1. Abstract

This document describes a method to improve the end-to-end bandwidth utilization of RTP streams over an IP network using compression and multiplexing. Several application level compression/multiplexing solutions have been evaluated so far in the IETF AVT Working Group. This proposal differs from other solutions in that neither compression nor multiplexing needs to be done at application level. Because of this, existing RTP based applications do not need to change to support this encapsulation format.

Instead of proposing a new encapsulation format for end to end multiplexing, this document describes the application of existing protocols for compression, multiplexing, and end to end tunneling.

2. Introduction

This document describes the application of existing protocols for compression, multiplexing, and end to end tunneling that can be used by RTP applications to implement an end to end multiplexing scheme for RTP transport. Header Compression is used to reduce the header overhead of a single RTP payload. Tunneling is used to transport compressed headers and payloads through a multiple hop IP network without having to compress and decompress at each link. Multiplexing is used to reduce the overhead of tunnel headers by amortizing a single tunnel header over many RTP payloads.
For compression, this document proposes the use of RFC 2508 based RTP header compression (CRTP). RFC 2508 describes the use of RTP header compression on an unspecified link layer transport. Most CRTP implementations use PPP as the link layer transport for the session. PPP has been integrated with a number of physical link layer protocols such as HDLC. When PPP is integrated with a physical link layer for CRTP transport, it has the disadvantage that headers must be compressed and decompressed at each IP hop in an end to end network.

A CRTP session can be made to work across multiple IP hops to enable end to end compression by tunneling the PPP session. The tunneling protocol proposed by this document is L2TP (RFC 2661). L2TP is a general tunneling protocol for PPP sessions. Since PPP is used as the link layer protocol for CRTP, the extensions described in RFC 2509 are required to negotiate the CRTP session.

When the overhead of a tunnel header is added to a single compressed RTP payload, there is very little bandwidth savings when compared to uncompressed transport of RTP streams. Multiplexing is required to amortize the overhead of the tunnel header over many RTP payloads. The multiplexing format that is proposed by this document is PPP multiplexing (Draft-ietf-pppext-pppmux-00.txt). PPP multiplexing allows many PPP payloads to be encapsulated as a single multiplexed PPP payload. The resulting multiplexed PPP payload can then be transported between two RTP endpoints using L2TP.

In order to make end to end transport of CRTP sessions efficient when using L2TP, some extensions are needed to both the CRTP protocol and the L2TP protocol. This document describes extensions that have been proposed for these protocols to make them more efficient when they are used as described in this document. These extensions to CRTP and L2TP have been proposed in separate Internet drafts.

3. Protocol Operation and Recommended Extensions

3.1. CRTP

When CRTP sessions are transported through a network using an L2TP tunnel, some of the basic assumptions used for CRTP over a single physical link may no longer be valid. Tunneling a CRTP session through multiple IP hops may increase the round trip delay and the chance of packet loss. CRTP contexts get invalidated due to packet loss. The CRTP error recovery mechanism using CONTEXT_STATE messages can compound the loss problem when long round trip delays are involved. This is because once the CRTP decompressor context state gets out of sync with the compressor, it will drop packets associated with the context until the two states are resynchronized. Resynchronization involves the transmission of the CONTEXT_STATE message from the decompressor to the compressor, and a FULL_HEADER message from the compressor to the decompressor.

Enhancements to CRTP are needed to minimize feedback based error recovery using context state messages. Draft-ietf-avt-crtp-enhance-00.txt proposes CRTP enhancements to make it more tolerant of packet loss, and minimize the need to use the CRTP error recovery mechanism. Specific recommendations for the use of the CRTP protocol when transported through a tunnel are described below.

The CU* packet format described in Draft-ietf-avt-crtp-enhance-00.txt should be used to synchronize CRTP compressor and decompressor state whenever the incoming packet stream causes a change in the compressor
context state. The CU* packet format allows any portion of the context state to be transmitted from the compressor to the decompressor.

