ACCEPTED FROM OPEN CALL

Mobile Video Delivery with HTTP

Kevin J. Ma, Azuki Systems, Inc. and University of New Hampshire
Radim Bartoš, University of New Hampshire
Swapnil Bhatia, Palo Alto Research Center and University of New Hampshire
Raj Nair, Azuki Systems, Inc.

ABSTRACT

Expansion in 3G cellular coverage and the emergence of more powerful mobile devices has increased demand for massively scalable mobile video delivery. The rapid adoption of the third screen as a primary screen for video has highlighted inefficiencies in the mobile delivery ecosystem and scalability issues in the mobile delivery infrastructure. This article provides an overview of the current mobile content delivery ecosystem and discusses the expanding role of HTTP-based mobile video delivery. A new class of HTTP-based mobile delivery protocols seeks to address existing quality and scalability issues by simplifying and standardizing mobile video delivery. This article shows how segment-based delivery has enabled HTTP-based live streaming and dynamic bitrate adaptation while increasing scalability through the use of existing CDN infrastructure.

INTRODUCTION

Over the past decade, Internet-based video delivery has become a fixture of modern life. Consumers have embraced the convenience, flexibility, and variety of Internet video, and content providers have acknowledged the earnings potential of this nascent market. The penetration of rich media into the mobile arena has fostered a new wave of innovation in mobile networking. Content providers and consumers alike see the third screen as a natural extension of the desktop video and television viewing experience. The wide availability of broadband access in the home and office combined with powerful desktop and laptop computers with ubiquitous support for the browser-based Adobe Flash® Player has enabled Internet-scale video delivery to the desktop. Challenges still exist, however, in migrating from traditional media outlets like TV and movies, and even from the desktop Internet, to the mobile environment. Limited cellular coverage with up to three orders of magnitude lower bandwidth and high latency, underpowered handsets with low screen resolutions, and a myriad of proprietary media player technologies continue to inhibit the scalable delivery of high quality video.

Content providers and content viewers demand and expect high quality video. Satisfying these demands is necessary for mass adoption, and mass adoption is required to make content delivery cost effective. The proliferation of 3G cellular access and the promise of 4G has enabled a new class of smart phones capable of providing high quality video to users. But this burgeoning market is hindered by the disparity in device capabilities and development environments. Smart phone hardware (e.g., CPU and memory), as well as operating systems, development tools, and third party libraries, have been evolving rapidly in recent years. Where PCs can be easily configured to support a broad array of delivery methods, protocol support varies widely across the major mobile device platforms, as shown in Table 1. Significant disparity between the desktop and mobile platforms makes it difficult to translate the common desktop solutions to the mobile domain. Device diversity among carriers further contributes to the lack of mobile platform convergence. The lack of convergence makes it difficult to optimize mass distribution of mobile content since custom approaches are often required for each platform. This need for customization has fostered a host of niche solution providers to handle each of the different cases. As each provider in the fragmented ecosystem seeks to differentiate themselves, an inconsistent user experience results. Employing different solution providers for every platform is very inefficient. A convergence of mobile video delivery methods is crucial to consolidation and growth of the mobile content delivery ecosystem. Be it at the device, operating system, or protocol level, consolidation will ease the transition from desktop to mobile, simplify mass distribution, enhance the monetization potential of mobile content, and drive industry growth.

This article takes a macroscopic view of the end-to-end mobile content delivery ecosystem. We first discuss the current state of the content delivery ecosystem and the mobile delivery network infrastructure. We then discuss advances in application-layer video delivery protocols, specifically the renewed interest in HTTP, and compare them to traditional approaches. We examine the media preparation involved in supporting these delivery protocols, for both video

0163-6804/10/$25.00 © 2010 IEEE

	Apple iPhone®	Blackberry	Google Android	Nokia®	Windows Mobile®
HTTP Progressive Download	Yes	Version 4.3+	Yes	No	Yes
RTSP Streaming	No	Version 4.3+	Yes	Yes	Non-native
HTTP Live Streaming	Version 3.0+	No	Not yet released	No	No
Windows Media® HTTP Streaming	No	No	No	No	Yes
Microsoft® Silverlight™ Smooth Streaming	No	No	No	Not yet released	Windows Phone 7
RTMP Streaming	No	Not yet released	Not yet released	Flash® Lite™	Flash® Lite™

Table 1. *Comparison of smart phone platforms.*

on demand (VoD) and live streaming, while taking into account the impact of these schemes on the ecosystem partners. While there are many important video delivery topics related to datalink, network, and transport protocols, video codecs and containers, error correction, etc., we focus our discussion on issues related to content delivery ecosystem partner integration. Understanding the interactions between ecosystem partners is a key component for evaluating mobile video delivery architectures. Understanding the business motivations of ecosystem partners allows optimization of video delivery architectures using a combination of business metrics and traditional networking metrics. Formulating these hybrid types of metrics would allow us to evaluate video delivery schemes from both a qualitative and quantitative perspective.

Mobile Video Delivery Ecosystem

The content delivery ecosystem is a complex network of partners and service providers. Figure 1 shows some of the primary entities, which interact to make Internet-based video delivery possible. There are many logistical hurdles involved in preparing content for mass distribution. In addition to producing content, the content provider must manage distribution, monetization, and delivery of the content. Figure 1 depicts this starting at the top with the content provider. Internet-based distribution partners are shown on the left, advertising partners are shown on the right, and the video preparation and delivery path is shown in the center.

Once the content is produced, it is placed in the content provider's content management system (CMS) for distribution to its partners. The content provider then initiates advertising and distribution discussions with its partners. Advertisers and sponsors are signed on to support the content, and advertising media is generated by the ad producers. The advertising media (videos, banners, interstitials, jump pages, etc.) are then published through the advertising CMS to be combined later with the content, by the content distributors. At the same time, contractors are employed for desktop Web development, mobile Web development, and mobile smart phone application development. These three distribution paths each require special expertise. The two mobile paths are also affected by the fragmented technology landscape. Different operating system development environments (for Apple iPhone®, Blackberry®, Google Android, Nokia®, Symbian, and Windows Mobile®, etc.), different carriers, and differences in device capabilities (e.g., screen resolution, audio/video codec support) require more specialized resources.

Figure 1 also depicts the interactions of Web sites and apps with ad traffickers and analytics tracking services. The ad traffickers determine which ads to show, and track which users saw which ads and when. Though the sponsored ads (solicited by the content providers and produced specifically for the content) may be delivered through the third party ad traffickers, they are typically directly integrated into the site or app. The primary role of the ad traffickers is to deliver remnant inventory (ads that are not sold against any specific content). The analytics tracking services track non-ad page views and app usage. Both provide valuable user demographic information to the content providers. Advertisements provide direct revenue, while the analytics information provides justification to advertisers and valuable market research for future projects. Without this revenue, the ecosystem cannot exist.

