Enhancing Internet Streaming Media with Cueing Protocols

Jack Brassil, Henning Schulzrinne[1]
Internet and Mobile Systems Laboratory
HP Laboratories Palo Alto
HPL-2000-173
December 15th, 2000*

E-mail jtb@hplhp.com, hgs@cs.columbia.edu

Real-Time Transport Protocol, multimedia signaling, content delivery networks

We propose a new, media-independent protocol for including program timing, structure and identity information in internet media streams. The protocol uses signaling messages called cues to indicate events whose timing is significant to receivers, such as the start or stop time of a media program. We describe the implementation and operation of a prototype internet radio station which transmits program cues in audio broadcasts using the Real-Time Transport Protocol. A collection of simple yet powerful stream processing applications we implemented demonstrate how application creation is greatly eased when media streams are enriched with program cues.

* Internal Accession Date Only
Approved for External Publication
[1] Columbia University, New York, NY

Enhancing Internet Streaming Media with Cueing Protocols

Jack Brassil
HP Laboratories
jtb@hpl.hp.com

Henning Schulzrinne
Columbia University
hgs@cs.columbia.edu

Abstract — We propose a new, media-independent protocol for including program timing, structure and identity information in internet media streams. The protocol uses signaling messages called *cues* to indicate events whose timing is significant to receivers, such as the start or stop time of a media program. We describe the implementation and operation of a prototype internet radio station which transmits program cues in audio broadcasts using the Real-Time Transport Protocol. A collection of simple yet powerful stream processing applications we implemented demonstrate how application creation is greatly eased when media streams are enriched with program cues.

Keywords — Real-Time Transport Protocol, multimedia signaling, content delivery networks

---

1. Introduction

The rapid deployment of Content Distribution Networks (CDNs) has refocused attention on large scale, live, multimedia internet broadcasts. CDNs rely on an interconnected network of servers providing 'application-level multicast' or 'splitting'; a media stream is transmitted to servers which replicate and forward the stream to either receivers or other downstream splitters. Transmission is typically unicast, and uses either a reliable transport protocol (e.g., TCP) or a proprietary, partially reliable protocol. Splitters can be located close to the network edge, such as at an internet access provider's point-of-presence. Proximity of splitters to receivers improves the likelihood of high reception quality. As we will see, this proximity also opens the door to the creation of a new generation of broadcast services.

What streaming CDNs lack in architectural elegance is more than compensated for by their global reach and lack of broadcast alternatives. Digital Island has stated their intent to create an infrastructure capable of transmitting 7.5 million simultaneous streams. Akamai has placed over 3000 edge servers, each capable of potentially serving hundreds of users. Mirror Image currently streams video

at rates up to 1Mbit/sec from its Content Access Points. Real Broadcast Networks successfully distributed over 2.3 million audio streams during a 24 hour live music concert in mid-1999. These emerging broadcasters are not lacking media content to distribute; there are approximately 2400 radio stations worldwide broadcasting on the internet today. And that rapid deployment of satellite CDNs such as IBeam Broadcasting seem likely to ensure that mass market video distribution is not far behind.

Given the rapid deployment of an internet broadcast network infrastructure, what additional technology is required to support internet broadcasting? We believe that the list of needed technologies is extensive. This would certainly include tools, protocols and systems to provide media discovery and announcements, broadcast network management, media personalization, and rights management. Indeed, one need only contemplate the services provided in conventional radio and television broadcasting to identify additional missing technology components and service offerings.

In this paper we consider what we believe to be one crucial yet missing enabling technology, a signaling mechanism — which we call a cueing protocol — for delivering program timing, structure and identity information in media streams. The elementary protocol message, a *cue*, typically indicates an event whose precise timing is significant to receivers, such as the start or stop time of a program or program segment. We define a program (or segment) to comprise a collection of transport layer media packets whose timestamps belong to a well-defined timestamp interval. Often the program will have an obvious application-level significance. For example, an internet television station might issue cues to delimit and label individual video presentations to facilitate recording by listeners using a Tivo-like consumer appliance.

The remainder of this paper is organized as follows. Section 2 provide background and motivation for our investigation of cueing protocols. Section 3 introduces the overall system objectives, a proposal for a cueing message format, and a discussion of the protocol's relationships with other internet protocols. An implementation of a prototype internet radio station using cues is presented in Section 4. A number of related issues and future research topics are discussed in Section 5, and our conclusions are stated in the final section.

