Network Working Group
Request for Comments: 4170
BCP: 110
Category: Best Current Practice

B. Thompson
T. Koren
D. Wing
Cisco Systems

November 2005

Tunneling Multiplexed Compressed RTP (TCRTP)

Status of This Memo

This document specifies an Internet Best Current Practices for the Internet Community, and requests discussion and suggestions for improvements. Distribution of this memo is unlimited.

Copyright Notice

Abstract

This document describes a method to improve the bandwidth utilization of RTP streams over network paths that carry multiple Real-time Transport Protocol (RTP) streams in parallel between two endpoints, as in voice trunking. The method combines standard protocols that provide compression, multiplexing, and tunneling over a network path for the purpose of reducing the bandwidth used when multiple RTP streams are carried over that path.

Table of Contents

1. Introduction

This document describes a way to combine existing protocols for compression, multiplexing, and tunneling to save bandwidth for some RTP applications.

1.1. Is Bandwidth Costly?

On certain links, such as customer access links, the cost of bandwidth is widely acknowledged to be a significant concern. protocols such as CRTP (Compressed RTP, [CRTP]) are well suited to help bandwidth inefficiencies of protocols such as VoIP over these links.

Unacknowledged by many, however, is the cost of long-distance WAN links. While some voice-over-packet technologies such as Voice over ATM (VoAAL2, [I.363.2]) and Voice over MPLS provide bandwidth efficiencies (because both technologies lack IP, UDP, and RTP headers), neither VoATM nor VoMPLS provide direct access to voice-over-packet services available to Voice over IP. Thus, goals of WAN link cost reduction are met at the expense of lost interconnection opportunities to other networks.

TCRTP solves the VoIP bandwidth discrepancy, especially for large, voice-trunking applications.

1.2. Overview of Protocols

Header compression is accomplished using Enhanced CRTP (ECRTP, [ECRTP]). ECRTP is an enhancement to classical CRTP [CRTP] that works better over long delay links, such as the end-to-end tunneling links described in this document. This header compression reduces the IP, UDP, and RTP headers.

Multiplexing is accomplished using PPP Multiplexing [PPP-MUX].

Tunneling PPP is accomplished by using L2TP [L2TPv3].

CRTP operates link-by-link; that is, to achieve compression over multiple router hops, CRTP must be employed twice on each router -- once on ingress, again on egress. In contrast, TCRTP described in this document does not require any additional per-router processing to achieve header compression. Instead, headers are compressed end-to-end, saving bandwidth on all intermediate links.

1.3. Document Focus

This document is primarily concerned with bandwidth savings for Voice over IP (VoIP) applications over high-delay networks. However, the combinations of protocols described in this document can be used to provide similar bandwidth savings for other RTP applications such as video, and bandwidth savings are included for a sample video application.

1.4. Choice of Enhanced CRTP

CRTP [CRTP] describes the use of RTP header compression on an unspecified link layer transport, but typically PPP is used. For CRTP to compress headers, it must be implemented on each PPP link. A lot of context is required to successfully run CRTP, and memory and processing requirements are high, especially if multiple hops must implement CRTP to save bandwidth on each of the hops. At higher line rates, CRTP's processor consumption becomes prohibitively expensive.

To avoid the per-hop expense of CRTP, a simplistic solution is to use CRTP with L2TP to achieve end-to-end CRTP. However, as described in [ECRTP], CRTP is only suitable for links with low delay and low loss. However, once multiple router hops are involved, CRTP's expectation of low delay and low loss can no longer be met. Further, packets can arrive out of order.

Therefore, this document describes the use of Enhanced CRTP [ECRTP], which supports high delay, both packet loss, and misordering between the compressor and decompressor.

1.5. Reducing TCRTP Overhead

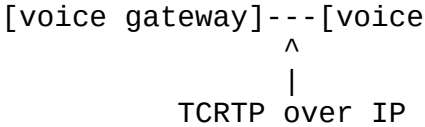
If only one stream is tunneled (L2TP) and compressed (ECRTP), there are little bandwidth savings. Multiplexing is helpful to amortize the overhead of the tunnel header over many RTP payloads. The multiplexing format proposed by this document is PPP multiplexing [PPP-MUX]. See Section 2.3 for details.