Mobile Video Transcoding

Unlike desktop systems which may support a wide variety of audio and video codecs and containers, mobile phones typically support very limited subsets of codecs and containers, with different devices supporting different subsets. The optimal parameters for each device, on a given network (video frame rate, video bitrate, audio sampling rate, audio bit rate, resolution, etc.) also differ. As such, the source media from the content providers need to be transcoded into many different encodings and formats. Complex compression schemes coupled with the massive amounts of data represented in high definition source videos make transcoding very CPU intensive.

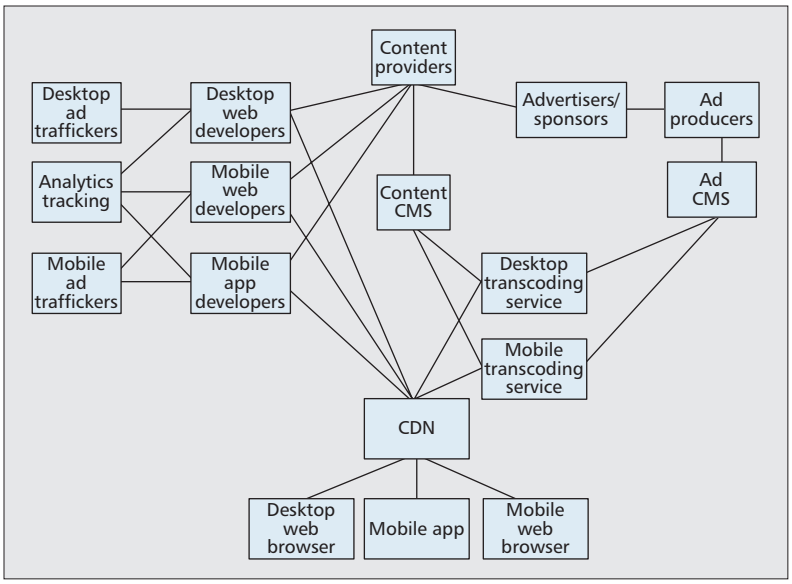

Figure 1. *Video delivery ecosystem.*

Transcoding services are often implemented as "Software as a Service" (SaaS) and deployed in a compute cloud, as depicted in Fig. 2. Because transcoding is a computationally intensive operation, it is typically only performed once, but needs to be performed quickly, to minimize distribution latency. Compute clouds offer large quantities of computational resources and bandwidth on demand. Using compute cloud resources is typically cheaper than maintaining dedicated servers and network connections for low frequency operations like transcoding. This provides content providers with incentives to use one-time transcoding services to produce static files for HTTP delivery. Though dynamic (on-demand) transcoding solutions offer greater flexibility [1], they are currently not a cost-effective solution.

Mobile Video Delivery Servers

In order to meet the challenges of Internet-scale delivery, content delivery networks (CDNs) are typically employed by content providers. Managing a global network of data centers requires significant resources. CDNs take advantage of economies of scale to more efficiently deliver content. Once transcoding is complete, transcoded files are uploaded to the CDN and replicated to the thousands of edge caches for delivery to end users. CDNs may provide an origin server for uploading content to, or may pull from a content provider's origin server and then manage the distribution to their edge servers. CDNs rely on distributed cache hierarchies for efficient synchronization of content. Delivery of relatively static content, via the simple and ubiquitous HTTP protocol, provides the most cost-effective solution for CDNs. Stable content requires less synchronization and less bandwidth, while optimized HTTP servers can handle extremely high load. Specialty streaming servers typically support fewer concurrent sessions due to CPU-intensive video processing, require specially trained support staff, and may incur licensing fees from the technology vendor. Strictly HTTP-based solutions limit operational expenses and allow the CDNs to pass cost savings on to their customers (i.e., the content providers).

Through DNS proximity and load balancing, CDNs attempt to ensure that end users are directed to the optimal edge server for high quality delivery. This works well for desktop users, as Web browsers can directly access the CDN content. The final stage of delivery from CDN to mobile client, however, must also traverse the mobile operator infrastructure. Mobile operators typically maintain a large private network that connects cell towers to the Internet through a series of gateways, across the mobile backhaul. All requests for media must go through a carrier gateway before crossing the public Internet to the CDN. The gateway obfuscates the actual mobile device location, invalidating any proximity-based delivery path optimizations. This, combined with the sparse coverage, over-subscription, and limited aggregate bandwidth of carrier networks, make for a challenging environment.

Figure 1 and Fig. 2 show some of the many parties involved in making modern video delivery possible. Many of the partner categories shown in Fig. 1 may consist of multiple actual vendors (e.g., multiple advertisers and ad traffickers to maximize sponsorship and remnant revenues, and multiple mobile app developers and transcoding services, one for each mobile platform). The logistical cost of employing and managing such a large network of partners currently limits the financial viability of mobile video delivery, though content providers remain committed to delivering video to the third screen. Simplifying cross-platform delivery for mobile devices will help content providers pare down the number of partners on which they rely, lowering operational expenses. The key components of this consolidation are delivery protocol and file format. A single delivery protocol will simplify CDN integration. A single file format will simplify pre-transcoding requirements. Combining both with deterministic quality guarantees will help foster organic growth in the ecosystem. The following sections detail the evolution in mobile video delivery paradigms currently underway.

Mobile Video Delivery Methods

Early mobile video players had two basic options for video retrieval: download using the HyperText Transfer Protocol (HTTP) or streaming via the Real Time Streaming Protocol (RTSP). Initially HTTP was only used for download and play where the entire file had to be downloaded before playback could commence; progressive download, which allowed rendering while downloading, came later. Due to bandwidth limitations, downloading prior to and playing incurred significant latency. Consequently, downloaded videos tended to be shorter and of low quality to reduce file sizes and shorten download times. This also encouraged adoption of RTSP by many mobile devices, since RTSP requires very little data to be buffered before playback can commence, allow-

