Network Working Group
Internet Draft
<draft-ash-e2e-crtp-hdr-compress-01.txt>
Expiration Date: October 2003

Jerry Ash
Bur Goode
Jim Hand
AT&T

March, 2003

End-to-End VoIP Header Compression Using cRTP

<draft-ash-e2e-crtp-hdr-compress-01.txt>

Status of this Memo

ABSTRACT:

VoIP typically uses the encapsulation voice/RTP/UDP/IP, wherein the packet header is at least 40 bytes, while the voice payload is typically no more than 30 bytes. VoIP header compression can significantly reduce the VoIP overhead through various compression mechanisms. This is important on access links where bandwidth is scarce, and can be important on backbone facilities, especially where costs are high (e.g., some global cross-sections). In this draft we propose to re-use the methods in cRTP to determine the header compression context and to use the cRTP session context ID to route a compressed packet between the ingress and egress routers.

Table of Contents

1. Introduction

Voice over IP (VoIP) typically uses the encapsulation voice/RTP/UDP/IP/, wherein the packet header is at least 40 bytes, while the voice payload is typically no more than 30 bytes. The interest in VoIP header compression is the possibility of significantly reducing the VoIP overhead through various compression mechanisms. The need may be important on access links where bandwidth is more scarce, but it could be important on backbone facilities, especially where costs are high (e.g., some global cross-sections). Typically, VoIP will use voice compression mechanisms (e.g., G.729) in order to conserve bandwidth. With VoIP header compression, significantly more bandwidth could be saved. For example, carrying VoIP headers for the entire voice load of a large domestic network with 300 million or more calls per day could consume on the order of about 20-40 gigabits-per-second on the backbone network for headers alone. This overhead could translate into considerable bandwidth capacity.

In principle, we could use existing header compression techniques, such as those described in [cRTP], to make the transport of the RTP/UDP/IP headers more efficient. [cRTP] header compression is often used on the access links from the CE router to the PE router. However, cRTP header compression must be implemented on a hop-by-hop basis, and does not scale well for large voice traffic loads. To implement 'end-to-end' VoIP header compression, a method of routing the compressed packets end-to-end is needed. In [VoMPLS], it is proposed that the ingress router would apply header compression to the IP packet and use an MPLS label for routing the compressed packet to the egress router, where the full header would be restored. Here we propose to extend [cRTP] in two ways. First, we describe a technique to enable packets compressed with the techniques used in [cRTP] to flow across multiple contiguous compression-enabled links, without requiring a decompression/compression cycle at each transit router. This technique uses the cRTP session context ID (SCID) to route the compressed packet between ingress and egress routers, in a manner analogous to switching MPLS packets based on MPLS labels. A method is described to associate the SCID and the next hop in the routing table, and also to set up a correspondence between the header compression tables at the ingress and egress routers in the session. Second, a means is described to send header-compressed traffic over the type of MPLS VPN infrastructure specified in [MPLS-VPN]. This method encapsulates header-compressed packets over an LSP set up using

RSVP-TE tunnels as described in [RSVP-TE] between provider-edge routers (PE's). The tunnels between PE's act as a logical point-to-point link between PE's over which header-compressed traffic may be sent. The result of these two extensions is a means of extending [cRTP] end-to-end between customer-premises routers through an MPLS network (or other topology) without requiring any compression/decompression cycles on transit routers.

Note that the techniques described in this document enable end-to-end header compression, but they do not require it. Header compression can be enabled on links independently (with an MPLS VPN PE-to-PE association considered a single link). If multiple contiguous links on a path are capable of, and configured for, transporting packets using RFC2508-style header compression, then this document describes a means by which transit routers between these links can avoid performing decompression/compression cycles on most of the compressed packets. On the other hand, if there are one or more transit links that cannot transport packets using header compression, this technique will still work on the remaining parts of the path where there are contiguous links that support header compression.