To ensure delivery of state changes, CU* packets should be delivered using either the N mode of operation or the ACK mode of operation described in Draft-ietf-avt-crtp-enhance-00.txt. The method that should be used depends on the expected loss rate of packets in the network. Networks with a low loss rate for packets in the tunnel should use the N mode of operation. Networks with a high loss rate should use the ACK mode of operation.

UDP checksums should be used for RTP packets transported using TCRTP. The twice algorithm described in RFC 2508 should be used by the CRTP decompressor to resynchronize context state in the event of packet loss within the tunnel. In the event that UDP checksums are not generated by the application, the CRTP compressor should use the CRTP Headers checksum described in Draft-ietf-avt-crtp-enhance-00.txt.

Tunneled transport does not guarantee in order delivery of packets. Therefore, the CRTP decompressor must be capable of operation in the presence of out of order packet delivery. A CRTP decompressor may treat out of order delivery the same as packet loss. There is no need to reorder packets that are delivered out of order.

3.2 PPP Multiplexing

Draft-ietf-pppext-pppmux-00.txt describes an encapsulation that allows combining multiple PPP payloads into one multiplexed payload. The encapsulation format used for PPP multiplexing allows any supported PPP payload type to be multiplexed.

Draft-ietf-pppext-pppmux-00.txt describes the logic of an example PPP multiplexing transmitter. When PPP multiplexing is used with an L2TP tunnel, the transmitter will typically not have access to an interface transmit queue. In many PPP multiplexing implementations, the PPP multiplex transmitter will send packets to a tunnel encapsulation module. The tunnel encapsulation module will typically be implemented above the IP layer. This means that when the PPP multiplex transmitter encapsulates packets, the outbound physical interface for the packet will not be known. The result is that in implementations such as this, the PPP multiplex transmission algorithm as described in draft-ietf-pppext-pppmux-00.txt will never multiplex multiple PPP payloads into one multiplex PPP payload.

To enable the PPP multiplex transmission algorithm to work properly in tunneled implementations, some modifications to the transmission logic are needed. The transmission logic could be modified to collect incoming payloads to be multiplexed until one of two conditions occurred. The first condition is that a target number (N) of payloads or bytes has arrived at the multiplexer. The second condition is that a timer (T) which bounds delay in the multiplexer has expired. The first condition is used to ensure that the multiplexer encapsulates multiple payloads in the same PPP multiplex payload independent of the method used to hand packets to the next encapsulation layer. The second condition is used to always bound the amount of delay to an acceptable value. Delay due to multiplexing may become unacceptable in cases where there are not enough payloads arriving at the multiplexer to allow the multiplexed packet to be sent in a timely manner using only the first condition. The timer is reset whenever a multiplexed payload is sent to the next encapsulation layer.
The optimal values for $N$ and $T$ will vary depending upon the rate at which payloads are expected to arrive at the multiplexer and the amount of acceptable delay allowed in the network.

### 3.3. L2TP

L2TP tunnels should be used to tunnel the CRTP payloads end to end. This is a natural choice since CRTP payloads are PPP payloads, and L2TP allows tunneled transport of PPP payloads. L2TP includes methods for tunneling messages used in PPP session establishment such as NCP. This allows the procedures of RFC 2509 to be used for negotiating the use of CRTP within a tunnel and to negotiate compression/decompression parameters to be used for the CRTP session.

To get reasonable bandwidth efficiency using multiplexing within an L2TP tunnel, multiple RTP streams must be active between the source and destination of an L2TP tunnel. If the source and destination of the L2TP tunnel are the same as the source and destination of the CRTP sessions, then the source and destination must have multiple active RTP streams to get any benefit from multiplexing. Because of this limitation, TCRTP is mostly useful for applications where many RTP sessions run between a pair of RTP endpoints. The number of simultaneous RTP sessions required to reduce the header overhead to a minimum depends on how big the L2TP header is. A smaller L2TP header will result in fewer simultaneous RTP sessions being required to produce bandwidth efficiencies similar to CRTP.