2. Background

A diverse collection of cueing mechanisms are used to facilitate program insertion, switching and recording applications in traditional radio and television broadcasting. Examples of existing systems using some form of

’cues’ include:

1. Radio Data System (RDS) for VHF/FM Broadcasting [5]

   The RDS system provides traffic, station and song information to RDS-capable radio receivers, primarily in cars. Cues, known as flags, are sent out-of-band on unused, alternate frequencies. Users can view program information on a small alphanumeric character display, and operate radios in automatic switch-over mode to receive a travel alert or a preferred program type (e.g., news program). Though introduced and widely used in Europe, RDS has been weakly supported in the US. Plans to enhance the existing RDS infrastructure with a digitally-encoded travel message channel (TMC) are underway.

2. Program Delivery Control (PDC) [6]

   PDC is a system for encoding television program identity and start and stop times in teletext to facilitate recording on PDC-capable VCRs. Program identity labels (PILs) are transmitted at the start of each broadcast, and at one second intervals throughout. Corresponding published program information (e.g., Gemstar’s ShowView and VideoPlus+) can be obtained in printed TV guides and entered into a PDC-capable VCR. A variant of PDC known as Video Programming System (VPS) is available in Germany, Switzerland and Austria.

3. Pass through with Local Program Insertion in Cable Headends

   Insertion of local programming content at conventional analog cable television headends has been supported by DTMF ’cue tones’ transmitted along with program content from a signal source. The demodulated tones have been used to automatically trigger remote ad insertors and channel multiplexors [15]. Examples of cue commands include an ’Entry’ (start, pre-roll) tone delivered 8 seconds prior to a local insertion to provide adequate setup time for insertion equipment initialization. A corresponding ’Exit’ (stop, switch to network) tone indicates the end of an insertion period.

Even the most casual viewer of live broadcast analog television has observed that local program insertions are often executed poorly. Viewers routinely see, for example, a few seconds of the start of an advertisement, only to have it cut away to a second, presumably intended local advertisement. Lack of coordination is even more evident when local content is mixed with the signal source (e.g., a stock ticker persists through a local ad insertion).

Let's now consider today's internet media streams. No mechanism exists to directly convey program structure and identity with any measure of precision and granularity. Hence, our first objective is to create a protocol intended to facilitate the creation of internet services comparable to RDS, PDC, and local ad insertion services as they exist in conventional broadcasting. Indeed, we claim that we can create comparable services with far higher levels of production quality.

But we also aspire to do considerably more with the aid of a well-designed cueing protocol. We anticipate that the computational capabilities of internet-connected devices will grow dramatically, enabling new services which process streams in ways unanticipated by developers of conventional broadcasting systems. Embedding cues in media streams should facilitate the creation of novel stream processing applications [14], which might exist at either a receiver (i.e., an internet connected home 'appliance') or at a network intermediary (e.g., gateway or proxy). In the latter scenario, a network node (i.e., intermediary) receives streams from one or more sessions, processes these streams, and retransmits one or more possibly modified streams to other network intermediaries or receivers. Intermediaries might choose to forward, add or remove cues based on their utility to downstream devices. We expect that smart receivers will also process streams with embedded cues, but those streams will be terminated (i.e., not forwarded). Examples of applications benefiting from program cues include:

a. Recording

   In a program recording application, a program (or segment) is captured for future playback. Cues which delimit programs facilitate recording by uniquely identifying program content and precisely indicating program start and end points.

b. Insertion

   A program insertion application places a program segment within another program, or within an interstice (i.e., program gap). A typical use of program insertion is the dynamic placement of a commercial advertisement within an entertainment program. Such local insertions are routinely performed during an *out-of-network* commercial break by insertion equipment located at a cable television headend. For an internet broadcast, such a function is easily envisioned at or near a CDN's edge server at an internet access point. Cues may be used in this setting to demarcate an interstice (i.e., commercial break) or a program segment suitable for replacement.

c. Modification

   Overlaying a logo on program content is an example of a program modification. A second is program 'blanking' or removal. In blanking applications, content is removed from programs according to user preferences. For example, if certain program content is accompanied by a parental advisory notice indicated by cues, that content can be removed at the listener's or viewer's discretion. This approach could also be used to implement local 'black outs' of sporting events with market restrictions.

d. Switching

   Program switching applications select programs for forwarding from among one or more active streams being received. We anticipate that emerging internet radio and television stations will seek to personalize program content by monitoring and switching between active streams based on established listener or viewer preferences.

e. Adaptation

   Program adaptation or repurposing applications manipulate program content on behalf of diverse receivers. A typical adaptation is the transcoding of a video stream for forwarding to receivers otherwise incapable of either receiving or rendering the original stream.

Note that each of the above applications can be realized by alternate means without developing a cueing protocol. For example, an application may choose to use wall clock time as a reference to start/stop recording of a scheduled program. As an alternative, an application could rely on a pre-existing agreement about sequence numbers or timestamps, or even the initiation or suspension of packet flow, to indicate program initiation or termination.