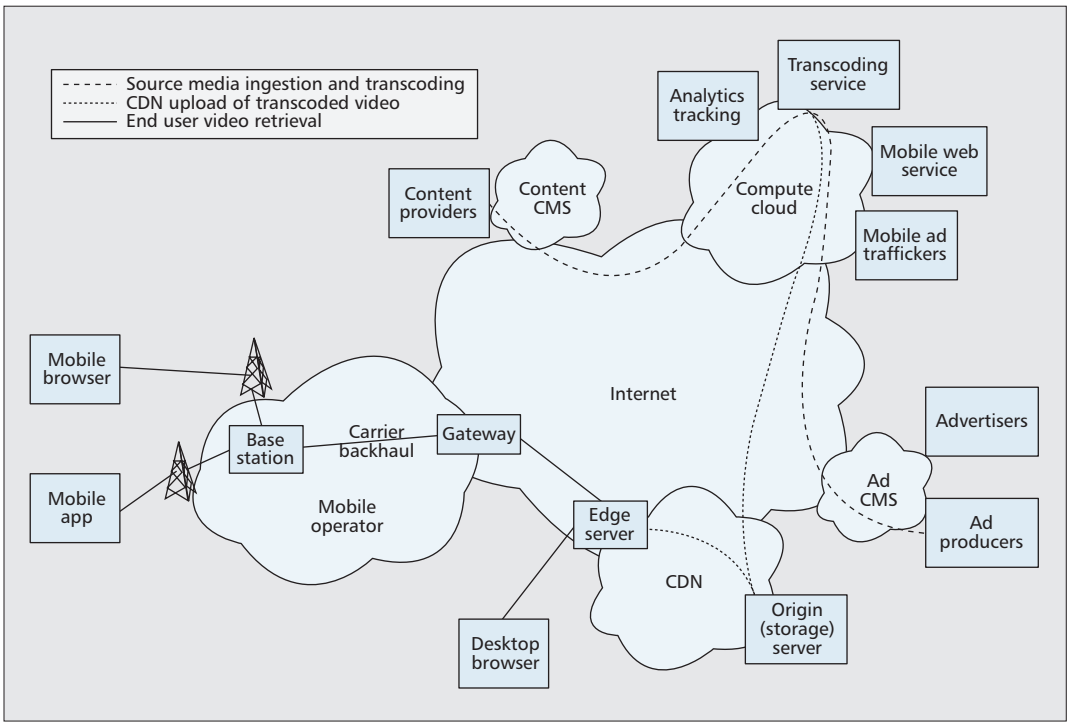

Figure 2. *Mobile video delivery network.*

The omnipresence of Web browsers in both desktop and mobile devices, the wide acceptance of HTTP for firewall traversal, the interchangeability of stateless HTTP servers, and the existing CDN data hosting and delivery infrastructures, make HTTP the de facto choice for most types of data delivery, including video.

ing for lower playback latencies. Download and streaming represent two extremes in video delivery. Figures 3a and 3b depict the traffic patterns for these two schemes. Hybrid schemes offer a compromise between download and streaming, as depicted in Figs. 3c and 3d.

Download and Play

Though numerous protocol options exist for downloading video files on desktop platforms, the most common protocol used in practice, especially for mobile platforms, is HTTP. The basic as-fast-as-possible delivery of HTTP was designed for delivering small amounts of data with minimal latency. Video files, however, tend to be much larger than HTML pages. HTTP is built upon TCP to ensure data integrity. For video, data integrity ensures that the intended picture quality is achieved, given timely delivery of data. In modern desktop environments, where broadband connections are typical, packet loss is relatively rare and bandwidth is relatively plentiful. But, in lossy, congested, high latency, low bandwidth 2G cellular networks, the relatively high number of high latency retransmissions required to support data integrity can disrupt playback continuity. The greedy delivery traffic pattern may also cause undue congestion due to premature delivery of data. In cases where the user access pattern involves partial viewing, bandwidth may be wasted if not all of the downloaded data is rendered [2]. Nonetheless, the primary advantage of HTTP is its ubiquity. The omnipresence of Web browsers in both desktop and mobile devices, the wide acceptance of HTTP for firewall traversal, the interchangeability of stateless HTTP servers, and the existing CDN data hosting and delivery infrastructures, make HTTP the de facto choice for most types of data delivery, including video.

Streaming

Unlike download, streaming relies on just-in-time data delivery with just-in-time rendering. Though many proprietary streaming protocols exist, the most common standardized protocol is RTSP. RTSP is a control protocol that uses the Real-time Transport Protocol (RTP) to deliver individual video frames over unreliable UDP transport. Just-in-time delivery uses less instantaneous bandwidth than as-fast-as-possible delivery and typically requires less client buffer space to be reserved. Spreading out bandwidth usage over time can help prevent congestion, assuming the delivery rate does not exceed the available bandwidth. This paced delivery can also prevent unnecessary bandwidth usage when user access patterns include random seeks or incomplete viewing. The real-time nature of streaming also makes it suitable for delivering live video. Though streaming is more bandwidth efficient, because frames arrive at the last possible moment, there is no time for retransmits. As such, there is no advantage to using a reliable transport like TCP. UDP provides "graceful" degradation of picture quality with minimal playback stoppages; as individual frames are lost, pixelation or rendering distortions will be noticeable to the user. Frame-based delivery allows for intelligent dropping of frames (e.g., non-key frames, or frames from low motion scenes) [3], but this requires an intelligent network that knows which frames to drop. This level of intelligence is not generally found in the Internet today. Frame-based delivery combined with feedback from the Real-time Transport Control Protocol (RTCP) can also be used to implement dynamic video bitrate adaptation [4].

As handset capabilities and mobile network infrastructure have steadily advanced, a new class of hybrid delivery schemes has emerged. These schemes combine the video quality guarantees of TCP-based HTTP, with the bandwidth management and dynamic video bitrate adaptation capabilities of RTSP/RTP.