This technique also enables end-to-end header compression in an MPLS VPN environment without any changes to the existing RFC2508-style header compression in Customer Edge (CE) routers.

In this draft we propose to re-use the methods in cRTP to determine the header compression context and to use the cRTP context ID to route a compressed packet between the ingress and egress router.

We recommend using enhancements to cRTP that would minimize feedback based error recovery using CONTEXT_STATE packets [cRTP-ENHANCE] to make cRTP more tolerant of packet loss, and minimize the need to use the cRTP error recovery mechanism. The reason is that basic cRTP would perform best with very low packet error rates on all hops of the path. Tunneling a cRTP session through multiple IP hops will increase the round trip delay and the chance of packet loss. cRTP contexts are invalidated due to packet loss. The cRTP error recovery mechanism using CONTEXT_STATE packets can compound the problem when long round trip delays are involved. When the cRTP decompressor context state gets out of synch with the compressor, it will drop packets associated with the context until the two states are resynchronized. To resynchronize context state at the two ends, the decompressor transmits the CONTEXT_STATE packet to the compressor, and the compressor transmits a FULL_HEADER packet to the decompressor. If the resynchronization were rare, then the basic cRTP approach would perform well for end-to-end header compression. Otherwise, enhanced cRTP is desirable. The extensions to [cRTP] described here do not preclude the use of the enhancements to [cRTP] described in [cRTP-ENHANCE].

Compressing and decompressing headers at the ingress and egress routers (versus, say, the backbone routers) disperses the header compression

computational load among many routers, and may achieve better scalability.

Section 2 presents the requirements for end-to-end VoIP header compression, and Section 3 presents the proposed protocol extensions.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC2119 [KEY].

2. Overview of End-to-End VoIP Header Compression Using cRTP

Header compression based on [cRTP] must be supported and configured on each link in the path in order for end-to-end header compression to work. If there are any 'holes"'in header-compression support along the path, then the router at the upstream end of the 'hole' must decompress the packet, and the router and the downstream end of the 'hole' must compress the packet again. Note a traversal of an MPLS network is considered a single link.

On IP-only routers and MPLS edge routers, once compression contexts are established, the SCID is used to determine the next hop in the routing table without decompressing the packet header. Header-compressed packets are switched normally using only MPLS labels on MPLS core routers.

The goal of cRTP header compression is to reduce the IP/UDP/RTP headers to two bytes for most packets in the case where no UDP checksums are being sent, or four bytes with checksums. (Note that the UDP checksum is required for cRTP-ENHANCE, so the compressed headers would be four bytes.) In cRTP header compression, the first factor-of-two reduction in header size comes from the observation that half of the bytes in the IP/UDP/RTP headers remain constant over the life of the connection. After sending the uncompressed header template once, these fields may be removed from the compressed headers that follow. The remaining compression comes from the observation that although several fields change in every packet, the difference from packet to packet is often constant and therefore the second-order difference is zero.

By maintaining both the uncompressed header and the first-order differences in the session state shared between the compressor and decompressor, all that must be communicated is an indication that the second-order difference was zero. In that case, the decompressor can reconstruct the original header without any loss of information simply by adding the first-order differences to the saved uncompressed header as each compressed packet is received. The compressed packet carries a small integer, called the session context identifier or SCID, to indicate in which session context that packet should be interpreted. The decompressor can use the SCID to index its table of stored session contexts directly.

The cRTP FULL-HEADER packet is used to establish the SCID routing table in each router in the path as described in the following section. The compressor communicates the SCID on each compressed packet to the decompressor (and to each router in the path). The decompressor uses the information in the cRTP FULL-HEADER packet to reconstruct the packet header information.

3. Protocol Extensions

We consider two scenarios in this section, first where no MPLS forwarding is involved in end-to-end header compression path, and second where MPLS forwarding is involved somewhere in the path.