Draft-ietf-l2tpext-l2tphc-00.txt describes a method of compressing L2TP tunnel headers from 36 bytes including the IP header to 21 bytes. L2TPHC packets include an IP header, using an IP protocol that is negotiated between the two hosts at the ends of the L2TP tunnel. The UDP header is omitted, and the L2TPHC header is reduced to 1 byte. The added overhead is now 21 bytes of the combined IP and L2TPHC headers.

When L2TP is used to carry CRTP streams, the RTP streams may use the EF DSCP. When an EF packet is tunneled, the tunnel header must be marked as EF. This is a requirement of RFC 2598. To prevent TCRTP tunnels from using excess EF bandwidth, it is recommended that only packets marked with the EF DSCP be transported in the tunnel with EF DSCP.

### 3.4 Encapsulation Formats

The packet format for an RTP packet compressed with RTP header compression is:

```
+-----------------+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Context       | MSTI  |             |                       |
|     ID          |   Link|   Checksum  |       RTP Data        |
|     (1-2)       |   (1)|     (0-2)   |                       |
+-----------------+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The packet format of a multiplexed PPP packet is as follows:

(diagram taken from draft-ietf-pppext-pppmux-00.txt)

```
+-----------------+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Mux | PPP  |
+-----|------+
| P   | Len1 |
| F    | PPP  |
+-----|------+
| P    | LenN |
| F    | PPP  |
+-----|------+
```
<table>
<thead>
<tr>
<th>Prot.</th>
<th>F</th>
<th>Prot.</th>
<th>Info1</th>
<th>~</th>
<th>F</th>
<th>Prot.</th>
<th>InfoN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td>(7 bits)</td>
<td>Field1</td>
<td></td>
<td>(7 bits)</td>
<td>FieldN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2)</td>
<td>(1 byte)</td>
<td>(0-2)</td>
<td></td>
<td>(1 byte)</td>
<td>(0-2)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The format of an L2TPHC packet with a PPP payload is:

```
+-----------------+---------------------+------------------------+
| IP header       | L2TPHC              | PPP payload            |
| (20)            | (1)                 |                        |
+-----------------+---------------------+------------------------+
```

The combined format used for TCRTP with a single payload is:

```
<table>
<thead>
<tr>
<th>IP hdr</th>
<th>L2TPHC</th>
<th>Mux</th>
<th>P</th>
<th></th>
<th></th>
<th></th>
<th>MSTI</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(20)</td>
<td>(1)</td>
<td></td>
<td>Prot.</td>
<td>Field</td>
<td>Field1</td>
<td>Prot.</td>
<td>ID</td>
<td>Link</td>
<td>Cksum</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(7 bits)</td>
<td>(2)</td>
<td>(1 byte)</td>
<td>(7 bits)</td>
<td>(0-2)</td>
<td>(1)</td>
<td>(1-2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(1 byte)</td>
<td>(0-2)</td>
<td>(1-2)</td>
<td>(1)</td>
<td>(1-2)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

CRTP is defined in RFC2508

Extensions to CRTP to make it more tolerant to packet loss are defined in: draft-ietf-avt-crtp-enhance-00.txt.

L2TPHC is defined in: draft-ietf-l2tpext-l2tphc-02.txt

PPP multiplexing is defined in: draft-ietf-pppext-pppmux-00.txt

4. Bandwidth Efficiency

The expected bandwidth efficiency attainable with TCRTP depends upon a number of factors. These factors include multiplexing gain, expected loss rate within the tunnel, and rates of change of fields within the IP and RTP headers. This section also describes how TCRTP significantly enhances bandwidth efficiency for voice over IP over ATM.

4.1 Multiplexing gains

Multiplexing reduces the overhead associated with the layer 2 and tunnel headers. Increasing the number of CRTP payloads combined into one multiplexed PPP payload increases multiplexing gain. As traffic increases within a tunnel, more payloads will be able to be combined in one multiplexed payload. This will increase multiplexing gain. The effect of multiplexing gain on per flow bandwidth will decrease as more payloads are added to the multiplexed payload.