But we maintain that cues are a simpler mechanism for maintaining tight time synchronization when processing streams. Failure to maintain precise time synchronization — say when switching between two source streams — could result in perceptible audible and visible artifacts. Tight time synchronization is also required in implementations where relatively little media packet buffering is available at a stream processing point.

Cues can be related to other types of signaling messages and protocol exchanges. Since cues typically identify events, they can be related to 'named events' as proposed for telephone signaling over RTP [2]. A named event is a message — in some cases in lieu of an in-band, encoded audio signal such as a DTMF tone — which can be used to trigger an event (e.g., tone generation) at a receiver. Cues can also be viewed as protocol message flowing downstream

with content for the purpose of content modification or enhancement at 'edge' servers, analogous to the role played by Internet Content Adaptation Protocol (I-CAP) [12] extensions to HTTP.

3. Protocol Elements and Architecture

3.1 Objectives

Our cueing protocol was designed to have the following desirable properties:

- Media encoding independence

  A single cueing mechanism should be used by all types of encoded media (e.g., MP3, PCM, MPEG-4). This property permits a single, common approach to stream handling and in most cases allows stream processing applications to be unaware of the details of the media encoding. Cues at the transport layer also enable stream processing without incurring latencies associated with identifying program structure as might be indicated at the application-level.

- Transport protocol independence

  In the remainder of this document we will focus attention specifically on how program cues can be added to RTP, the internet standard media transport protocol. However, it is our intent to have a common cueing protocol operate across all internet media transport protocols, including proprietary transport protocols as used by the Real Player (i.e., RDT) and Windows Media Player. Note that a common cueing mechanism still permits proprietary media encodings, if desired.

- Consistency with markers at other protocol layers

  Information about the structure or semantics of program content might exist at protocol layers other than the Transport Layer. For example, markers indicate appropriate entry and exit 'splice' points in MPEG-2 transport streams [16, 17]. Hence, it is possible that program cues, if used, could be either redundant or merely helpful indicators for certain media types. For example, a cue might indicate that the next arriving packet contains more detailed, media-specific splicing information requiring consideration of an application-level header within the transport packet payload.

- Cues as separate, optional packets

  Cues are chosen to be separate packets (i.e., distinct from media packets) and are always optional within a transport stream. Network intermediaries which receive an stream with embedded cues may add or remove cues to or

from a source stream prior to forwarding. For example, an audio stream broadcast to affiliate radio stations (for the purpose of rebroadcast) might include certain cues which contain locally significant or private information which need not be forwarded to listeners of the affiliate radio stations.

It is not strictly necessary to use separate packets for cues. For example, an alternate mechanism such as an optional media transport header extension could be defined to implement cues. However, the use of separate packets facilitates the addition or removal of cues from a stream, as well as other stream processing functions.

- In-band and out-of band operation

  Application creation is eased somewhat if cues flow in-band, that is, with cues and media packets forming a single stream, with cues distinguished by a separate identifier. We anticipate that in-band cues will be the dominant operating mode. However, out-of-band cues (e.g., cues and media packets forming two or more separate streams) are also supported. For example, out-of-band cues might be sent to a separate unicast port from the media stream, allowing a receiver to accept either the cues, the media stream, or both. Note that out-of-band delivery of cues might be desirable if privacy is sought, or if the out-of-band communications uses an underlying reliable protocol (e.g., TCP) while the media stream is carried by an unreliable protocol (e.g., UDP) [4]. In principle, an application could be written using both in-band and out-of-band cues. The cue format is identical regardless of the channel used to convey the cue. It is also possible to envision applications which receive cue streams but don't receive any media packets (e.g., a program directory service, or a digital rights tracking and accounting service).

- Time-sensitive program information

  Cues are intended for the limited purpose of carrying time-sensitive program information. Other out-of-band communication mechanisms (e.g., HTTP, Session Description Protocol [7] via Session Announcement Protocol [21]) should be used to carry program information which is relatively time-insensitive. An example of such program information would be an internet television station's weekly programming schedule announcement or future playlist.

3.2 Cue Types

The basic element of our protocol, the program cue, is constructed by creating a new RTP payload type. Hence, cues are distinguished from media packets of separate payload type when carried in-band.

This payload format is used for four principal types of signals:

1. Event Notification

   An Event Notification (EN) cue notifies the recipient of the initiation of an event.

2. Event Termination

   An Event Termination (ET) cue notifies the recipient of the completion of an event.

3. Event Pending

   An Event Pending (EP) cue notifies the recipient of an upcoming event. Depending on application requirements, a sender may issue multiple (redundant) EPs associated with each event at various times prior to the event.