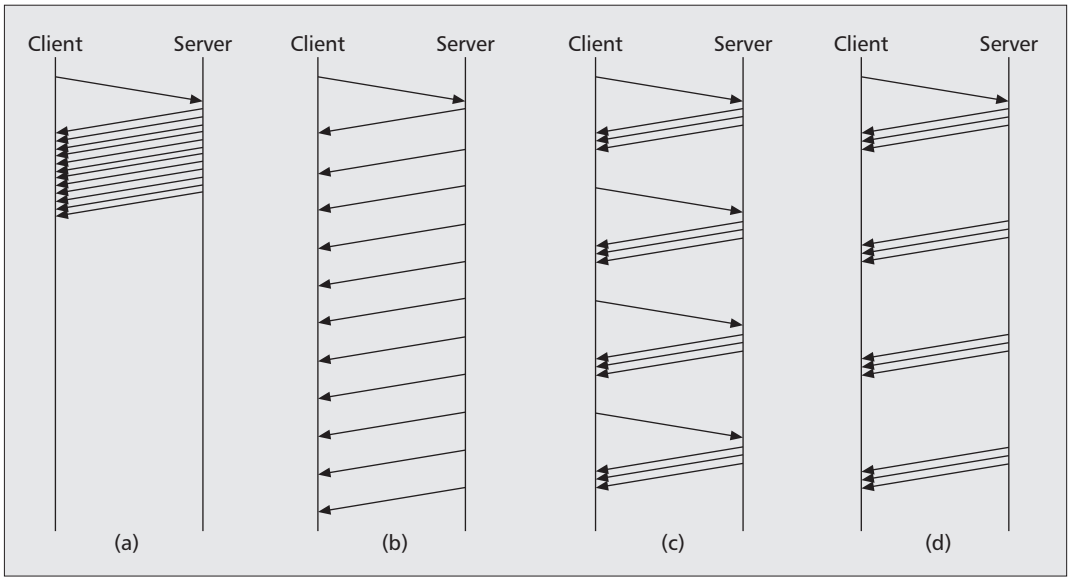

Figure 3. *Video delivery methods: a) HTTP straight download; b) simplified RTSP/RTP streaming (Fig. 4; c) HTTP paced range/segment download; d) HTTP paced download.*

HYBRID SCHEMES

Over the past couple of years, as handset capabilities and mobile network infrastructure have steadily advanced, a new class of hybrid delivery schemes has emerged. Resource scarcity with first generation smart phones and second generation cellular networks limited the effectiveness of HTTP delivery optimizations, but recent advances have made optimized HTTP more viable. Hybrid schemes seek to strike a balance between the two extremes depicted by Figs. 3a and 3b. The former sends the entire video as fast as possible, while the latter sends the smallest possible increment of video as slow as possible. Figures 3c and 3d show two alternatives for retrieving video in bursted segments. These schemes combine the video quality guarantees of TCP-based HTTP, with the bandwidth management and dynamic video bitrate adaptation capabilities of RTSP/RTP.

Figure 3c depicts the traffic pattern for a *client-side* pacing scheme, while Fig. 3d depicts the traffic pattern for a *server-side* pacing scheme. Client-side pacing over HTTP is typically implemented using Range requests or by requesting segmented files (which have been pre-chopped), to retrieve smaller amounts of data over time. Server-side pacing schemes typically rely on extracting metadata (i.e., the playout rate) from the video file and simply pacing the output. Many HTTP-based schemes also employ bursting data initially to preload the client-side buffer to decrease playback latency [5]. Both client and server pacing schemes typically base their pacing rates on the aggregate constant video bitrate. Schemes will often employ delivery rates that overestimate the video bitrate to prevent player underruns [2]. Most schemes also estimate available bandwidth and latency either through explicit checks, or implicitly through TCP ACKs. Though less network efficient than pure streaming, paced progressive download enables video quality guarantees, through reliable TCP delivery, not available in traditional streaming schemes.

The differences between the four schemes shown in Fig. 3 are summarized in Table 2. The primary advantages of the two hybrid schemes are:

- Pacing provides more efficient bandwidth usage compared to straight download, especially for long-form content.
- Bursting segments containing many frames increases the timing margin for retransmitting a lost packet within the segment, compared to RTSP/RTP frame-based delivery.
- Defined segment boundaries provide convenient transition points at which to change the video bitrate, when performing dynamic bitrate adaptation.

The hybrid schemes rely on HTTP for data delivery which make them ideal for use with CDNs, whose infrastructure is already optimized for distributing mass quantities of data via HTTP. Though CDNs do support RTSP and other streaming protocols, the overhead of maintaining separate, more specialized servers makes supporting those protocols expensive and less desirable. Beyond the significant processing overhead, RTSP, in particular, also requires the use of four UDP channels (two RTP connections, one for audio, one for video, along with their corresponding RTCP connections) which further limits server scalability and complicates network design. For many networks, dynamic provisioning for a large range of UDP ports is undesirable as it typically requires real-time firewall "fixups" which tax the firewall and in many cases violates security policies. Figure 4 diagrams the network connections necessary for RTSP/RTP/RTCP-based streaming.

With RTSP/RTP, there is also the issue of gracefully degraded quality, due to random packet loss (e.g., network error or discard-based traffic shaping). Graceful degradation is nondeterministic and undesirable to content providers. Many interesting schemes have been shown to improve quality and predictability by limiting key frame loss [6], recovering from key

	Straight HTTP	RTSP/RTP	Client/Server Paced HTTP
Transport Protocol	TCP	TCP/UDP	TCP
Ports Required	1	1/4	1
Quality Assurance (TCP Retransmits)	Yes	No	Yes
Graceful Degradation (UDP Drops)	No	Yes	No
Bandwidth Management (Data Pacing)	No	Yes	Yes
Dynamic Bitrate Adaptation	No	Yes	Yes

Table 2. *Comparison of video delivery protocols.*

In most cases, small variations in bitrate are not discernable due to limitations in human perception, as well as limitations in compression techniques. As such, the benefits of dynamic transcoding are often not worth the cost.

frame loss [7], or proactively dropping non-key frames [3], but the non-deterministic nature of loss remains unchanged. With the hybrid schemes, TCP-based transport guarantees frame delivery, though not necessarily on-time delivery. Late delivery may result in playback stoppage, but stoppage is deterministic in terms of rendering (unlike pixelation due to frame loss).

Deterministic Bitrate Adaptation

Bitrate adaptation can take many forms. Dynamic transcoding may be used to provide continuously variable bitrate encodings, but dynamic transcoding is computationally and financially expensive. In most cases, small variations in bitrate are not discernable due to limitations in human perception, as well as limitations in compression techniques. As such, the benefits of dynamic transcoding are often not worth the cost. Deterministic bitrate adaptation relies on using a discrete number of pre-determined bitrates. Content providers may choose and approve bitrates based on previewed content. With guaranteed delivery, they can be confident that the quality of their delivered product will match that of the previewed content. Selection of which bitrate to use is typically based on bandwidth estimations, either implicitly by the server [8] or through direct client feedback [4, 9]. Deterministic bitrate adaptation minimizes congestion and reduces the possibility of late delivery. Most schemes require the availability of multiple pre-transcoded files. The following section compares some of the different video file organizations used to support deterministic bitrate adaptation.

Mobile Video File Formatting

Straight HTTP download was originally used in cases where the video was retrieved in its entirety before playback would begin. The simplicity of this scheme was that it required no file pre-processing other than transcoding to the correct formats and codecs for the target devices. For RTSP and some of the hybrid schemes, additional pre-processing is required, to support pacing and bitrate adaptation. For RTSP, videos must typically be "hinted" to aid in decoding the audio and video tracks in real-time, to support RTSP frame-based delivery. Some of the hybrid schemes rely on file segmentation. While file segmentation has been proposed for cache optimization [10, 11], in this context, file segmentation is used to support deterministic bitrate adaptation and near-live streaming.