4.2 Packet loss rate

The expected loss rate in a tunnel will affect the decision of which method is used to indicate state changes.

In cases where the loss rate is relatively low (<5%) the "N mode" should be sufficient to ensure delivery of state changes (described in draft-ietf-avt-crtp-enhance-00.txt). The optimal value of N will vary depending on the loss rate in the tunnel. A value of N=2 will protect against the loss of a single packet within a compressed session. A value of N=3 will protect against the loss of two packets in a row.
within a compressed session and so on. Assuming a random distribution of loss within a single compressed session and tunnel loss rates < 1%, the probability of a loss of two or more packets in a row in a compressed session should be < .01%. For networks with a tunnel loss rate of 1% or less, N=2 should be sufficient to ensure reliable transmission of compression state changes.

The other mechanism described in draft-ietf-avt-crtp-enhance-00.txt, "ACK mode", is not described in this document. "ACK mode" is mostly applicable to very lossy networks (>5%).

4.3 Changing headers

There are two fields in the RTP/UDP/IP header whose deltas are likely to change fairly often. These fields are the RTP Time Stamp and the Identification field in the IP header.

4.3.1 Voice Activity Detection

For voice applications, voice activity detection will cause an unpredictable gap in the RTP Time Stamp field whenever a new talk spurt begins. This gap in the RTP Time Stamp field will cause a change in the CRTP Time Stamp delta field. To allow all the compressed state information for the RTP Time Stamp field to be updated in the event of packet loss, the new RTP timestamp needs to be delivered at the beginning of a talk spurt. The CU* compressed header format is used to deliver the absolute RTP timestamp. N mode is used to ensure these changes are transmitted from compressor to decompressor.

4.3.2 IPID

Depending on the implementation of the source node of an RTP flow, the Identification field in the IP header (IPID) can change randomly from packet to packet within a flow.

An IP transmitter typically increments the IPID field by a constant value from packet to packet, but the delta IPID between two packets within an RTP flow may actually change by any value. When compressing an RTP stream where the IPID is not incrementing by a constant value, the best way to maintain context synchronization is to use the CU* header format and include the absolute IPID in each compressed packet.

4.4 Bandwidth calculation formula

The formula below uses the factors that have been described above to come up with a model for per flow bandwidth usage.

The following variables are defined:

SOV-TCRTP = The per sample overhead of CRTP and the multiplexed PPP header in bytes. This value does not include additional overhead for updating IPID or the RTP Time Stamp fields. The value assumes the use of the COMPRESSED-RTP payload type. It consists of 1 byte for the CRTP context ID, 1 byte for COMPRESSED-RTP flags, 2 bytes for the UDP checksum, 1 byte for PPP protocol ID, and 1 byte for the multiplexed PPP length field. This gives a total of 6 bytes.

POV-TCRTP = The per packet overhead of tunneled CRTP in bytes. This is the overhead for the tunnel header and the multiplexed PPP payload type. This value is 20 bytes for the IP header, 1 byte for the L2TPHC header, and 2 bytes for the multiplexed PPP protocol ID. Additional
overhead needed to be added for layer 2 encapsulation. For HDLC, the layer 2 overhead would be 7 bytes. This gives a total per packet overhead of 30 bytes.

VAD-LENGTH = The average length of a talk spurt for voice streams with voice activity detection enabled. This is typically around 1500 msec. Expressed in milliseconds.

SOV-TSTAMP = The additional per sample overhead of the CU* header that includes the absolute time stamp field. This value includes 1 byte for the extra flags field in the CU* header and 2 bytes for the absolute time stamp for a total of 3 bytes.

SOV-IPID = The additional per sample overhead of the CU* header that includes the absolute IPID field. This value includes 2 bytes for the absolute IPID. This value also includes 1 byte for the extra flags field in the CU* header. The total is 3 bytes.

IPID-RATIO = The frequency that IPID need to be updated. This value is 0 or 1. If IPID is changing randomly and always needs to be updated, then IPID-RATIO will be 1. If IPID is changing by a constant amount between samples of a flow, then IPID-RATIO will be 0.