3.1 Routing Compressed Packets with cRTP SCID Switching (No MPLS Forwarding in Path)

We assume that a VoIP flow is routed from R1 --> R2 --> R3 --> R4, where each router in the path supports cRTP header compression.

The ingress router R1 receives a VoIP packet and determines that the next_hop router R2 supports cRTP header compression. As such, router R1 follows the procedures in [cRTP] and sends a FULL_HEADER packet on the link to R2. A FULL_HEADER packet is a packet with an uncompressed header in which:

a. the link layer packet type field is modified to indicate that it is not a normal IP packet
b. the 8-bit or 16-bit SCID and the 4-bit initial sequence number are encoded in the length fields of the packet.

Router R2 processes the FULL_HEADER packet information to determine the next_hop for the packet based on the destination IP address and perhaps other routing information in the FULL_HEADER packet. Router R2 determines if the next_hop router, router R3, supports cRTP header compression, and since it does in this example, router R2 creates an entry in the SCID routing table which associates the SCID with the next_hop. The SCID routing tables contains the following information:

a. Incoming interface
b. Incoming SCID
c. Outgoing interface
d. Outgoing SCID

Router R2 assigns a unique outgoing SCID for this VoIP flow. The entry in the SCID routing table must be tied to the IP routing table entry that was used to route the packet to the outgoing interface. This allows the context routing table entry to be invalidated if the associated IP route were removed.

Router R2 then sends the FULL_HEADER packet to router R3. Router R3 performs the same function to create the SCID routing table entry for

the next_hop to router R4 and sends the FULL_HEADER packet to router R4. Router R4 processes the FULL_HEADER packet in a similar manner, however at this point router R4 determines that the next_hop router does not support cRTP. Router R4 then becomes the decompressor for this SCID session, and stores the FULL_HEADER information, SCID, and sequence number as the context for the flow, as described in [cRTP]. Router R4 does not transmit the FULL_HEADER packet any further.

Router R1 sends compressed packets for this VoIP session, with the associated SCID, to router R2. Router R2 uses the SCID to access the next_hop from its SCID routing table, as does router R3. Router R4 recognizes that it is the decompressor for this SCID, and performs the normal decompressing functions [cRTP]. From router R4, the packet is sent to the normal IP routing function for routing.

As new, compressed packets arrive on the incoming interface for the flow, rather than decompressing the packets and using normal IP routing to route the packets, the router can instead look up the <incoming interface, incoming SCID> pair in the SCID routing table. This lookup will return the outgoing interface and outgoing SCID. The router can then route the packet to the outgoing interface and rewrite the header to contain the new SCID, and possibly update other parts of the header, such as the sequence number.

If compressed packets arrive on the interface, and there is a valid SCID on the incoming interface, but there is no entry for the SCID in the SCID routing table, then the packet is de-compressed, and routed using normal IP routing. If the outgoing interface determined by the normal IP routing process supports [cRTP], then the router may send the packet as a FULL_HEADER packet on the outgoing interface, and then add an entry to the SCID routing table, so that subsequent packets may be switched without a decompression/compression cycle.

If there is a change to the IP routing table that changes the next_hop that would be used for the destination IP address of any contexts in the SCID routing table, then these contexts should be removed from the SCID routing table.

SCID switching is not applied to FULL_HEADER or CONTEXT_STATE packets. FULL_HEADER packets re-initialize the flow associated with an SCID, possibly associating a new flow with an SCID. Therefore, when a FULL_HEADER packet is received on an interface, any entries in the SCID routing table which match the incoming interface and SCID of the FULL_HEADER packet should be removed. Similarly, when a CONTEXT_STATE packet is received on an interface, any entries in the SCID routing table in which the incoming interface of the CONTEXT_STATE packet matches the outgoing interface of the SCID routing table entry, and the SCID(s) in the CONTEXT_STATE packet matches the outgoing SCID of the SCID routing table entry, should be removed from the table.