2. Protocol Operation and Recommended Extensions

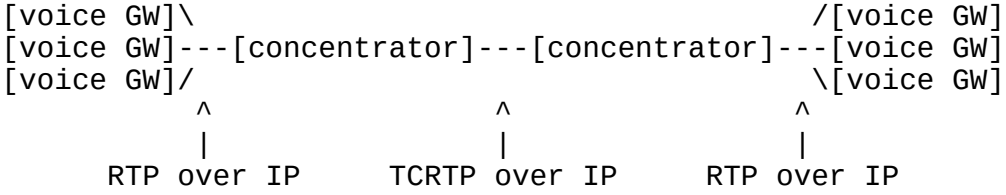
This section describes how to combine three protocols: Enhanced CRTP, PPP Multiplexing, and L2TP Tunneling, to save bandwidth for RTP applications such as Voice over IP.

2.1. Models

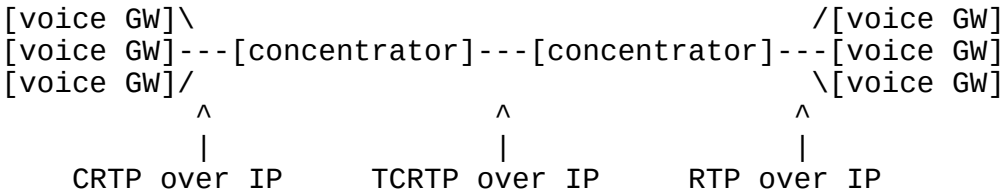
TCRTP can typically be implemented in two ways. The most straightforward is to implement TCRTP in the gateways terminating the RTP streams:



Another way TCRTP can be implemented is with an external concentration device. This device could be placed at strategic places in the network and could dynamically create and destroy TCRTP sessions without the participation of RTP-generating endpoints.



Such a design also allows classical CRTP [CRTP] to be used on links with only a few active flows per link (where TCRTP isn't efficient; see Section 3):



2.2. Header Compression: ECRTP

As described in [ECRTP], classical CRTP [CRTP] is not suitable over long-delay WAN links commonly used when tunneling, as proposed by this document. Thus, ECRTP should be used instead of CRTP.

2.2.1. Synchronizing ECRTP States

When the compressor receives an RTP packet that has an unpredicted change in the RTP header, the compressor should send a COMPRESSED_UDP packet (described in [ECRTP]) to synchronize the ECRTP decompressor state. The COMPRESSED_UDP packet updates the RTP context in the decompressor.

To ensure delivery of updates of context variables, COMPRESSED_UDP packets should be delivered using the robust operation described in [ECRTP].

Because the "twice" algorithm described in [ECRTP] relies on UDP checksums, the IP stack on the RTP transmitter should transmit UDP checksums. If UDP checksums are not used, the ECRTP compressor should use the CRTP Headers checksum described in [ECRTP].

2.2.2. Out-of-Order Packets

Tunneled transport does not guarantee ordered delivery of packets. Therefore, the ECRTP decompressor must operate correctly in the presence of out of order packets.

The order of packets for RTP is determined by the RTP sequence number. To add robustness in case of packet loss or packet reordering, ECRTP sends short deltas together with the full value when updating context variables, and repeats the updates in N packets, where N is an engineered constant tuned to the kind of pipe ECRTP is used for.

By contrast, [ROHC] compresses out the sequence number and another layer is necessary for [ROHC] to handle out-of-order delivery of packets over a tunnel [REORDER].

2.3. Multiplexing: PPP Multiplexing

Both CRTP and ECRTP require a layer two protocol that allows identifying different protocols. [PPP] is suited for this.

When CRTP is used inside of a tunnel, the header compression associated with CRTP will reduce the size of the IP, UDP, and IP headers of the IP packet carried in the tunnel. However, the tunnel itself has overhead due to its IP header and the tunnel header (the information necessary to identify the tunneled payload). One way to reduce the overhead of the IP header and tunnel header is to multiplex multiple RTP payloads in a single tunneled packet.

[PPP-MUX] describes an encapsulation that combines multiple PPP payloads into one multiplexed payload. PPP multiplexing allows any supported PPP payload type to be multiplexed. This multiplexed frame is then carried as a single PPPMUX payload in the IP tunnel. This allows multiple RTP payloads to be carried in a single IP tunnel packet and allows the overhead of the uncompressed IP and tunnel headers to be amortized over multiple RTP payloads.