N = The value of N for N mode. This is the number of times an update field will be repeated in CRTP headers to increase the delivery rate between the compressor and decompressor. For this example, we will assume N=2.

PAYLOAD-SIZE = The size of the voice payload in bytes.

MUX-SIZE = The number of PPP payloads to be multiplexed into one multiplexed PPP payload.

SAMPLE-PERIOD = The average sample period of all calls in the multiplex. In msec.

BANDWIDTH = The average amount of bandwidth used per call. In kbits/sec.

SOV-TOTAL = The total amount of per sample overhead associated with tunneled CRTP. It includes the per sample overhead of CRTP and PPP, timestamp update overhead, and IPID update overhead.

SOV-TOTAL = SOV-TCRTP + SOV-TSTAMP * (N * SAMPLE-PERIOD / VAD-LENGTH) + SOV-IPID * (N * IPID-RATIO)

BANDWIDTH = ((PAYLOAD-SIZE + SOV-TOTAL + (POV-TCRTP / MUX-SIZE)) * 8) / SAMPLE-PERIOD

To create an example using the above formulas, we will assume the following usage scenario. Compressed voice streams using G.729 compression with a 20 msec packetization period. In this scenario, VAD is enabled and the average talk spurt length is 1500 msec. The IPID field is changing randomly between payloads of streams. There is enough traffic in the tunnel to allow 3 payloads to be multiplexed in a multiplexed PPP payload. The following values apply:

SAMPLE-PERIOD = 20 msec
VAD-LENGTH = 1500 msec
IPID-RATIO = 1
PAYLOAD-SIZE = 20 bytes
MUX-SIZE = 3
For this example, per call bandwidth is 16 kbits/sec. Non tunneled CRTP using the same factors as above yields 12.8 kbits/sec.

The effect of IPID can have a large effect on per call bandwidth. If the above example is recalculated using an IPID-RATIO of 0, then the per call bandwidth is reduced to 14.4 kbits/sec. Non tunneled CRTP using these same factors yields 12 kbits/call.

4.5 TCRTP Efficiency for Voice over IP over ATM

Layer 2 encapsulation can have a large effect on the amount of bandwidth used with TCRTP encapsulation. Multiplexing gain has a dramatic effect on bandwidth when ATM encapsulation is used. IP transport over AAL-5 causes a quantizing effect to bandwidth utilization. This is because packet sizes will always be in multiples of ATM cell sizes, due to the small sample sizes associated with voice with low bit rate coders.

For example, the sample size for a G.729 coder using 10 msec sample rate is 10 bytes. This is much smaller than the payload size of an ATM cell (48 bytes). When standard IP header compression schemes (CRTP) are applied in an ATM environment, the result is VOIP packets that are small. However, AAL-5 encapsulation and the resulting cell padding cause the minimum PDU size to be one ATM cell.

Instead of wasting this padding, the multiplexing of TCRTP allows this previously wasted space in the ATM cell to contain useful data. This is one of the main reasons why multiplexing has such a large effect on bandwidth utilization with Voice over IP over ATM.

4.6 TCRTP Efficiency for Voice over IP over non-ATM networks

When TCRTP is used with other layer 2 encapsulations that do not have a minimum PDU size, the benefits of multiplexing is not as great. Depending upon the exact overhead of the layer 2 encapsulation, the benefits of VOIP multiplexing might be slightly better or worse than link-by-link CRTP header compression. The per sample overhead of CRTP tunneling is either 4 or 6 bytes. If classical CRTP plus layer 2 overhead is greater than this amount, VOIP multiplexing will end up having lower bandwidth than classical CRTP when the outer IP header is amortized over a large number of samples.