4. Event Continuing

   An Event Continuing (EC) cue notifies the recipient that an event is in progress. Depending on application requirements, a sender may issue multiple EPs associated with each event at various times during an event.

A compliant receiving implementation should support the cues listed above. In some cases, an implementation may simply ignore or delete some or all cues. The extensible protocol design permits the addition of new cue types, as necessary.

3.3 Use of RTP Header Fields

In addition to a new RTP payload type, crucial information needed for cue handling is located in the RTP header, which we now briefly review; see [3] for a complete explanation of RTP.

RTP is an internet standard protocol for transporting continuous media. Figure 1 shows the header format. The first twelve bytes of the header are required. The synchronization source (SSRC) is a random number which uniquely identifies the source of an RTP packet stream. Packets from a synchronization source are distinguished by a timestamp and sequence number. These fields are used by receivers for proper signal reconstruction and playout timing. The initial sequence number value is also random, and is incremented for each consecutively transmitted packet. Packets can and do arrive at their destination out-of-order.

The timestamp indicates the time of the sampling instant of the RTP payload relative to the initial timestamp value, which is random. The sampling rate for

many audio/video encoding formats is constant, well known, and registered with IANA; other formats can have time-varying sampling rates. Media formats are specified by the Payload Type (PT) field. Multiple packets can have the same timestamp, as in the case where a large video frame is grabbed, encoded, but then transported in multiple packets.

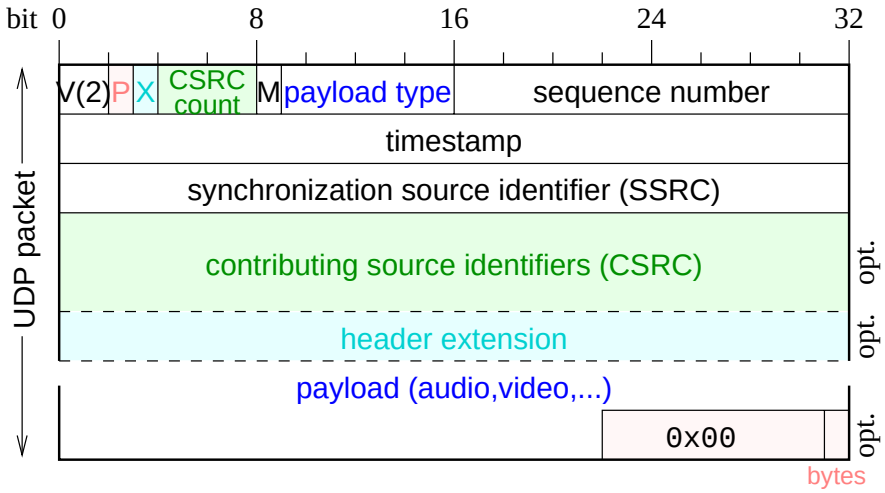

Figure 1 — RTP version 2 header.

A list of contributing source identifiers is present only if multiple RTP streams have been mixed. In this case, the CSRC count (CC) field indicates the number of contributors, and the CSRC list contains the original SSRC identifier of each contributing source.

RTP header fields in cue packets are used as follows:

1. Timestamp: The RTP timestamp reflects the measurement point for the event indicated by the current packet. The event duration, as described in the next section, extends forwards from that time. When sent in-band with media packets, the timestamp rate of cues is identical to the timestamp rate of the associated media.

2. Marker bit: The RTP marker bit set to 1 (0) indicates the beginning (end) of an event.

In accordance with current practice, this payload format does not have a static payload type number, but uses a RTP payload type number established dynamically and out-of-band (e.g., via a session announcement).

3.4 Cue Payload Format

The payload format is shown in Fig. 2.

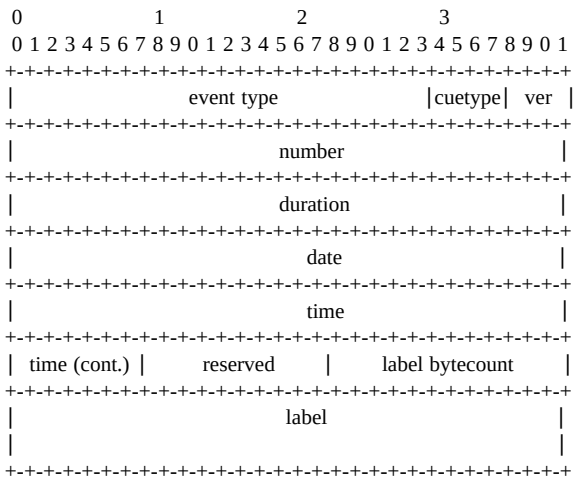

Figure 2 — Payload Format for Cues

Each field in the cue payload is defined as follows:

— event type: The event type is encoded as shown in Table 1.