During PPP establishment of the TCRTP tunnel, only LCP and IPCP (for header compression) are required -- IP addresses do not need to be negotiated, nor is authentication necessary. See Section 4.1 for details.

2.3.1. PPP Multiplex Transmitter Modifications for Tunneling

Section 1.2 of [PPP-MUX] describes an example transmitter procedure that can be used to implement a PPP Multiplex transmitter. The transmission procedure described in this section includes a parameter MAX-SF-LEN that is used to limit the maximum size of a PPP Multiplex frame.

There are two reasons for limiting the size of a PPP Multiplex frame. First, a PPPMUX frame should never exceed the Maximum Receive Unit (MRU) of a physical link. Second, when a PPP session and its associated flow control are bound to a physical link, the MAX-SF-LEN parameter forms an upper limit on the amount of time a multiplex packet can be held before being transmitted. When flow control for the PPP Multiplex transmitter is bound to a physical link, the clock rate of the physical link can be used to pull frames from the PPP Multiplex transmitter.

This type of flow control limits the maximum amount of time a PPP multiplex frame can be held before being transmitted to MAX-SF-LEN / Link Speed.

Tunnel interfaces are typically not bound to physical interfaces. Because of this, a tunnel interface has no well-known transmission rate associated with it. This means that flow control in the PPPMUX transmitter cannot rely on the clock of a physical link to pull frames from the multiplex transmitter. Instead, a timer must be used to limit the amount of time a PPPMUX frame can be held before being transmitted. The timer along with the MAX-SF-LEN parameter should be used to limit the amount of time a PPPMUX frame is held before being transmitted.

The following extensions to the PPPMUX transmitter logic should be made for use with tunnels. The flow control logic of the PPP transmitter should be modified to collect incoming payloads until one of two events has occurred:

(1) a specific number of octets, MAX-SF-LEN, has arrived at the multiplexer, or

(2) a timer, called T, has expired.

When either condition is satisfied, the multiplexed PPP payload is transmitted.

The purpose of MAX-SF-LEN is to ensure that a PPPMUX payload does not exceed the MTU size of any of the possible physical links that the tunnel can be associated with. The value of MAX-SF-LEN should be less than or equal to the minimum of MRU-2 (maximum size of length field) and 16,383 (14 bits) for all possible physical interfaces that the tunnel may be associated with.

The timer T provides an upper delay bound for tunnel interfaces. Timer T is reset whenever a multiplexed payload is sent to the next encapsulation layer. The behavior of this timer is similar to AAL2's Timer_CU described in [I.363.2]. Each PPPMUX transmitter should have its own Timer T.

The optimal values for T will vary depending upon the rate at which payloads are expected to arrive at the multiplexer and the delay budget for the multiplexing function. For voice applications, the value of T would typically be 5-10 milliseconds.

2.3.2. Tunneling Inefficiencies

To get reasonable bandwidth efficiency using multiplexing within an L2TP tunnel, multiple RTP streams should 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 ECRTP 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 the desired level depends on the size of the L2TP header. A smaller L2TP header will result in fewer simultaneous RTP sessions being required to produce bandwidth efficiencies similar to CRTP.

2.4. Tunneling: L2TP

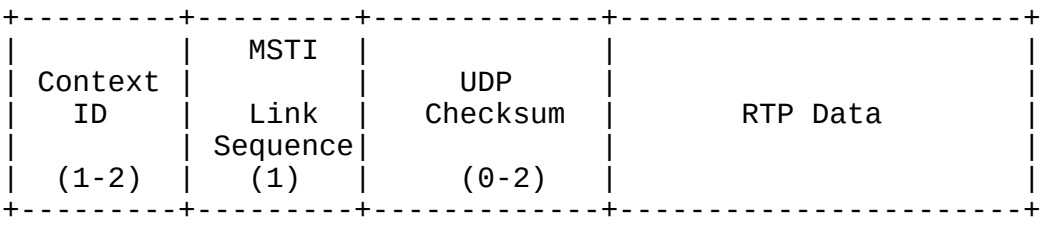
L2TP tunnels should be used to tunnel the ECRTP payloads end to end. L2TP includes methods for tunneling messages used in PPP session establishment, such as NCP. This allows [IPCP-HC] to negotiate ECRTP compression/decompression parameters.