The breakeven point in samples can be determined by the following formula:

POV-L2 * MUX-SIZE >= POV-L2 + POV-TUNNEL + POV-PPPMUX + SOV-PPPMUX * MUX-SIZE

Where:
- **POV-L2** = Layer 2 packet overhead: 7 bytes for HDLC encapsulation
- **POV-TUNNEL** = Packet overhead due to tunneling: 20 bytes IP header, 1 byte L2TPHC header: 21 bytes
- **POV-PPPMUX** = Packet overhead for the multiplexed PPP protocol ID: 2 bytes
- **SOV-PPPMUX** = Per sample overhead of PPPMUX that includes the sample length and sometimes the CRTP protocol ID: 1-3 bytes

For HDLC, the breakeven point is when MUX-SIZE = 6.
5. Example implementation of TCRTP

This section describes an example implementation of TCRTP. Implementations of TCRTP may be done in many ways as long as the requirements of the associated RFCs are met.

Here is the path an RTP packet takes in this implementation:

```
|          Application          |   |              RTP              |   |              UDP              |   |              IP               |   |
|              Application       |   |              RTP              |   |              UDP              |   |              IP               |   |
|     IP forwarding             |   |     IP forwarding             |   |     IP forwarding             |   |
|            CRTP              |   |              PPPMUX            |   |              L2TP              |   |              IP               |
|            PPP               |   |              PPP              |   |              L2TP              |
|+---+---+---+---+---+---+---+---+|   |+---+---+---+---+---+---+---+---+|
|            L2TP              |   |              IP               |
|+---+---+---+---+---+---+---+---+|   |+---+---+---+---+---+---+---+---+|
|            Layer 2            |   |              Phys             |
|+---+---+---+---+---+---+---+---+|   |+---+---+---+---+---+---+---+---+|
|            Phys              |
|+---+---+---+---+---+---+---+---+|
```

A protocol stack is configured to create an L2TP tunnel interface to a destination host. The tunnel is configured to negotiate NCP IPCP with CRTP header compression and PPPMUX. IP forwarding is configured to route packets with the same destination address of the previously configured L2TP tunnel interface to that tunnel interface. To ensure that other traffic destined to the same IP address is not routed to this tunnel interface, the forwarding module should be configured to examine additional information in the IP packet. The destination UDP port number is an example of the additional information that may be used to select the L2TP tunnel interface.

The transmitting application gathers the RTP data and formats an RTP packet. Lower level application layers add UDP and IP headers to form a complete IP packet.

The RTP packets are routed to the tunnel interface where they are compressed, multiplexed, and tunneled to the destination host.
The operation of the receiving node is the same as the transmitting node in reverse.

6. Acknowledgements

The authors would like to thank the authors of RFC2508, Stephen Casner and Van Jacobson, and the authors of RFC2507, Mikael Degermark, Bjorn Nordgren, and Stephen Pink.

The authors would also like to thank Dana Blair, Alex Tweedley, Paddy Ruddy, Francois Le Faucheur, Tim Gleeson, Matt Madison, Hussein Salama, Mallik Tatipamula, Mike Thomas, Mark Townsley, Andrew Valencia, Herb Wildfeuer, J Martin Borden, John Geevarghese, and Shouu Yiu.

7. References


8. Authors’ Addresses

Bruce Thompson
170 West Tasman Drive
San Jose, CA  95134-1706
United States of America

Phone: +1 408 527 0446
Email: brucet@cisco.com

Tmima Koren
170 West Tasman Drive
San Jose, CA  95134-1706
United States of America

Phone: +1 408 527 6169
9. Full Copyright Statement

Copyright (C) The Internet Society (2000). All Rights Reserved.

This document and translations of it may be copied and furnished to
others, and derivative works that comment on or otherwise explain it
or assist in its implementation may be prepared, copied, published
and distributed, in whole or in part, without restriction of any
kind, provided that the above copyright notice and this paragraph
are included on all such copies and derivative works. However, this
document itself may not be modified in any way, such as by removing
the copyright notice or references to the Internet Society or other
Internet organizations, except as needed for the purpose of
developing Internet standards in which case the procedures for
copyrights defined in the Internet Standards process must be
followed, or as required to translate it into languages other than
English.

The limited permissions granted above are perpetual and will not be
revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an
"AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING
TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING
BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION
HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF
MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.