This procedure does NOT require SCIDs to be the same size on the

incoming and outgoing interfaces. Compliant implementations MUST be able to switch packets between interfaces that use different size SCIDs [i.e. 8-bit vs. 16-bit].

3.2 Routing Compressed Packets with cRTP SCID Switching (MPLS Forwarding in Path)

We assume that a VoIP flow is routed from CE1 --> PE1 --> P --> PE2 --> CE2, where each router in the path supports cRTP header compression.

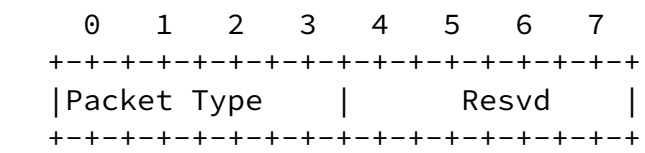
MPLS procedures and objects for end-to-end (CE1-CE2) cRTP-based header compression are given in [VoMPLS]. Here we discuss the application of these procedures to VoIP flows in which the CE1-PE1 and PE2-CE2 legs use cRTP without MPLS, and the PE1-PE2 'leg' uses MPLS procedures.

The basic means of extending [cRTP] over MPLS networks is to add a label stack to header-compressed packets at the PE router, send it over the MPLS network as normal MPLS traffic, and then remove the label stack at PE2. Header compression is transparent to core (P) LSRs.

[cRTP] is specified for point-to-point links, and assumes a single VPN. We need to extend the operation of [cRTP] to MPLS VPNs. The extensions must a) set up LSPs over which header-compressed packets may be sent, and b) handle the fact that [cRTP] assumes point-to-point connectivity. In particular, a decompressor must know the upstream neighbor for each SCID, so that it can send CONTEXT_STATE packets to it. Also, in [cRTP] the compressor assigns SCIDs to flows. Because there is only a single compressor on each link, the compressor owns the SCID space and can trivially guarantee that SCIDs are unique. However, with MPLS, several compressors may be sending to a single decompressor over a link. A method is provided in [VoMPLS] to ensure that the assignment of SCIDs to flows is unique. The VPN associated with each flow must also be identified.

RSVP-TE is used to set up two LSP tunnels, one in each direction, between PE routers, as a logical, full-duplex point-to-point link. [cRTP] is then run over the link. New RSVP-TE objects are proposed in [VoMPLS] to allow PE1 to request a unique SCID(s) from PE2, and for PE2 to communicate the SCID assignment(s) back to PE1. All SCID assignments are associated with a particular LSP.

[cRTP-ENCAP] uses a separate link-layer packet type defined for header compression. Using a separate link-layer packet type for every packet type used in header compression avoids the need to add extra bits to the compression header to identify the packet type. However, this practice does not extend well to MPLS encapsulation conventions [MPLS-ENCAP], in which a separate link-layer packet type translates into a separate LSP for each packet type. So for extending [cRTP] over MPLS VPNs, each packet type defined in [cRTP] MUST have prepended to it a packet type field. This field adds 1 octet to the header, and is encoded as follows (most significant bit is 0):



where:

"Packet Type" is encoded with the following values:

0: Reserved
1: FULL_HEADER
2: COMPRESSED_TCP
3: COMPRESSED_TCP_NODELTA
4: COMPRESSED_NON_TCP
5: COMPRESSED_RTP_8
6: COMPRESSED_RTP_16
7: COMPRESSED_UDP_8
8: COMPRESSED_UDP_16
9: CONTEXT_STATE

"Resvd" is reserved and must be set to 0.

In order to identify the VPN associated with each flow, FULL_HEADER packets sent over LSRs set up for compressed traffic MUST be pre-pended with the VPN ('bottom') label that would be pushed onto the packet's MPLS label stack by the sending PE [MPLS-VPN], if the packet were being sent uncompressed. This VPN label is the VPN identifier. The VPN label must also be added to the context state kept for each flow by the compressor and decompressor. PE2 can use the VPN label to route packets associated with the context, should PE2 need to decompress these packets. Other types of compressed packets sent over MPLS VPNs MUST NOT have the VPN label prepended, since it is not necessary once the FULL_HEADER packet has set up the compression context.