2.4.1. Tunneling and DiffServ

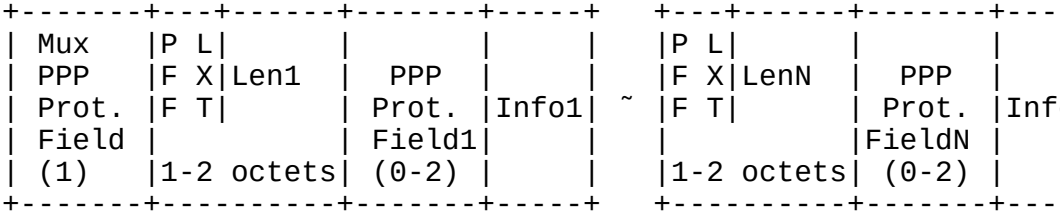
RTP streams may be marked with Expedited Forwarding (EF) bits, as described in [EF-PHB]. When such a packet is tunneled, the tunnel header must also be marked for the same EF bits, as required by [EF-PHB]. It is important to not mix EF and non-EF traffic in the same EF-marked multiplexed tunnel.

2.5. Encapsulation Formats

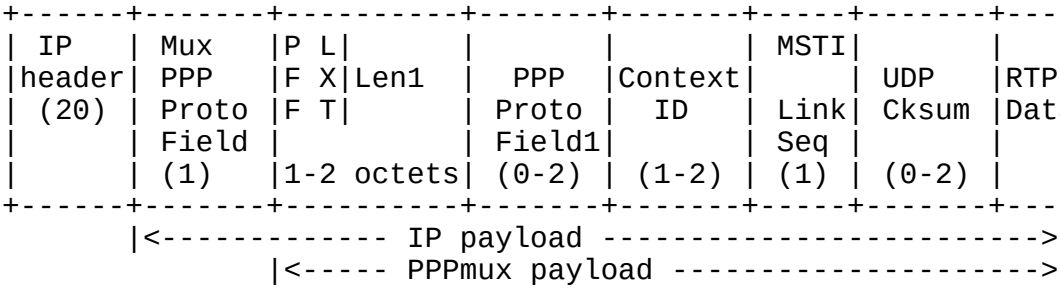
The packet format for an RTP packet, compressed with RTP header compression as defined in ECRTP, is:



The packet format of a multiplexed PPP packet as defined by [PPP-MUX] is:



The combined format used for TCRTP with a single payload is all of the above packets concatenated. Here is an example with one payload:



If the tunnel contains multiplexed traffic, multiple "PPPMux payload"s are transmitted in one IP packet.

3. Bandwidth Efficiency

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

3.1. 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 are combined in one multiplexed payload. This will increase multiplexing gain.

3.2. Packet Loss Rate

Loss of a multiplexed packet causes packet loss for all of the flows within the multiplexed packet.

When the expected loss rate in a tunnel is relatively low (less than perhaps 5%), the robust operation (described in [ECRTP]) should be sufficient to ensure delivery of state changes. This robust operation is characterized by a parameter N, which means that the probability of more than N adjacent packets getting lost on the tunnel is small.

A value of N=1 will protect against the loss of a single packet within a compressed session, at the expense of bandwidth. A value of N=2 will protect against the loss of two packets in a row within a compressed session and so on. Higher values of N have higher bandwidth penalties.

The optimal value of N will depend on the loss rate in the tunnel. If the loss rate is high (above perhaps 5%), more advanced techniques must be employed. Those techniques are beyond the scope of this document.

3.3. Bandwidth Calculation for Voice and Video Applications

The following formula uses the factors described above to model per-flow bandwidth usage for both voice and video applications. These variables are defined:

SOV-TCRTP, unit: octet. Per-payload overhead of ECRTP and the multiplexed PPP header. This value does not include additional overhead for updating IP ID or the RTP Time Stamp fields (see [ECRTP] for details on IP ID). The value assumes the use of the COMPRESSED_RTP payload type. It consists of 1 octet for the ECRTP context ID, 1 octet for COMPRESSED_RTP flags, 2 octets for the UDP checksum, 1 octet for PPP protocol ID, and 1 octet for the multiplexed PPP length field. The total is 6 octets.