— cue type: The cue type specifies the significance of the cue, as follows:

i. Notification: cue_type = 1 indicates an EN packet.

ii. Termination: cue_type = 2 indicates an ET packet.

iii. Pending: cue_type = 3 indicates an EP packet.

iv. Continuation: cue_type = 4 indicates an EC packet.

— ver: This field identifies the cue command protocol version. This paper describes a draft protocol with value 0x0.

— number: The number uniquely identifies an event of specified type. That is, the {event type, number} tuple uniquely describes a distinct event. Event type values can be either random, sequential, or assigned by a numbering authority. If no identifier is used, the value 0x00000000 is used.

— duration: The duration of an event is the time remaining before completion of the specified event, in timestamp units. For example:

1. An EP packet's duration specifies the time before the expected occurrence of the associated pending event.

2. An EN packet indicating the start-of-event has a duration set at the expected time until the corresponding end-of-event.
3. An ET indicating the end-of-event has a duration set to zero. However an exception exists if multiple ETs are required, which we discuss below.
4. An EC packet has a duration set to the expected time until the end of the currently continuing event's end.

— date: Society of Motion Picture and Television Engineer's (SMPTE) date encoding.

— time: SMPTE time encoding.

— reserved: This field is currently unused and reserved for future use.

— label bytecount: The byte count holds the length (in bytes) of the subsequent variable-length text field.

— label: A variable-length text field, possibly containing either a Universal Resource Name or a token suitable for display.

Tables 1 summarizes the encoding of the event type field in the cue payload format.

encoding (decimal)	Event type
0-10	<reserved>
11	<advertisement>
12	<video-frame>
13	<interstice>
14	<audio-track>
15	<audio-segment>
16	<video-segment>
17	<program-title>
18	<program-description>
19	<program-label>
20	<content-type>
21	<program-advisory>
22-1023	to be specified
>1023	private, by assignment

Table 1: A list of event types

The application developer will determine the appropriate event type for each application. In general, application developers decide how cues are most effectively used for their specific purpose. We make no attempt to develop a particular application protocol other than for the purpose of illustration. We

encourage the adoption of standard conventions by different developers creating similar or related applications. It is the responsibility of the application encapsulating data to ensure that packets are filled, and cues are added, in a manner facilitating stream processing, where possible.

3.5 Cue Placement and Redundancy

Cues placed in an RTP stream might arrive to a destination out-of-order. Hence, the precise placement of a cue in an RTP stream is not required. The timestamp and duration fields of a cue convey the precise time of an event, not the cues position within an RTP stream nor its sequence number.

It is the responsibility of a processing application to buffer enough packets to handle lost or out-of-order cues. However, placement of a cue at the time of an event it is marking can reduce the need for buffering. Hence, cues marking the beginning (EN) or end (ET) of an event cue should be placed in the stream within 50ms of the time of the associated event. Providing this timely notification will facilitate the creation of applications using small amounts of packet buffering per stream. A source should also avoid sending cues to mark events which have occurred at much earlier times, since these cues are unlikely to be useful to the receiver.

Cues placed in an RTP stream might fail to arrive to their destination. To achieve higher reliability, particularly when using an unreliable protocol (e.g., UDP) on lossy communication channels, a source may issue multiple cues to signal the same event (e.g., multiple EP followed by an EN packet). Applications will generally issue multiple cues in advance of the corresponding event for the purpose of redundancy. Implementations should be capable of properly handling redundant cues.

In general, application requirements dictate the appropriate *placement* of cues in an RTP stream. For example, EC packets might be repeated on a regular basis or injected at random times during an event.

Cues should not affect any mechanism that would normally be used to provide reliability to a media stream. For example, the forward error correction mechanism described in RFC 2733 [13] may of course continue be used to recover from media packet losses.

3.6 An example

Consider the following commercial ad insertion application. A broadcaster issues an EP cue (event type 13) 8 seconds prior to an interstice suitable for a program insertion. The network affiliate receives the notice, and initiates setup of insertion equipment. A second, redundant notification is sent 0.5 seconds

prior to the final RTP packet of the program segment preceding the interstice, providing the affiliate with an improved estimate of the upcoming interstice's start time. Subsequent to the final packet in the terminating program segment, an EN cue (event type 13) is issued. The downstream affiliate begins transmitting a new program to the user. This is preceded by the affiliate issuing an EN cue (event type 11). EC cues are issued by the broadcaster to the affiliate at 1 second intervals during the interstice. Immediately prior to transmitting a new program segment to the affiliate, the broadcaster issues an ET (event type 13) packet indicating the end of the interstice. The affiliate concurrently issues an ET (event type 11) to the viewer indicating the end of the inserted program.