Figures 5b–e show some of the common file transcoding options for supporting bitrate adaptation, given the source file shown in Fig. 5a. In all cases a small number of bitrates is used (e.g., three in the case of Fig. 5).

Figure 5b shows the simplest form of transcoding, where the source file is transcoded, in its entirety, into the lower bitrates. Given consistent bandwidth, these files could be used by any of the delivery protocols by selecting the highest bitrate that does not exceed the available bandwidth. This is the most common solution today. However, most packet switched networks are bursty and jittery and lack bandwidth consistency. This makes dynamic bitrate adaptation desirable. The files shown in Fig. 5b are not conducive to dynamic bitrate adaptation, given the overhead of retrieving the new bitrate file.

For HTTP (assuming HTTP Range requests are supported), switching bitrates requires the client to issue a request to retrieve the headers from the new bitrate file, then determine the byte offset for the frame corresponding to where the player left off, then issue another request to start retrieving data from that offset. For high latency cellular networks, these two additional round trip times (RTT) can cause playback disruptions. RTSP is better suited to take advantage of these files, but typically requires the additional overhead of RTCP feedback channels to determine when to switch [4].

One artifact of the streaming paradigm is that the server will continue to send data at the current bitrate until the client terminates the session or requests a rate switch. If the rate is too high, graceful degradation allows the network to drop packets as necessary. However, during periods of network congestion, a constant stream of RTP packets may actually prevent recovery. RTP packets may continue to flood the network at too high a bitrate, prolonging congestion. This may inhibit delivery of RTCP packets and further delay rate adaptation. With TCP-based approaches like HTTP, the built-in flow control mechanisms could help ease congestion sooner. TCP, though, is less well suited for real-time

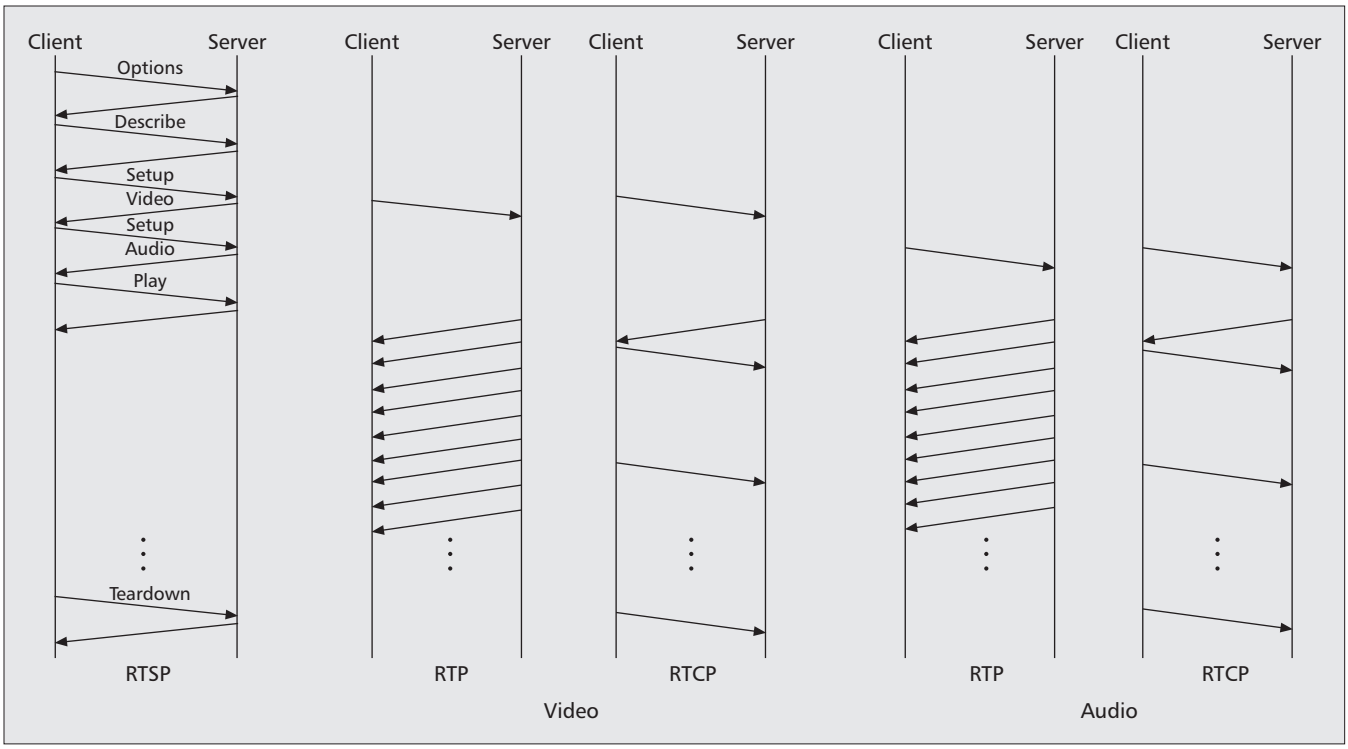

Figure 4. *RTSP/RTP connection setup.*

data delivery, e.g., live streaming. The lack of graceful degradation with progressive download requires more buffering, which introduces latency into the live streaming system, making it only near-live.

To mitigate the need for multiple requests when using HTTP to switch between the files shown in Fig. 5b, some hybrid schemes use segmented files, as shown in Fig. 5c. The source file in Fig. 5a is still transcoded into the same number of bitrates, but for each bitrate, a series of independently playable segments is created, rather than a single file, and the segments are played in succession to render the entire video. By creating segments, the header information is physically distributed, rather than being stored at the beginning of a monolithic file. Bitrate adaptation with segments may occur on segment boundaries without the need for separate header and data requests. Changing the segment duration allows for more or less dynamic bitrate adaptation. Though RTSP or dynamic transcoding may allow bitrate adaptation to occur on a shorter time scale, given the high RTT of most cellular networks, it is impractical to expect subsecond bitrate adaptation. Use of a reasonable segment duration can produce near-optimal bitrate adaptation in real world environments.