POV-TCRTP, unit: octet. Per-packet overhead of tunneled ECRTP. This is the overhead for the tunnel header and the multiplexed PPP payload type. This value is 20 octets for the IP header, 4 octets for the L2TPv3 header and 1 octet for the multiplexed PPP protocol ID. The total is 25 octets.

TRANSMIT-LENGTH, unit: milliseconds. The average duration of a transmission (such as a talk spurt for voice streams).

SOV-TSTAMP, unit: octet. Additional per-payload overhead of the COMPRESSED_UDP header that includes the absolute time stamp field. This value includes 1 octet for the extra flags field in the COMPRESSED_UDP header and 4 octets for the absolute time stamp, for a total of 5 octets.

SOV-IPID, unit: octet. Additional per-payload overhead of the COMPRESSED_UDP header that includes the absolute IPID field. This value includes 2 octets for the absolute IPID. This value also includes 1 octet for the extra flags field in the COMPRESSED_UDP header. The total is 3 octets.

IPID-RATIO, unit: integer values 0 or 1. Indicates the frequency at which IPID will be updated by the compressor. If IPID is changing randomly and thus always needs to be updated, then the value is 1. If IPID is changing by a fixed constant amount between payloads of a flow, then IPID-RATIO will be 0. The value of this variable does not consider the IPID value at the beginning of a voice or video transmission, as that is considered by the variable TRANSMIT-LENGTH. The implementation of the sending IP stack and RTP application controls this behavior. See Section 1.1.

NREP, unit: integer (usually a number between 1 and 3). This is the number of times an update field will be repeated in ECRTP headers to increase the delivery rate between the compressor and decompressor. For this example, we will assume NREP=2.

PAYLOAD-SIZE, unit: octets. The size of the RTP payload in octets.

MUX-SIZE, unit: count. The number of PPP payloads multiplexed into one multiplexed PPP payload.

SAMPLE-PERIOD, unit: milliseconds. The average delay between transmissions of voice or video payloads for each flow in the multiplex. For example, in voice applications the value of this variable would be 10ms if all calls have a 10ms sample period.

The formula is:

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

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

The results are:

BANDWIDTH, unit: kilobits per second. The average amount of bandwidth used per voice or video flow.

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

3.3.1. Voice Bandwidth Calculation Example

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

```
SAMPLE-PERIOD      = 20 milliseconds
TRANSMIT-LENGTH    = 1500 milliseconds
IPID-RATIO         = 1
PAYLOAD-SIZE       = 20 octets
MUX-SIZE           = 3
```

For this example, per call bandwidth is 16.4 kbits/sec. Classical CRTP over a single HDLC link using the same factors as above yields 12.4 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 13.8 kbits/sec. Classical CRTP over a single HDLC link, using these same factors, yields 11.2 kbits/call.

3.3.2. Voice Bandwidth Comparison Table

The bandwidth values are as follows when using 5 simultaneous calls, no voice activity detection (VAD), G.729 with 20ms packetization interval, and not considering RTCP overhead:

```
Normal VoIP over PPP:            124 kbps
with classical CRTP on a link:    50 kbps (savings: 59%)
with TCRTP over PPP:              62 kbps (savings: 50%)
with TCRTP over AAL5:             85 kbps (savings: 31%)
```

3.3.3. Video Bandwidth Calculation Example

Since TCRTP can be used to save bandwidth on any type of RTP encapsulated flow, it can be used to save bandwidth for video applications. This section documents an example of TCRTP-based bandwidth savings for MPEG-2 encoded video.

To create an example for a video application using the above formulas, we will assume the following usage scenario. RTP encapsulation of MPEG System and Transport Streams is performed as described in RFC 2250. Frames for MPEG-2 encoded video are sent continuously, so the TRANSMIT-LENGTH variable in the bandwidth formula is essentially infinite. The IPID field is changing randomly between payloads of streams. There is enough traffic in the tunnel to allow 3 multiplexed payloads. The following values apply:

```
SAMPLE-PERIOD      = 2.8 milliseconds
TRANSMIT-LENGTH    = infinite
IPID-RATIO         = 1
PAYLOAD-SIZE       = 1316 octets
MUX-SIZE           = 3
```

For this example, per flow bandwidth is 3.8 Mbits/sec. MPEG video with no header compression, using the same factors as above, yields 3.9 Mbits/sec. While TCRTP does provide some bandwidth savings for video, the ratio of transmission headers to payload is so small that the bandwidth savings are insignificant.

