 2002;4: 539–50.

Ma KJ, Bartoš R, Bhatia S. Scalability of HTTP streaming with intelligent bursting. In: Proceedings of the 2009 IEEE international conference on multimedia & expo (ICME 2009); 2009.

Markopoulou AP, Tobagi FA, Karam MJ. Assessing the quality of voice communications over Internet backbones. IEEE/ACM Transactions on Networking 2003;11:747–60.

Martinez JL, Cuenca P, Delicado F, Orozco-Barbosa L. On the capabilities of quality measures in video compresion standards. In: Proceeding of the 2006 IEEE canadian conference on electrical and computer engineering (CCECE 2006); 2006. p. 527–32.

Microsoft, IIS smooth streaming technical overview. Website 〈http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=03d22583-3ed6-44da-8464-b1b4b5ca7520〉; 2009.

Microsoft, [MS-WMSP]: Windows media HTTP streaming protocol specification. Website 〈http://msdn.microsoft.com/en-us/library/cc251059(PROT.10).aspx〉; 2010.

Nguyen T, Zakhor A. Distributed video streaming over Internet. In: Proceedings of the 2002 SPIE multimedia computing and networking (MMCN 2002); 2002. p. 186–95.

Nguyen T, Zakhor A. Multiple sender distributed video streaming. IEEE Transactions on Multimedia 2004;6:315–26.

Pantos R. HTTP Live Streaming, internet-draft version 4 (draft-pantos-http-live-streaming-04). Internet Engineering Task Force (IETF), work in progress. 〈http://tools.ietf.org/html/draft-pantos-http-live-streaming-05〉.

Pao I, Sun M. Encoding stored video for streaming applications. IEEE Transactions on Circuits and Systems for Video Technology 2001;11:199–209.

Schulzrinne H, Rao A, Lanphier R. Real time streaming protocol (RTSP), RFC 2326, Internet Engineering Task Force (IETF); 1998.

Schulzrinne H, Casner S, Frederick R, Jacobson V. RTP: a transport protocol for real-time applications. RFC 3550, Internet Engineering Task Force (IETF); 2003.

Sen S, Rexford J, Towsley D. Proxy prefix caching for multimedia streams. In: Proceedings of the 1999 IEEE international conference on computer communications (InfoCom 1999); 1999. p. 1310–9.

Setton E, Noh J, Girod B. Low latency video streaming over peer-to-peer networks. In: Proceedings of the 2006 IEEE international conference on multimedia & expo (ICME 2006); 2006. p. 569–72.

Shen L, Tu W, Steinbach E. A flexible starting point based partial caching algorithm for video on demand. In: Proceedings of the 2007 IEEE international conference on multimedia & expo (ICME 2007); 2007. p. 76–9.

Systems A. Real-time messaging protocol (RTMP) specification. Website 〈http://www.adobe.com/devnet/rtmp/〉; 2009.

Tu W, Steinbach E, Muhammad M, Li X. Proxy caching for video-on-demand using flexible start point selection. IEEE Transactions on Multimedia 2009;11: 716–29.

Tullimas S, Nguyen T, Edgecomb R. Multimedia streaming using multiple TCP connections. ACM Transactions on Multimedia Computing Communications and Applications 2008;2(12).

Wang Y. Survey of objective video quality measurements. Technical Report WPI-CS-TR-06-02, Computer Science Department, Worcester Polytechnic Institute; 2006.

Wang B, Wei W, Guo Z, Towsley D. Multipath live streaming via TCP: scheme, performance and benefits. ACM Transactions on Multimedia Computing Communications and Applications 2009;5(25) [Article 25].

Wee S, Tan W, Apostolopoulos J, Etoh M. Optimized video streaming for networks with varying delay. In: Proceedings of the 2002 IEEE international conference on multimedia & expo (ICME 2002); 2002. p. 89–92.

Yajnik M, Moon S, Kurose J, Towsley D. Measurement and modelling of the temporal dependence in packet loss. In: Proceedings of the 1999 IEEE international conference on computer communications (InfoCom 1999); 1999. p. 345–52.

Please cite this article as: Ma KJ, et al. A survey of schemes for Internet-based video delivery. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.02.001