In the above example, no cues were forwarded to receivers by the network affiliate; all cues transmitted by the broadcaster were removed from the stream.

3.7 Indicating Cue Usage in SDP and RTSP

Cues can be sent either with media packets or as a separate stream; the former case is anticipated to be most common. For the latter case, cues can be sent on separate multicast groups or separate ports from the media. In either case, these configuration options must be indicated out-of-band. Straightforward extensions (e.g., new attributes) can be used to communicate desired cue operation in both SDP and the Real-Time Streaming Protocol (RTSP) [20]. These descriptions are omitted for brevity.

4. Implementation

We next describe a simple prototype implementation of an 'internet radio station' transmitting streams enhanced by cues. In addition, we describe the implementation of stream processing applications which support desirable listener and broadcaster services.

Audio content for our internet broadcast is sourced from CD-audio and transcoded to MP3 files at 96kbs using a widely available tool such as Real Network's *RealJukebox*. The audio is transmitted on private, 10Mbs local area networks on a well-known multicast address with restricted scope. Individual MP3 audio tracks are streamed using RTP from Live.com's streaming server (i.e., livecaster version 1.5 for Linux).

A 'cue server' runs on the same machine as the streaming server. The cue server was little more than a modified version of rtpsend [18], a public domain program written by the author (HGS) to transmit stored or pre-recorded RTP packets. Of course, cue support will be most effective when natively supported by media servers rather than running as a separate server. The basic cue

operations are easily envisioned as implemented by extensions to the library rtplib [22]. The transmitted stream is 'marked up' with structural cues indicating the beginning and end of a program (i.e., new CD). In addition, cues indicate the beginning and end of each track, as well as program interstices for third party program insertions. In each case the transmitted objected is identified by cues with an appropriate event type and unique event number.

Our desire was to ensure that any media player could be used with our radio station without modification. But few players support RTP over IP multicast natively; plug-ins do exist for some players such as Nullsoft's popular *WinAmp*. However, even in cases where plug-ins exist we found that some media players become confused by receiving an RTP packet with payload type different from that of the media packets. A typical response observed was a media player 'hanging' (i.e., abruptly stop rendering the media stream) upon receiving a program cue. It is clear that future media players will need to be aware of in-band cues to ensure correct operation, even in the case where the player chooses to ignore all cues. For now, we were pleased to find a general solution which operates with all players. A freely available tool, Live.coms's playRTPMPEG, terminates RTP over IP (unicast or multicast) and pseudo-streams the MP3 payload via HTTP to any player, virtually all of which accept MP3/HTTP. This solution worked on a variety of clients we tested, including a prototype 'internet radio' appliance running FreeAmp on Linux [23].

To demonstrate how cues are helpful to both broadcasters as well as end users, we wrote the following two simple but illustrative applications:

1. Network-based recording service

   As a benefit for our internet radio station listeners, we developed a system to provide hands-free recording of individual MP3 tracks. The recording service operated on a separate machine from the internet radio station's streaming server. To use the service, a listener would view the radio station's web page, on which an advance playlist of a CD's tracks would be generated by querying the CD audio database www.CDDB.com and parsing the response. Viewing the playlist, a listener requested the recording of a specific track, invoking a CGI script. The script initiated a modified version of rtpdump [18], a publicly available tool which examines and outputs the header of RTP packets, which remained in a loop waiting for the program cue indicating the start of the desired track. Upon receipt of the start-of-track cue, the program invoked playRTPMPEG to save the incoming packets to a local MP3 file. The corresponding end-of-track cue terminated the writing. The resulting file was delivered to the requester as an attachment to a mail message, though any alternate

fulfillment mechanism could have been used. Note that no advance coordination was required between the media and recording servers, other than their shared information about the cueing protocol and its track identifier.

2. Third-party ad insertion

   To show the benefits of cues to broadcasters, affiliates, CDNs and other service providers, we developed a simple prototype system to provide third party ad insertion. To use the service, an advertiser would use an out-of-band method to acquire the right to use a specific program interstice in our broadcast. In fact, we routinely placed a 60 second program gap between broadcast audio tracks. The ad insertion program, operating on a separate machine from the internet radio station's streaming server, also used a modified version of `rtpdump`, which remained in a loop waiting for the cue indicating the start of the program gap. Upon receipt of the appropriate start-of-interstice cue, the program assumed control of the broadcast. The ad inserter first issued a start-of-ad cue using `rtpsend`, invoked its own streaming server, and broadcast an audio advertisement stored locally as an MP3 file. Upon conclusion of the insertion `rtpsend` was invoked again to issue the corresponding end-of-ad cue. The internet radio station then resumed control of the broadcast by issuing its corresponding end-of-interstice cue.