3.3.4. TCRTP over ATM

IP transport over AAL5 causes a quantizing effect on bandwidth utilization due to the packets always being multiples of ATM cells.

For example, the payload size for G.729 using 10 millisecond packetization intervals is 10 octets. This is much smaller than the payload size of an ATM cell (48 octets). When classical CRTP [CRTP] is used on a link-by-link basis, the IP overhead to payload ratio is quite good. However, AAL5 encapsulation and its cell padding always force the minimum payload size to be one ATM cell, which results in poor bandwidth utilization.

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.

This multiplexing efficiency of TCRTP is similar to AAL2 sub-cell multiplexing described in [I.363.2]. Unlike AAL2 sub-cell multiplexing, however, TCRTP's multiplexing efficiency isn't limited to only ATM networks.

3.3.5. TCRTP over Non-ATM Networks

When TCRTP is used with other layer 2 encapsulations that do not have a minimum PDU size, the benefit of multiplexing is not as great.

Depending upon the exact overhead of the layer 2 encapsulation, the benefit of multiplexing might be slightly better or worse than link-by-link CRTP header compression. The per-payload overhead of CRTP tunneling is either 4 or 6 octets. If classical CRTP plus layer 2 overhead is greater than this amount, TCRTP multiplexing will consume less bandwidth than classical CRTP when the outer IP header is amortized over a large number of payloads.

The payload breakeven point can be determined by the following formula:

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

Where:

POV-L2, unit: octet. Layer 2 packet overhead: 5 octets for HDLC encapsulation

POV-TUNNEL, unit: octet. Packet overhead due to tunneling: 24 octets IP header and L2TPv3 header

POV-PPPMUX, unit: octet. Packet overhead for the multiplexed PPP protocol ID: 1 octet

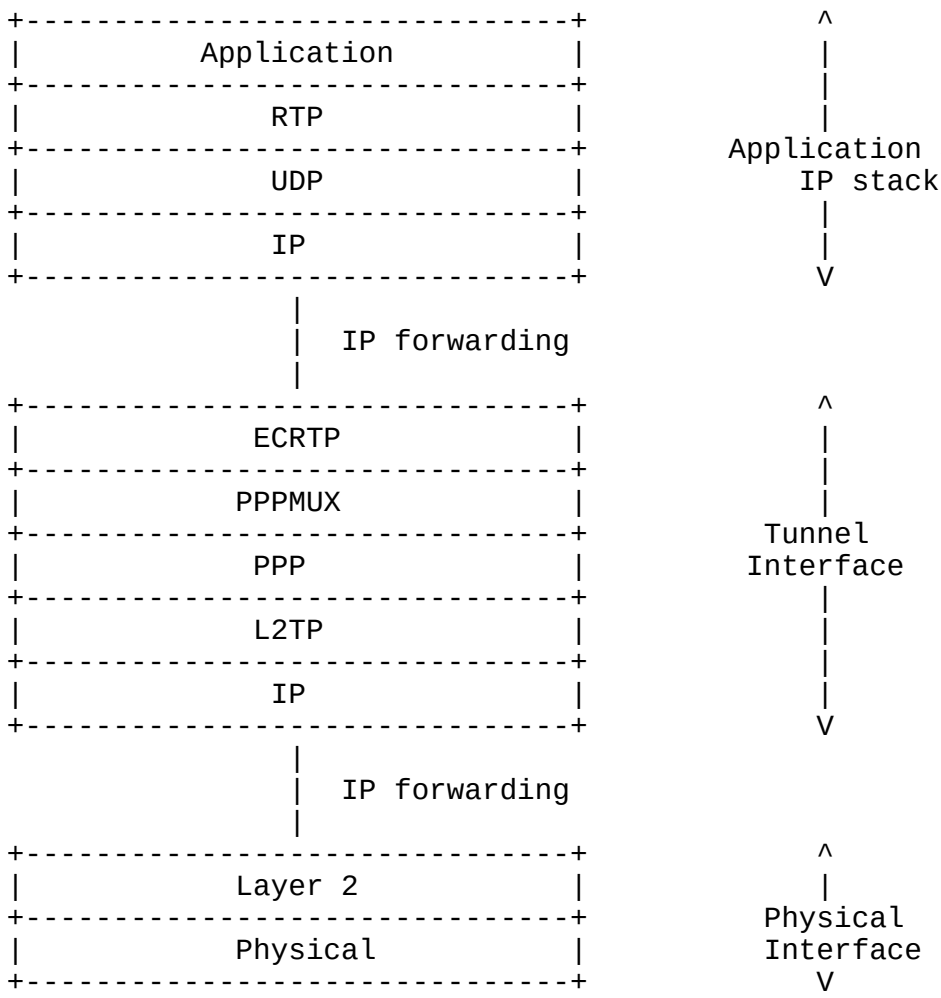
SOV-PPPMUX, unit: octet. Per-payload overhead of PPPMUX, which is comprised of the payload length field and the ECRTP protocol ID. The value of SOV-PPPMUX is typically 1, 2, or 3.