Figure 5d represents a cumulative layered encoding. The layers use a base encoding, and one or many layers that may be used to increase the quality of the base encoding [12]. Each layer is only useful given the base encoding and all previous layers. Layers can be sent independently, though it requires that the rendering engine issue parallel requests for each layer, as well as understand how to decode the data [13]. It has also been proposed that layers be sent from multiple senders [9], though most CDNs and clients have not yet reached that level of sophistication. For client-side bitrate adaptation, the client must always know where each layer resides and retrieve them in the proper combinations to reproduce the desired rate. Loss of a single layer can negate the value of all the higher layers; this makes cumulative encodings more complex to manage than non-cumulative encodings. Layered encodings can also be used in server-side optimizations (e.g., as part of an RTSP server) where each frame is constructed from as many layers as are appropriate for specific client bandwidth constraints.

Figure 5e represents an alternative to layered encodings: multiple description coding (MDC). MDC encoding can be either cumulative or non-cumulative [12], where non-cumulative encodings may be decoded independently. When distributed across multiple servers, MDC provides both data and path redundancy as well as allowing for bitrate adaptation, though the use of multiple descriptions incurs network header, video container, and data redundancy overhead. MDC schemes employ varying levels of redundancy between descriptions in order to provide higher or lower error resiliency [14]. There is a trade-off between limiting the distortion caused by the loss of a description and the increased data transmission overhead of the data redundancy. Higher data redundancy is also often required to produce non-cumulative encodings. The independently decodable nature of non-cumulative encodings improves resiliency and simplifies description selection. For client-side bitrate adaptation, the client may choose any descriptions it wishes from any of the servers. This flexibility reduces the data retrieval complexity compared to cumulative encodings, but it is still more complex than retrieving a single, pre-transcoded bitrate file.

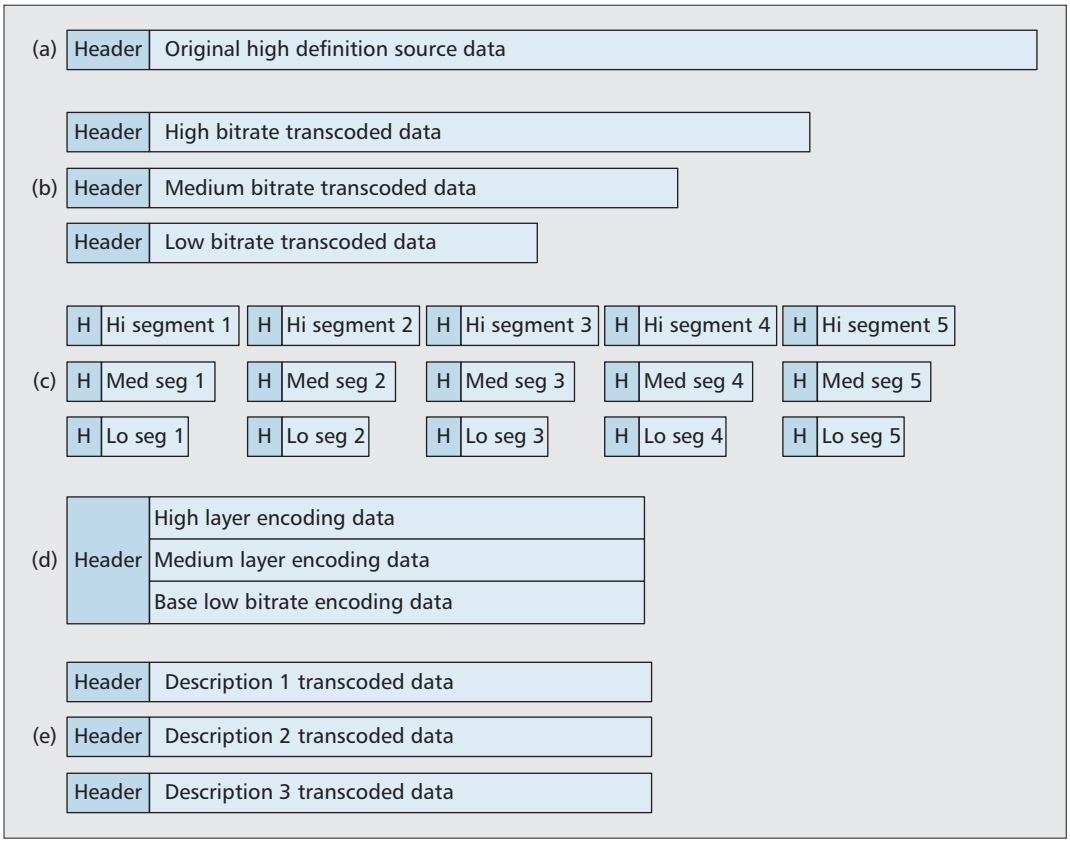

Figure 5. *Methods for preparing a video file: a) source file; b) independent bitrate encodings; c) segmented bitrate encodings; d) multi-layer encoding; and e) multiple description codings.*

Though RTSP or dynamic transcoding may allow bitrate adaptation to occur on a shorter time scale, given the high RTT of most cellular networks, it is impractical to expect sub-second bitrate adaptation. Use of a reasonable segment duration can produce near-optimal bitrate adaptation in real world environments.

Mobile Live Video Streaming

Most file pre-processing schemes work well for video on demand (VoD) but are less suitable for live streaming situations where the video is not available *a priori* for pretranscoding or stitching. Transcoding may be performed on site, by the live broadcaster. For desktop feeds, a single bitrate and format is typically sufficient, so there is no additional burden on the broadcaster. However, given the multitude of specialized formats required for mobile, deploying transcoders for each format and bitrate combination is impractical. Remote broadcast sites often lack sufficient bandwidth to upload multiple formats, and deploying multiple transcoders increases logistical complexity and operational cost.

Three commonly supported mobile live streaming protocols include: RTSP, Windows Media® HTTP Streaming (MS-WMSP), and HTTP Live Streaming [15]. Microsoft® Silverlight™ has not yet arrived for mobile, and Adobe Flash Lite continues to see limited market penetration. Windows Media®, Silverlight™ HTTP Streaming and Microsoft® Smooth Streaming, used by Windows Mobile® devices, and HTTP Live Streaming [15], used by Apple iPhone® devices, all rely on HTTP for data delivery. Adobe Flash® typically uses the Real Time Messaging Protocol (RTMP), which until recently, was an unpublished, proprietary protocol. Table 1 compares some of the current differences in the major smart phone platforms.

RTSP/RTP, with its frame-based processing, is ideal for processing live streams with minimal latency, as only the minimum amount of data (a single frame) needs to be processed before transmission can be continued. Segmentation schemes, though they incur a latency penalty for generating, transcoding, and uploading the initial segments, offer a near-live alternative to RTSP. Furthermore, segmentation schemes inherently record the live stream, which is not the case with RTSP. Since most live streams are recorded for subsequent VoD access, segmentation schemes simplify the recording and transcoding process for companion VoD.