3.2.1 Header Compression Object Formats

A new L3PID (ethertype), XXXX, is defined in [RSVP-TE] for RFC2508 header compression over MPLS LSPs. This is needed to define the type of traffic used in RSVP-TE Label Request Objects.

An SCID_Request object and Header_Compression_Reply object are defined in [VoMPLS]. PE1 creates an LSP to PE2 router by creating an RSVP-TE PATH message that contains:

a. Label_Request object with the L3PID for cRTP over MPLS VPN (XXXX - TBD),
b. an SCID_Request object.

PE1 will receive a RESV message containing a Label object and a Header_Compression_Reply object. PE1 uses the label and SCID to send

compressed traffic to PE2.

Procedures for treatment of these objects are contained in [VoMPLS].

3.2.2 Data Plane Procedures

PE1 and PE2 follow the procedures in [cRTP], including the sending of FULL_HEADER packets, compressed packets, CONTEXT_STATE packets, etc., with some exceptions. These exceptions are:

1. PE1 must associate an outgoing ('top') LSP label as part of the context state for each flow. Each outgoing flow must be associated with an outgoing interface, LSP label and SCID.
2. PE1 does not have an implicit block of 256 or 65536 SCIDs to use to assign to flows. Instead, it has a single SCID or a set of N SCIDs, which it received from PE2 and can assign to flows. Both PE1 and PE2 must keep track of which SCIDs are valid for each LSP carrying compressed traffic.
3. the 'packet type' octet described above, must be prepended to each header.
4. the VPN ('bottom') label of the two-level label stack described in [MPLS-VPN] must be pre-pended to each FULL_HEADER packet. The VPN label is maintained as part of the compression context of each compressed flow. This is needed in case PE2 must decompress and route the packet. It uses the VPN label to determine which routing table to use to route the packet.

In terms of SCID switching (Section 3.1), the primary change required as a result of the extension to use MPLS is that the SCID switching table must be modified to include the outgoing label on the Upstream PE. That is, the SCID switching table must contain the following fields:

a. Incoming interface
b. Incoming SCID
c. Outgoing interface
d. Outgoing MPLS label (if the outgoing interface is MPLS)
e. Outgoing SCID

4. Security Considerations

No additional security risks/requirements are posed.

5. References

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

[cRTP-ENCAP] Engan, M., Casner, S., Bormann, C., "IP Header Compression over PPP", RFC2509, February 1999.

[cRTP-ENHANCE] Koren, T., et. al., "Compressing IP/UDP/RTP Headers on

Links with High Delay, Packet Loss, and Reordering," work in progress.

[KEY] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997.

[MPLS-ENCAP] Rosen, E., Tappan, D., et al, "MPLS Label Stack Encoding", RFC 3032, January 2001.

[MPLS-VPN] Rosen, E., Rekhter, Y., "BGP/MPLS VPNs", RFC 2547, March 1999.

[RSVP-TE] Awduche, D., "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, December 2001

[VoMPLS] Ash, G., Goode, B., Hand, J., "End-to-End VoIP over MPLS Header Compression," work in progress.

6. Authors' Addresses

Jerry Ash
AT&T
Room MT D5-2A01
200 Laurel Avenue
Middletown, NJ 07748, USA
Phone: +1 732-420-4578
Email: gash@att.com

Bur Goode
AT&T
Phone: + 1 203-341-8705
E-mail: bgoode@att.com

Jim Hand
AT&T
Room MT A2-4F36
200 Laurel Avenue
Middletown, NJ 07748, USA
Phone: +1 732-420-6179
E-mail: jameshand@att.com

7. Full Copyright Statement