If using HDLC as the layer 2 protocol, the breakeven point (using the above formula) is when MUX-SIZE = 7. Thus 7 voice or video flows need to be multiplexed to make TCRTP as bandwidth-efficient as link-by-link CRTP compression.

4. 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:



A protocol stack is configured to create an L2TP tunnel interface to a destination host. The tunnel is configured to negotiate the PPP connection (using NCP IPCP) with ECRTP header compression and PPPMUX. IP forwarding is configured to route RTP packets to this tunnel. The destination UDP port number could distinguish RTP packets from non-RTP packets.

The transmitting application gathers the RTP data from one source, 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 headers are compressed, payloads are multiplexed, and then the packets are tunneled to the destination host.

The operation of the receiving node is the same as the transmitting node in reverse.

4.1. Suggested PPP and L2TP Negotiation for TCRTP

This section describes the necessary PPP and LT2P negotiations necessary for establishing a PPP connection and L2TP tunnel with L2TP header compression. The negotiation is between two peers: Peer1 and Peer2.

4.2. PPP Negotiation TCRTP

The Point-to-Point Protocol is described in [PPP].

4.2.1. LCP Negotiation

Link Control Processing (LCP) is described in [PPP].

4.2.1.1. Link Establishment

```
            Peer1                        Peer2
            -----                        -----
   Configure-Request (no options) ->
                                  <- Configure-Ack
                                  <- Configure-Request (no options)
   Configure-Ack                  ->
```

4.2.1.2. Link Tear Down

```
      Terminate-Request              ->
                                      <- Terminate-Ack
```

4.2.2. IPCP Negotiation

The protocol exchange here is described in [IPHCOMP], [PPP], and [ECRTP].

```
          Peer1                        Peer2
          -----                        -----
  Configure-Request              ->
    Options:
    IP-Compression-Protocol
      Use protocol 0x61
      and sub-parameters
      as described in
      [IPCP-HC] and [ECRTP]
                                  <- Configure-Ack
                                  <- Configure-Request
                                       Options:
                                       IP-Compression-Protocol
                                         Use protocol 0x61
                                         and sub-parameters
                                         as described in
                                         [IPCP-HC] and [ECRTP]
  Configure-Ack                  ->
```

4.3. L2TP Negotiation

L2TP is described in [L2TPv3].

4.3.1. Tunnel Establishment

```
            Peer1                          Peer2
            -----                          -----
     SCCRQ                          ->
       Mandatory AVP's:
       Message Type
       Protocol Version
       Host Name
       Framing Capabilities
       Assigned Tunnel ID
                                    <- SCCRP
                                         Mandatory AVP's:
                                         Message Type
                                         Protocol Version
                                         Host Name
                                         Framing Capabilities
                                         Assigned Tunnel ID
     SCCCN                          ->
     Mandatory AVP's:
       Message Type
                                    <- ZLB
```

4.3.2. Session Establishment

```
            Peer1                          Peer2
            -----                          -----
     ICRQ                           ->
       Mandatory AVP's:
       Message Type
       Assigned Session ID
       Call Serial Number
                                   <- ICRP
                                         Mandatory AVP's:
                                         Message Type
                                         Assigned Session ID
     ICCN                           ->
       Mandatory AVP's:
       Message Type
       Tx (Connect Speed)
       Framing Type
                                    <- ZLB
```

4.3.3. Tunnel Tear Down

```
            Peer1                          Peer2
            -----                          -----
     StopCCN                        ->
       Mandatory AVP's:
       Message Type
       Assigned Tunnel ID
       Result Code
                                    <- ZLB
```

5. Security Considerations

This document describes a method for combining several existing protocols that implement compression, multiplexing, and tunneling of RTP streams. Attacks on the component technologies of TCRTP include attacks on RTP/RTCP headers and payloads carried within a TCRTP session, attacks on the compressed headers, attacks on the multiplexing layer, or attacks on the tunneling negotiation or transport. The security issues associated individually with each of those component technologies are addressed in their respective specifications, [ECRTP], [PPP-MUX], [L2TPv3], along with the security considerations for RTP itself [RTP].