Figure 6 depicts an example of HTTP Live Streaming [15] with dynamic bitrate adaptation. The HTTP Live Streaming scheme relies on a hierarchy of m3u8 playlist files. The m3u8 format is an extension to the m3u format used for mp3 audio playlists. The top level playlist contains static pointers to separate playlists for the individual bitrates. Each of the bitrate playlists contains a rolling list of pointers to segments. A segmenter is responsible for recording from the live stream and transcoding segments into the different target bitrates. Once new segments are available, the bitrate playlists are updated, adding the new segment and removing the oldest segment. The new segments and the updated playlist are pushed to the CDN for delivery, at regular intervals.

The client in Fig. 6 is passed a link to the master playlist, from which it obtains a list of available bitrates. Once a bitrate is selected, the client begins polling the playlist file correspond-

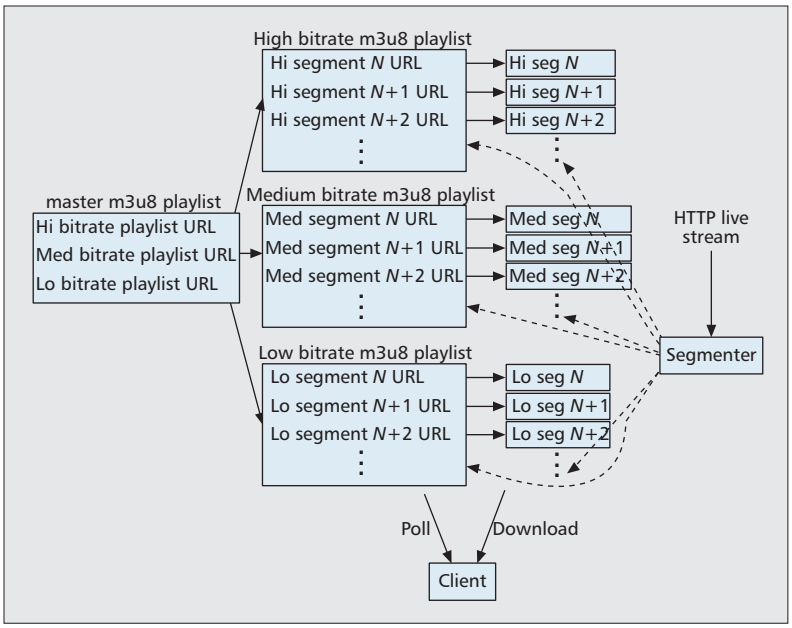

Figure 6. *HTTP live streaming with dynamic bitrate adaptation setup.*

ing to the selected bitrate. After the initial read of the playlist, subsequent reads would ideally occur at a regular interval equal to the duration of the segments. For uninterrupted playback, the asynchronous upload of segments and playlists, polling for playlist updates, and download and rendering of segments would need to align properly. The HTTP Live Streaming specification provides guidelines for polling and polling retry delays, where the delay should be equal to the segment duration if the playlist has changed, or half the segment duration for the first retry if the playlist has not changed (with subsequent retries backing off to 1.5 and 3.0 times the segment duration) [15].

Alternate polling schemes could be used to minimize the playlist update latency, but higher polling rates may incur greater network overhead. In general, the polling interval must be shorter than the duration of the segments, but longer than the end-to-end latency between client and server. If the polling interval is longer than the segment duration, then segments would finish playing before new segments could be retrieved. If the polling interval is shorter than the RTT, then the minimum effective polling interval is still just the RTT. The same playlist and segmentation scheme can be used to provide bitrate adaptation for VoD. In the VoD case, playlists contain a complete list of segments and may only be downloaded once.

When the client is initialized, it has two options: it can download all of the segments in the playlist or it can start by downloading the last (newest) segment in the playlist. The former option provides more data for protection against network error, but it incurs the largest viewing delay. The latter option minimizes viewing delay, providing a viewing experience as close to live as possible. Once the stream is playing, the client can monitor bandwidth and if it determines that a bitrate change is desired, it can begin polling a different playlist. The next segment retrieved will be for the new bitrate.

The polled playlist offers the flexibility to specify segments from different locations for redundancy, stream switching, or ad insertion. For live streams, stream switching (e.g., changing camera angles) or ad insertion is typically handled by the broadcaster. Ignoring the possible need for redundant servers, segmentation schemes can be implemented without playlists by using a well known file naming convention (e.g., using the segment number). Early versions of Silverlight™, as well as other commercial implementations, have taken this approach. Newer versions of Silverlight™ pack segments into a single file, and clients make requests providing indices which the server uses to extract the correct segment. This requires a specialized server to handle requests, which is expensive for CDNs and their customers. Another alternative is to provide an index file, where the index file maps segment boundaries to byte offsets, allowing the use of standard HTTP range requests. There are many options for implementing live streaming with HTTP, and we see significant commercial interest in exploring these options to find an optimal HTTP-based streaming solution.

The Road Ahead

Mobile rich media delivery is currently poised for a massive expansion, but lacks the platform uniformity and protocol convergence necessary for supporting a consistently high quality user experience. Disparities among handsets, handset manufacturers, operating system vendors, and mobile carriers all contribute to the ecosystem's inability to achieve mass adoption. End users, developers, and mobile operators would all benefit from ecosystem consolidation and common understanding of a converged video distribution solution. The current movement toward HTTP-based video streaming is one step on the road ahead.

Content providers want the highest quality delivery for their videos to protect their brand integrity. Streaming protocols based on unreliable protocols, like RTP over UDP, cannot by themselves provide deterministic quality guarantees. In addition, most carriers prefer not to open their networks to the dynamic UDP port allocation required by streaming protocols like RTP. The use of HTTP as a delivery protocol addresses both of these issues. The reliable TCP-based delivery, in conjunction with deterministic rate adaptation, provides deterministic quality guarantees, while the ubiquity of HTTP allows it to easily traverse firewalls and take advantage of optimized CDN caching infrastructures. Though HTTP was not designed for real-time applications and may not be optimal for video delivery, it provides the desired functionality using a broadly supported protocol. New protocols or enhancements to existing protocols, such as RTP, may eventually provide quality guarantees and ubiquity to surpass HTTP, but in the near term HTTP offers a pragmatic alternative. Likewise, client-side hardware support and server-side distribution support for layered and MDC encodings may eventually reduce complexity and make them more commercially viable solutions, but for now pre-transcoded single bitrate files

provide natively supported solutions.

