Payload Format for HTTP Encoding in RTP

1. Status of this Memo

2. Abstract

This document specifies a payload format for use in encoding HTTP within RTP. This payload format can be used for unreliable multicasting of resources such as Web pages, stock tickers, etc. As with other RTP applications, receiver feedback and group membership information is provided via RTCP. This specification is not expected to be used with unicast, since unicast applications will instead use HTTP over TCP.

3. Introduction

3.1. Purpose

Considerable interest has recently arisen in the multicasting of resources residing on HTTP servers. Many of these uses can be satisfied by unreliable multicast transport.

In this context, unreliable refers to applications where a repair mechanism is not required. These are typically applications where the material is of time value (stock tickers), so that it makes more sense to wait for the resource to be re-multicast than to attempt to repair it; applications in which an alternative means is available for retrieving the resource (cache filling); applications in which error

correction is performed at the datalink layer; or applications in which a separate error correction stream is transmitted along with the data, typically on a separate group.

In a cache filling application, there is a relatively small probability that a particular missing resource will be hit, and so it is often more costly to request a repair than to leave an incompletely received resource in the cache, where it may never be requested. Cache hits when they do occur, will typically be spread out over time, and therefore not synchronized to the original transmission. As a result, such applications present no danger of a NAK implosion.

The encoding specified in this document is not appropriate for use in applications where reliable transmission is required. Such applications present the possibility of a NAK implosion or congestive collapse, and so must be carefully analyzed prior to deployment.

3.2. Requirements

Before discussing the proposed HTTP encoding in RTP, it is useful to describe the requirements for unreliable multicast transmission of resources:

Source differentiation
Resource demultiplexing
Receiver reporting
Sender reporting
Layered encoding
Ability to synchronize with other media
Low overhead

3.2.1. Source differentiation

Since mixing is not useful for transmission of resources, and allowing multiple sources would make it difficult to maintain rate control, it is likely that only one source will be sending to a group at one time.

Nevertheless, in the case where there is a handoff, it is necessary to be able to differentiate sources, since packets from the two sources may be intermingled.

3.2.2. Resource demultiplexing

For multicast resource transmission, it is not desirable to have each resource transmitted with a unique source ID. Resources are typically of small size, and therefore the overhead of obtaining a source ID and setting up the transmission would be excessive. As a result, multiple resources will typically be transmitted with a single source ID. Since it is possible for packets from one resource to become intermingled with another due to out of order delivery, it is necessary to be able to demultiplex resources within a single source ID.

3.2.3. Receiver reporting

Although the encoding described in this document is to be used only for unreliable transmission, receiver feedback may still be desirable. Such feedback can be used to estimate listenership, packet loss rates, and receiver bandwidth availability.

Typically, receiver reporting information will be used both for engineering purposes (diagnosis of transmission problems) as well as for business purposes (listenership information). While receiver reports could be useful in allowing senders to adjust transmission parameters, typically it is more desirable to allow receiver-driven rate adaptation via layered encoding.

By transmitting a resource on several groups, each starting transmission with a different offset, the receiver may adjust their reception rate based on the available bandwidth. Typically, the group transmission rates will be tailored to commonly available bandwidths, i.e. 10 Kbps for 14.4 Kbps modems, 20 Kbps for 28.8 Kbps, 30 Kbps for single channel ISDN, etc.

Sender-driven transmission rate adjustment appears to be useful in only a limited number of circumstances. In cases where a small fraction of listeners are experiencing problems, it is undesirable to adjust the transmission rate; instead, the affected receivers should adjust their rate by leaving the higher bandwidth groups. If this does not work, they should stop listening to the transmission altogether.

A circumstance in which sender-driven transmission rate adjustment appears useful is in the case where the majority of listeners are only subscribed to the lowest transmission rate group, yet appear to lack the bandwidth to also join an error correction group appropriate to their packet loss rate. In this case the sender should back off the transmission rate on the lowest group to allow for successful reception of the error correction information.

As there are applications in which receiver feedback may not be feasible or desirable (satellite transmission), it must be possible to turn off the receiver reporting mechanism if desired.