However, there may be additional security considerations arising from the use of these component technologies together. For example, there may be an increased risk of unintended misdelivery of packets from one stream in the multiplex to another due to a protocol malfunction or data error because the addressing information is more condensed. This is particularly true if the tunnel is transmitted over a link-layer protocol that allows delivery of packets containing bit errors, in combination with a tunnel transport layer option that does not checksum all of the payload.

The opportunity for malicious misdirection may be increased, relative to that for a single RTP stream transported by itself, because addressing information must be unencrypted for the header compression and multiplexing layers to function.

The primary defense against misdelivery is to make the data unusable to unintended recipients through cryptographic techniques. The basic method for encryption provided in the RTP specification [RTP] is not suitable because it encrypts the RTP and RTCP headers along with the payload. However, the RTP specification also allows alternative approaches to be defined in separate profile or payload format specifications wherein only the payload portion of the packet would be encrypted; therefore, header compression may be applied to the encrypted packets. One such profile, [SRTP], provides more

sophisticated and complete methods for encryption and message authentication than the basic approach in [RTP]. Additional methods may be developed in the future. Appropriate cryptographic protection should be incorporated into all TCRTP applications.

6. Acknowledgements

The authors would like to thank the authors of RFC 2508, Stephen Casner and Van Jacobson, and the authors of RFC 2507, 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 Shoou Yiu.

7. References

7.1. Normative References

[PPP-MUX] Pazhyannur, R., Ali, I., and C. Fox, "PPP Multiplexing", RFC 3153, August 2001.

[ECRTP] Koren, T., Casner, S., Geevarghese, J., Thompson, B., and P. Ruddy, "Enhanced Compressed RTP (CRTP) for Links with High Delay, Packet Loss and Reordering", RFC 3545, July 2003.

[CRTP] Casner, S. and V. Jacobson, "Compressing IP/UDP/RTP Headers for Low-Speed Serial Links", RFC 2508, February 1999.

[IPHCOMP] Degermark, M., Nordgren, B., and S. Pink, "IP Header Compression", RFC 2507, February 1999.

[IPCP-HC] Engan, M., Casner, S., Bormann, C., and T. Koren, "IP Header Compression over PPP", RFC 3544, July 2003.

[RTP] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

[L2TPv3] Lau, J., Townsley, M., and I. Goyret, "Layer Two Tunneling Protocol - Version 3 (L2TPv3)", RFC 3931, March 2005.

[I.363.2] ITU-T, "B-ISDN ATM Adaptation layer specification: Type 2 AAL", I.363.2, September 1997.

[EF-PHB] Davie, B., Charny, A., Bennet, J.C., Benson, K., Le Boudec, J., Courtney, W., Davari, S., Firoiu, V., and D. Stiliadis, "An Expedited Forwarding PHB (Per-Hop Behavior)", RFC 3246, March 2002.

[PPP] Simpson, W., "The Point-to-Point Protocol (PPP)", STD 51, RFC 1661, July 1994.

7.2. Informative References

[SRTP] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

[REORDER] G. Pelletier, L. Jonsson, K. Sandlund, "RObust Header Compression (ROHC): ROHC over Channels that can Reorder Packets", Work in Progress, June 2004.

[ROHC] Bormann, C., Burmeister, C., Degermark, M., Fukushima, H., Hannu, H., Jonsson, L-E., Hakenberg, R., Koren, T., Le, K., Liu, Z., Martensson, A., Miyazaki, A., Svanbro, K., Wiebke, T., Yoshimura, T., and H. Zheng, "RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed ", RFC 3095, July 2001.

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
EMail: tmima@cisco.com

Dan Wing
170 West Tasman Drive
San Jose, CA 95134-1706
United States of America

EMail: dwing@cisco.com

Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.