HTTP has proven its versatility as a reliable data delivery protocol, exemplified by its use in CDNs. The unmatched scalability of CDNs have fundamentally changed media delivery in the Internet by bringing economies of scale to individual content providers. By pushing content to the network edge through well provisioned caching hierarchies, CDNs have increased media delivery scalability and improved the user experience. However, mobile carrier infrastructure still poses an obstacle to optimal HTTP delivery. The closed nature of carrier networks restricts the ecosystem's ability to fully optimize mobile video delivery. Extending the reach of the CDN into the mobile operator domain is a logical next step on the road ahead. These wireless CDNs (wCDN) constitute the final missing partner in the mobile content delivery ecosystem. We are just beginning to see the emergence of HTTP-based mobile video delivery protocols, but this is a pivotal step in the evolution of the mobile content ecosystem.

These wireless CDNs (wCDN) constitute the final missing partner in the mobile content delivery ecosystem. We are just beginning to see the emergence of HTTP-based mobile video delivery protocols, but this is a pivotal step in the evolution of the mobile content ecosystem.

References

[1] B. Shen, W. Tan, and F. Huve, "Dynamic Video Transcoding in Mobile Environments," *IEEE Multimedia Mag.*, vol. 15, no. 1, Jan. 2008, pp. 45–51.
[2] L. Guo *et al.*, "Analysis of Multimedia Workloads with Implications for Internet Streaming," *Proc. 14th ACM Int'l. Conf. World Wide Web (WWW 2005)*, May 2005, pp. 519–28.
[3] Y. Li *et al.*, "Content-Aware Playout and Packet Scheduling For Video Streaming Over Wireless Links," *IEEE Trans. Multimedia*, vol. 10, no. 5, Aug. 2008, pp. 885–95.
[4] P. Fröjdh *et al.*, "Adaptive Streaming Within the 3GPP Packet-Switched Streaming Service," *IEEE Network Mag.*, vol. 20, no. 2, Mar. 2006, pp. 34–40.
[5] L. Guo *et al.*, "Delving into Internet Streaming Media Delivery: A Quality and Resource Utilization Perspective," *Proc. 6th ACM SIGCOMM Conf. Internet Measurement (IMC 2006)*, Oct. 2006, pp. 217–30.
[6] Y. J. Liang, J. Apostolopoulos, and B. Girod, "Analysis of Packet Loss for Compressed Video: Effect of Burst Losses and Correlation Between Error Frames," *IEEE Trans. Circuits and Systems for Video Tech.*, vol. 18, no. 7, July 2008, pp. 861–74.
[7] Y. J. Liang and B. Gerod, "Network-Adaptive Low-Latency Video Communication Over Best-Effort Networks," *IEEE Trans. Circuits and Systems for Video Tech.*, vol. 16, no. 1, Jan. 2006, pp. 72–81.
[8] B. Wang *et al.*, "Multipath Live Streaming via TCP: Scheme, Performance and Benefits," *ACM Trans. Multimedia Computing Commun. and Applications*, vol. 5, no. 3, Aug. 2009, article 25.
[9] J. Chakareski and P. Frossard, "Distributed Collaboration for Enhanced Sender-Driven Video Streaming," *IEEE Trans. Multimedia*, vol. 10, no. 5, Aug. 2008, pp. 858–70.
[10] S. Chen *et al.*, "Sproxy: A Caching Infrastructure to Support Internet Streaming," *IEEE Trans. Multimedia*, vol. 9, no. 5, Aug. 2007, pp. 1064–74.
[11] X. Li, W. Tu, and E. Steinback, "Dynamic Segment Based Proxy Caching for Video on Demand," *Proc. 2008 IEEE Int'l. Conf. Multimedia & Expo (ICME 2008)*, June 2008, pp. 1181–84.
[12] T. Kim and M. H. Ammar, "A Comparison of Heterogeneous Video Multicast Schemes: Layered Encoding or Stream Replication," *IEEE Trans. Multimedia*, vol. 7, no. 6, Dec. 2005, pp. 1123–30.
[13] S. Chattopadhyay, L. Pamaswamy, and S. M. Bhandarkar, "A Framework for Encoding and Caching of Video for Quality Adaptive Progressive Download," *Proc. 15th ACM Int'l. Conf. Multimedia*, 2007, pp. 775–78.
[14] Y. Wang, A. R. Reibman, and S. Lin, "Multiple Description Coding for Video Delivery," *Proc. IEEE*, vol. 93, no. 1, Jan. 2005, pp. 57–70.
[15] R. Pantos, "HTTP Live Streaming," Internet Engineering Task Force (IETF), Internet-Draft Version 5 (draft-pantos-http-live-streaming-05), Nov. 2010.

Biographies

KEVIN J. MA (kevin.ma@azukisystems.com) received his B.S. in computer science from the University of Illinois at Urbana-Champaign in 1998, his M.S. in computer science from the University of New Hampshire in 2004, and is currently a Ph.D. candidate at the University of New Hampshire. He is a founding engineer at Azuki Systems where he is responsible for software system architecture. Prior to Azuki, he was a software engineer at Cisco Systems, coming to Cisco via the Arrowpoint Communications acquisition. He holds three patents in the area of content networking with numerous patents pending including many in the area of mobile content delivery.

RADIM BARTOŠ (rbartos@cs.unh.edu) received his Ph.D. degree in Mathematics and Computer Science from the University of Denver in 1997. He is currently an Associate Professor in the Department of Computer Science at the University of New Hampshire. Besides mobile media delivery, his research interests include mobile sensor networks, underwater networking and access networks.

SWAPNIL BHATIA (sbhatia@parc.com) is a postdoctoral researcher in the Bioinformatics group at the Palo Alto Research Center (formerly Xerox PARC). His research interests include modeling and analysis of networks and systems, theoretical work in Computer Science, and lately, proteomics. He received a B. Engg. degree in Computer Engineering from Bombay University's V.E.S. Institute of Technology and a Ph.D. degree in Computer Science from the University of New Hampshire.

RAJ NAIR (raj.nair@azukisystems.com) received his B.Tech in Mechanical Engineering from the Indian Institute of Technology, Madras in 1983, his M.S. in Mechanical Engineering in 1986 and Computer Science in 1988 from Iowa State University, and his Ph.D. in computer science from the University of Maryland in 1991. He is the CTO and a founder of Azuki Systems responsible for product architecture. Prior to Azuki, he was at Cisco Systems through the Arrowpoint Communications acquisition. As a founding engineer at Arrowpoint he pioneered the concept of layer 4-7 content switching. Previously he was also a principal engineer at Fujitsu Network Communications via the Nexion acquisition and also worked at Netrix Corporation prior to that. He holds eight patents in the areas of content switching and ATM flow control with multiple other patents pending.