3.2.4. Sender reporting

Just as receivers may wish to provide feedback to senders, so may senders wish to provide instructions to receivers. This may include information about the particular resource being transmitted (such as the resource length, related keywords, or URL), or information on the status of the transmission itself, such as the highest offset transmitted.

3.2.5. Layered encoding

Layered multicasting appears to be essential for multicast transmission of resources, since it provides for receiver-driven rate adaptation, as well as for transmission of error-correction information. While layered encoding was originally proposed for use in audio/video transmission, it can also be applied to data carousel style transmissions. In this application, each of the layers transmits the same resource, beginning at a different offset. Receivers with sufficient bandwidth may then subscribe to multiple groups, and receive the resource more quickly.

Layered encoding may also provide for error correction, by allowing error correction information to be transmitted on separate groups. Receivers may then subscribe to these groups according to their average packet loss rate; receivers experiencing high loss rates will typically join a higher bandwidth error correction group. In order to allow for the additional bandwidth of an error correction group, the sender transmission rate should be set appropriately.

3.2.6. Ability to synchronize with other media

While most uses of unreliable resource multicasting do not have real-time requirements, this may not be true of all such applications. For example, it may be desirable to synchronize display of a resource with an audio/video stream, as in a presentation or lecture, and in such an application it may be desirable to include an encoded clock.

3.2.7. Low Overhead

Since resource multicasting will typically use a small MTU size (i.e. 536 octets), it is important that a low overhead encapsulation be chosen. In order to achieve this, the GET request must not be sent in every packet. In addition, it may be desirable to support header compression.

3.3. Why RTP?

Given that RTP was originally created for use in realtime applications, it is not entirely obvious that it is the appropriate protocol to use for resource multicasting. However, given that this application appears to require receiver and sender reporting, layered encoding, and source and resource de-multiplexing, it is likely that an alternative framing will end up more closely resembling RTP than a simple UDP-based approach.

Where an alternative encoding would be most likely to differ would be in its reporting mechanisms. For example while the basic header of RMFP, described in [9], bears considerable resemblance to RTP, the sender and receiver packet formats differ considerably from RTCP. However, given the current state of knowledge, it is far from clear that the reporting needs of unreliable resource multicasting differ substantially from those provided by RTCP.

Thus while it is not apparent that RTP is the ideal protocol for use in this application, it appears to meet the application requirements. In addition, the behavior of RTP is well understood, and the protocol provides for functions such as RTP monitoring and header compression.

3.4. Overview

Multicasting of HTTP payloads is useful in a variety of applications, and as a result, several approaches have been taken. One of these is to put a single resource within a payload; another is to pack multiple resources in a payload. If multiple resources are placed within a single payload, this can be accomplished either via encapsulation or via a packing method such as MHTML, defined in [5].

This document specifies encapsulation of a single resource per payload. As defined in this specification, the HTTP payload consists of a preamble header, a MIME-like header, and entity-body content. Information on the resource being transmitted (such as the URI and the offset) is placed in the preamble header so as to avoid requiring receivers to parse MIME-headers in the HTTP payload in order to determine what portions of a resource have been received. As a result, this specification does not propose any new MIME headers, and any MIME headers allowable in an HTTP GET response may be enclosed in the payload.

This simplifies construction of unicast-multicast proxies, since the MIME-like header in the payload can be identical to that returned in the response to an HTTP-GET request. Proxies may therefore make a request for a URL, stuff the response into a payload, and multicast it. This is an efficient process since the proxy does not need to parse the response or construct an MHTML package.

4. RTP header

Rather than defining a new profile, this specification assumes that HTTP payloads defined in [8] will be transmitted using the RTP profile defined in [3], that is the RTP profile for audio and video conferencing. Additional required parameters are accommodated via definition of an HTTP payload format. As a result, there is no need for header extensions, SDES private extensions, new sender or receiver report fields, new RTCP packet types, or changes in the reporting interval constants.

Nevertheless, some clarifications are useful in describing how HTTP payload encoding is to be used with the profile defined in [3].

4.1. Extension bit

Although this specification does not preclude use of header extensions, it does not define any such extensions, and thus it is expected that the extension bit will typically be cleared.

4.2. CSRC count

Since HTTP payloads do not require mixing, there is no need for a contributing source field. As a result, the CSRC count field is set to zero.