   Note once again that no advance coordination was required between partners sharing control of the broadcast, other than their shared information about the cueing protocol and the interstice identifier. Of course, a practical ad insertion system would likely operate differently. For example, we envision an inserter operating at — or in conjunction with — a CDN's edge server. Advertisement content could equally well be locally stored or cached, or requested from remote sources (e.g., using RTSP). All communication would be unicast, avoiding the well-known security problems of IP multicast (e.g., susceptibility to denial-of-service attacks). Cues between broadcast partners would generally not be passed to receivers In principle, different receivers could receive different edge-inserted advertisements, according to their preferences.

In each of the above applications cues were lightly used. With the audio stream operating at 96kbs, an additional 0.1% or less of bandwidth was consumed by cues. Even when cues are generously used in audio streams, there seems little reason to expect the incremental percent bandwidth to grow larger, particularly as streaming bandwidth increases. With the nominal streaming bandwidth for video being larger, there too we expect cue bandwidth consumption to be

insignificant.

5. Related Topics and Conclusions

5.1 Conveying Program Semantics

In addition to communicating program structure, cues may be used to convey program semantics. For example, cues within a video stream might delimit a collection of packets which contain a single image of significance to a receiver (e.g., <photo op> ... </photo op>). Use of such cues could facilitate a variety of highly time-sensitive adaptation or repurposing applications.

5.2 Proper Audio/Video Encoding and Transition Concealment

Consider again the problem of inserting an advertisement in a ’live’ or ’simulated live’ broadcast stream. It is clear that content coming from different sources need not be encoded with the same media type or bandwidth. The RTP specification [3] states that "A synchronization source may change its data format, e.g., audio encoding, over time." Hence, an RTP stream can be inserted in any standard format receivers can decode. It is each viewer’s responsibility to identify the payload type change and correctly adapt to the change. In practice, however, we have observed some video players responding correctly to changes in encoding scheme, but we have also seen audio players ’hang’ on rate changes between back-to-back MP3 tracks.

Even when a player can accommodate changes in rate and encoding, the transition should be aesthetically pleasing. When a local television station affiliate, for example, switches in a stored program advertisement during an out-of-network commercial break, the program transition should be free of visible switching artifacts. For example, a full frame should be sent first (e.g., an I-frame in MPEG-2). Such a recommendation is common in any video mixing system [19]. In principle this can be relatively easily realized for stored program advertisements. But achieving this becomes more difficult for "live" insertions, and also when returning control to "live" programming content. One possible solution is to have the intermediary discard video data until a full frame is received. The disadvantage of such a technique is obviously that switching between sources will take a longer, more variable period of time (e.g., perhaps 1/2 second before receiving the next I-frame in a typical MPEG sequence).

Concealment techniques can and should be used to hide aesthetically displeasing transitions. Perhaps the simplest video concealment technique is for both parties to agree on content surrounding the transition; the party completing its transmission can fade-to-black, while the party initiating its

transmission can begin with multiple black frames. Any number of special effects can be used to aesthetically improve the receiver's transition. In a similar fashion, proper volume control can be used to minimize audio artifacts (e.g., clicks).

5.3 Electronic Commerce

Cues enable new services, and these new services must be accompanied by a payment mechanism. For example, once a program insertion system exists, it is necessary to arrange for payment for insertions, as well as a mechanism for accounting for viewers. This is particularly true when insertions are 'personalized', and more than one program is simultaneously broadcast to different users of the same edge server. As viewer demographic information is returned to content providers in near real time, we expect to see payment mechanisms become more auction-like.

5.4 Security and Robustness

Methods to authenticate cues need to be explored. Signing a packet using a public-key algorithm is computationally difficult and would introduce at least 64 bytes of overhead per packet. Thus, it appears that one has to resort to non-technical means to discourage intrusions by malicious third parties.

The presence of cues allow end users to perform functions that might not be desired by broadcasters or content providers. For example, receivers can use software filters to block advertisements at their end systems. While hardware-based ad *blankers* exist for broadcast receivers, their use is not widespread; viewing or hearing nothing in lieu of an undesired advertisement offers viewers little advantage. But software filters are easy to construct, so their use is a threat. If filter use were to become common, broadcasters might be forced to consider additional countermeasures.

6. Conclusion

We have designed and implemented a prototype cueing protocol to support a new generation of internet broadcasting services. While we focused on the ability of cues to provide a mechanism for conveying program identity and structure, the protocol is extensible, and we envision it used for a variety of applications, including emergency notifications, alerts, and media personalization.

Our work optimistically anticipates that global IP network infrastructure improvements will continue at a rate sufficient to support acceptable quality real-time transport. To this end, we strove to develop a protocol which fits well with the tiered architecture of emerging CDNs.