4.3. Marker Bit

For use with HTTP payloads, a zero marker bit is used to indicate the last packet for a resource. The first packet is distinguished by a zero offset, and as a result, does not need to be marked.

4.4. Payload types

A dynamic payload type in the range of 96-128 is used. The binding of payload type and application is accomplished by non-RTP means, such as use of the "m=" and "a=" fields of the session description:

```
m=data 32768 RTP/AVP 98
a=rtpmap: 98 MHTTP
```

4.5. Port numbers

Although there is no official policy on this, current practice dictates usage of port ranges as follows:

[0, 16384]   - lowest priority, unclassified
[16384, 32768] - highest priority, i.e. audio
[32768, 49152] - medium priority, i.e. whiteboard
[49152, 65536] - low priority, i.e. video

It is recommended that unreliable multicast HTTP use the medium priority port range.

4.6. SSRC

Unlike with A/V payloads, a sender may use a single synchronization source for transmission of multiple HTTP payload streams. Thus a JPEG file may be transmitted with the same SSRC as an HTML file. This does not present a problem since the timestamp is used to uniquely identify resource streams.

4.7. Timestamp

Since a single synchronization source is used for transmission of multiple resources, an additional parameter is needed to identify packets within the same resource. The URI was not felt to be sufficient for this purpose, since a given resource may be multicast in data-carousel style, changing with each transmission.

As a result, the RTP timestamp field is used for this purpose. For use with HTTP payloads, the timestamp is constant for all packets that form a single transmission of a resource, and represents the time at which the sender began to transmit the resource.

Due to out of order delivery it is possible that packets from one resource will be intermingled with packets from another resource, sent to the same group and port. In this case the combination of the SSRC and timestamp can be used to demultiplex the resource stream.

5. HTTP Payload format

HTTP payloads consist of a preamble header, a MIME-like header, and entity-body content.

The purpose of the preamble header is identify the resource being sent. Depending on the caching implementation, partially received resources will either be discarded, or will be appropriately marked for storage in the cache. This will allow the cache to request the missing portion of the resource if a cache hit occurs.

Since this specification is primarily intended for use in situations where a local repair may not be timely (such as with a frequently updated stock ticker), where the resource will be re-multicast, or where error correction information is transmitted on a separate group, no repair mechanism is required.

While MIME-like headers could be used for resource identification (to provide information such as the URI, referring URI, content length and content range), prepending of a binary header reduces the amount of parsing required to get at this critical information. The use of a preamble header has the additional benefit of not requiring MIME-like headers to be present other than in the first packet. Thus, subsequent packets will typically contain only the preamble header and body content.

Thus, rather than adding MIME-like headers, it is intended that senders will simply retransmit the MIME-like header obtained in the HTTP GET response. Similarly, it is expected that the entity-body content will be retransmitted without modification.

5.1. Preamble header

The preamble header is comprised of an 8 octet fixed portion, and a single variable-length string.

```
                    1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                      Offset (8 octets)                        |
```

```
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       GET request...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   GET request....   |0|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

5.1.1. Offset

The offset field is 64-bits long. For a data packet, it represents the octet offset within the resource identified by the RTP timestamp. Although the offset field will typically increase with increasing sequence numbers, this need not always be the case, since portions of a resource may be transmitted out of order. Receivers should therefore be prepared to handle packet ordering based on the offset rather than the sequence number.

While the offset information could have also been provided using the Content-range: HTTP response header, this would have required insertion of a MIME-like header within each packet, and parsing of this header by the receiver. It is felt that the offset mechanism is more efficient.

5.1.2. GET request

The GET request field is a null-terminated string, representing the full text of the GET request that caused the resource to be transmitted. At a minimum, the GET request, defined in [8], includes the GET method, the URI and the HTTP version. While it begins on a 32-bit boundary, the GET request field is not padded to such a boundary.

Given the expected size of the GET request field, this variable-length string is expected to comprise the majority of the encapsulation overhead. Given that the string may be 40 octets or larger, when the 40 octets of IP/UDP/RTP header are added, the result may be 80 octets or more of overhead. This level of overhead would be unacceptable if it were present in every packet.

The GET request, offset, and packet length uniquely identify the portion of the resource being transmitted. Since receivers may join late, or miss portions of the transmission, receivers must be able to quickly bind a timestamp to a GET request so that the incoming resource and missing portions can be identified.

However, it is believed that this goal can be accomplished by placing the GET request within the first packet of a series, and then only within every subsequent nth packet. This results in a substantial reduction in overhead. For the purposes of this specification, it is suggested that n=4.

For packets in which the GET request is not included, a single null octet is used to indicate the missing field.

6. Resource length

The resource length is not included in the preamble header. Typically, this information is included within the first packet via the Content-length: header. Although it is possible that the first and/or final packet will be lost, we do not believe that the resource length justifies inclusion within the preamble header. This is because it is not necessary to know the total length of the resource to make a partial GET request for the remainder of the resource should a cache hit occur.

Note that it is possible that the final packets of a resource are lost. In this case, the receiver will note that it has not yet received a packet with the marker bit indicating completion of the resource transmission, after waiting a suitable interval after reception of the last packet. After this interval has expired, the receiver will write the incomplete resource out to the disk.

7. Non-RTP means

In addition to information transmitted within the RTP encoding, it is expected that receivers will make use of session information transmitted by non-RTP means.

For example, the existence of a data-carousel style session is expected to be determined via SDP, transmitted in one of its encapsulations, such as SAP. The SDP announcement will provide information on the bandwidth allocated to the session, as well as the group address (or in the case of a multi-group session, addresses) allocated to the session.

The SDP announcement will also be expected to indicate whether the data carousel transmission provides for re-multicast of the same resource (in which case receivers can make partial GET requests for the missing portion), or whether it provides continually updated information (in which case receivers missing a portion of the resource should make a GET request for the entire resource).

8. Layered encoding

Reference [4] discusses use of RTP in layered multicasting, and proposes guidelines for group address and port allocation, as well as modifications to RTP semantics suitable for allocation of SSRC identifiers across layered streams. SDES packets are then sent only on the lowest layer. It is expected that these modifications, once available, should be applicable to transmission of layered HTTP payloads.

9. Acknowledgements

Thanks to Peter Parnes of the University of Lulea, and Jim Gemmell, Paul Leach and Thomas Pfenning of Microsoft for useful discussions of this problem space.

10. References

[1] R. Braden. "Requirements for Internet hosts - application and support." STD 3, RFC 1123, IETF, October 1989.

[2] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson. "RTP: A Transport Protocol for Real-Time Applications." RFC 1889, GMD Fokus, January 1996.

[3] H. Schulzrinne. "RTP Profile for Audio and Video Conferences with Minimal Control." RFC 1890, GMD Fokus, January 1996.

[4] M. F. Speer, S. McCanne. "RTP Usage with Layered Multimedia Streams." draft-speer-avt-layered-video-01.txt, Sun Microsystems, LBNL, June 1996.

[5] J. Palme, A. Hopmann. "MIME E-mail Encapsulation of Aggregate

Documents, such as HTML (MHTML)." draft-ietf-mhtml-spec-03.txt, Stockholm University/KTH, ResNova Software, August, 1996.

[6] E. Levinson. "The MIME Multipart/Related Content-type." draft-ietf-mhtml-related-00.txt, Xison, May, 1996.

[7] J. Palme. "Sending HTML in E-mail, an informational supplement." draft-ietf-mhtml-info-03.txt, Stockholm University/KTH, August, 1996.

[8] R. Fielding, et al. "Hypertext Transfer Protocol - HTTP/1.1." draft-ietf-http-v11-spec-07, UC Irvine, August, 1996.

[9] J. Crowcroft, Z. Wang, A. Ghosh, C. Diot. "RMFP: A Reliable Multicast Framing Protocol." draft-crowcroft-rmfp-00.txt, UCL, November, 1996.

[10] P. Parnes. "RTP Extension for Scalable Reliable Multicast." draft-parnes-rtp-ext-srm-01.txt, LuTH/CDT, November, 1996.

11. Authors' Addresses

Bernard Aboba
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

Phone: 206-936-6605
EMail: bernarda@microsoft.com