A successful cueing protocol will effectively support local ad insertion. By allowing rich media content to be increasing personalized, we are supporting a business model for funding programming events which might be acceptable to content providers, internet broadcasters and viewers. As a consequence, internet broadcasting growth will be stimulated. Alternative funding models (e.g., pay-per-view) will likely coexist with commercial advertising as they do on broadcast television. We also recognize that many viewers find commercial advertising unappealing. For these viewers we have proposed mechanisms to ensure that ads can potentially delivered according to user preferences, or not viewed at all.

The rapid acceptance of software viewers such as *RealPlayer* and *Windows Media Player* and the emergence of commercial internet broadcasters suggest continued strong demand for dissemination of multimedia content. The commercial media broadcasting market is obviously not waiting for transport service guarantees, higher speed residential access technologies, new electronic payment systems, nor improved IP multicast infrastructure. Looking ahead, it appears as though the existing multimedia distribution approaches will serve only to accelerate demand for increased bandwidth and service quality. It remains to be seen just how much of this growing demand for multimedia 'broadcasts' will be satisfied by emerging CDN infrastructure.

References

[1] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: a transport protocol for real-time applications," *Request for Comments (Proposed Standard) 1889*, Internet Engineering Task Force, Jan. 1996.

[2] H. Schulzrinne, S. Petrack, " RTP Playload for DTMF Digits, Telephony Tones and Telephony Signals," *Internet Draft*, AVT Working Group, Oct. 1999.

[3] S. Bradner, "Key words for use in RFCs to indicate requirement levels," *Request for Comments (Best Current Practice) 2119*, Internet Engineering Task Force, Mar. 1997.

[4] J. Brassil, S. Garg, H. Schulzrinne, "Program Insertion in Real-Time IP Multicast," *ACM Computer Communication Review*, April. 1999.

[5] http://www.rds.org/rds98

[6] http://www.gemstar.co.uk/en/showview/pdc.html

[7] M. Handley and V. Jacobson, "SDP: session description protocol," *Request for Comments (Proposed Standard) 2327*, Internet Engineering Task Force, Apr. 1998.

[8] C. Perkins, I. Kouvelas, O. Hodson, V. Hardman, M. Handley, J. C. Bolot, A. Vega-Garcia, and S. Fosse-Parisis, "RTP payload for redundant audio data," *Request for Comments (Proposed Standard) 2198*, Internet Engineering Task Force, Sept. 1997.

[9] J. Whittiker, *DTV - The Revolution in Electronic Imaging*, McGraw-Hill, 1998.

[10] H. Schulzrinne, "RTP profile for audio and video conferences with minimal control," *Request for Comments (Proposed Standard) 1890*, Internet Engineering Task Force, Jan. 1996.

[11] Cugnini, A. G., "MPEG-2 Bitstream Splicing," *Proceedings of the Digital Television'97 Conference*, Overland Park, KS, Dec. 1997.

[12] http://www.i-cap.org/

[13] Rosenberg, J. and H. Schulzrinne, "An RTP Payload Format for Generic Forward Error Correction," *Request for Comments 2733*, Internet Engineering Task Force, Dec. 1999.

[14] Mitchell, S. et al, "A QoS Support Framework for Dynamically Reconfigurable Multimedia Applications," *Technical Report*, Distributed Systems Laboratory, University of London, 1998.

[15] DVS-075, "Cue Commands in Digital Systems", Digital Video Subcommittee, Society of Cable Telecommunications Engineers, Inc., March 25, 1997.

[16] DVS-253, "Digital Program Insertion Cueing Message for Cable", Digital Video Subcommittee, Society of Cable Telecommunications Engineers, Inc., September 27, 1999.

[17] J. After, "MPEG-2 Bit Splicing Revolution in Electronic Imaging", McGraw-Hill, 1998.

[18] H. Schulzrinne, *RTP Toolset Version 1.10*, http://www.cs.columbia.edu/~hgs/software.

[19] Cable Television Laboratories, Inc., "Digital Program Insertion: Request for Information," (April 1997).

[20] A. Dutta, H. Schulzrinne, Y. Yemini, "MarconiNet - An Architecture for Internet Radio and TV Networks," *IEEE NOSSDAV'99*, (1999).

[20] H. Schulzrinne, A. Rao, and R. Lanphier, "RTSP: A Transport Protocol for Real-Time Applications," *Internet Draft* (February 1998).

[21] M. Handley, "SAP: Session Announcement Protocol," *Internet Draft* (November 1996).

[22] *RTP Library*, http://gaia.cs.umass.edu.com/~drubenst/rtp_api.html.

[23] V. Krishnan, G. Chang, "Customized Internet Radio," *Proceedings of the 9th International World Wide Web Conference*, Amsterdam, 2000.