R.Bonica
WorldCom
D.Tappan
Cisco Systems
D.Gan
Juniper Networks

Internet Draft
Document: draft-bonica-icmp-mpls-02.txt
November 2000

ICMP Extensions for MultiProtocol Label Switching

Status of this Memo

1. Abstract

The current memo documents extensions to ICMP that permit Label Switching Routers to append MPLS header information to ICMP messages. These ICMP extensions support an MPLS aware traceroute application that network operators can use to trace paths through the MPLS user plane.

Although these extensions are not being proposed as Internet Standards, they are documented here because they have been implemented by several vendors and deployed by several operators.

2. Conventions used in this document

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 [RFC-2119].

3. Introduction

Routers and destination hosts use the Internet Control Message Protocol (ICMP) [RFC-792] to convey control information to source hosts. Network operators use this information to diagnose routing problems.

When a router receives an undeliverable IP datagram, it can send an ICMP message to the host that originated the datagram. The ICMP message indicates why the datagram could not be delivered. It also contains the IP header and leading payload octets of the "original datagram".

In this document, the term "original datagram" refers to the datagram to which the ICMP message is a response.

MPLS Label Switching Routers (LSRs) also use ICMP to convey control information to source hosts. Sections 2.3 and 2.4 of [ENCODE] describe the interaction between MPLS and ICMP.

When an LSR receives an undeliverable MPLS encapsulated datagram, it removes the entire MPLS label stack, exposing the previously encapsulated IP datagram. The LSR then submits the IP datagram to a network-forwarding module for error processing. Error processing can include ICMP message generation.

The ICMP message indicates why the original datagram could not be delivered. It also contains the IP header and leading octets of the original datagram.

The ICMP message, however, includes no information regarding the MPLS label stack that encapsulated the original datagram when it arrived at the LSR. This omission is significant because the LSR would have routed the original datagram based upon information contained by the MPLS label stack.

The current memo documents extensions to ICMP that permit an LSR to append MPLS label stack information to ICMP messages. ICMP messages regarding MPLS encapsulated datagrams can include the MPLS label stack, as it arrived at the router that sent the ICMP message. The

ICMP message MUST also include the IP header and leading payload octets of the original datagram.

Network operators will use this information to diagnose routing problems.

4. Motivation

ICMP extensions defined in the current memo support enhancements to TRACEROUTE. The enhanced TRACEROUTE, like older implementations,

indicates which nodes the original datagram visited en route to its ultimate destination. It differs from older implementations in that it also indicates the original datagrams MPLS encapsulation status as it arrived at each node.

Figure 1 contains sample output from an enhanced TRACEROUTE implementation.

```
>Traceroute 166.45.2.74
traceroute to 166.45.2.74, 30 hops max, 40 byte packets
1 166.45.5.1 1.281 ms 1.103 ms 1.096 ms
2 166.45.4.1 1.281 ms 1.103 ms 1.096 ms mplsLabel1=2001
mplsExpBits1=0
3 166.45.3.1 1.281 ms 1.103 ms 1.096 ms mplsLabel1=2002
mplsExpBits1=0
4 166.45.6.1 1.281 ms 1.103 ms 1.096 ms mplsLabel1=2003
mplsExpBits1=0
5 166.45.2.1 1.281 ms 1.103 ms 1.096 ms
6 166.45.2.74 1.281 ms 1.103 ms 1.096 ms
```

Figure 1. Enhanced TRACEROUTE sample output

5. Disclaimer

The current memo does not define the general relationship between ICMP and MPLS. Sections 2.3 and 2.4 of [ENCODE] define this relationship.

Specifically, this document defers to [ENCODE] with respect to the following issues:

- conditions upon which an LSR emits ICMP messages

- handling of ICMP messages bound for hosts that are identified by private addresses

The current memo does not define encapsulation specific TTL manipulation procedures. It defers to Section 10 of [MPLSATM] and Section 5.4 of [MPLSFRAME] in this matter.

When encapsulation specific TTL manipulation procedures defeat the basic TRACEROUTE mechanism, they will also defeat enhanced TRACEROUTE implementations.

The current memo does not address extensions to ICMPv6. These should be addressed in a separate draft.

6. Formal Syntax

This section defines a data structure that an LSR can append to selected ICMP messages. The data structure contains the MPLS label

stack that encapsulated the original datagram when it arrived at the LSR.

The data structure defined herein can be appended to the following ICMP message types:

1) Destination Unreachable
2) Time Exceeded

According to RFC-792, bytes 0 through 19 of any ICMP message contain a header whose format is analogous to that of the IP datagram. Bytes 20 through 23 contain an ICMP message type, code and checksum. Bytes 24 through 27 contain message specific data.

Also according to RFC-792, the final field contained by each of the ICMP message types listed above begins at byte 28. It reflects the IP header and leading 64 bits of the original datagram. [RFC-1812] recommends that this final field be extended to include as much of the original datagram as possible.

When an LSR appends the data structure defined herein to an ICMP message, the final field of the ICMP message body MUST contain the first 128 octets of the original datagram. At least 20 of these 128 octets represent the IP header of the original datagram.

If the original datagram was shorter than 128 octets, the final field MUST be padded with 0Æs.

When an LSR appends the data structure defined herein to an ICMP message, the ICMP "total length" MUST be equal to the data structure length plus 156. The first octet of the data structure must be displaced 156 octets from the beginning of the ICMP message.

The data structure defined in this section consists of a common header followed by object instances. Each object instance consists of an object header plus contents.

Currently, two object classes are defined. One object class contains an entire MPLS label stack, formatted exactly as it was when it arrived at the LSR that sends the ICMP message. The other contains some portion of the original datagram that could not be included in the final field of the ICMP message body (i.e., the octet 129 and beyond).

Both object classes are optional.

In the future, additional object classes may be defined.

6.1 Common Header

```
      0             1            2              3
+-------------+-------------+-------------+-------------+
| Vers |     (Reserved)     |          Checksum         |
+-------------+-------------+-------------+-------------+
```

The fields in the common header are as follows:

Vers: 4 bits

ICMP extension version number.

This is version 2.

Checksum: 16 bits

The one's complement of the one's complement sum of the data structure, with the checksum field replaced by zero for the purpose of computing the checksum. An all-zero value means that no checksum was transmitted.

If the checksum field contains a value other than described above, the ICMP message does not include the extensions described in this memo. This, however, does not imply that the ICMP message is malformed. It may be in strict compliance with RFC-1812.

Reserved: Must be set to 0.

6.2 Object Header

Every object consists of one or more 32-bit words with a one-word header, with the following format:

```
+-------------+-------------+-------------+-------------+
|           Length          | Class-Num   | C-Type      |
+-------------+-------------+-------------+-------------+
|                                                       |
|               // (Object contents) //                 |
|                                                       |
+-------------+-------------+-------------+-------------+
```

An object header has the following fields:

Length: 16 bits

Length of the object, measured in octets, including the object header and object contents.

Class-Num: 8 bits

Identifies object class.

C-Type: 8 bits

Identifies object sub-type.

6.3 MPLS Stack Entry Object Class

A single instance of the MPLS Entry Object class represents the entire MPLS label stack, formatted exactly as it was when it arrived at the LSR that sends the ICMP message

In the illustration below, octets 0-3 depict the first member of the MPLS label stack. Each remaining member of the MPLS label stack is represented by another 4 octets that share the same format.

Syntax follows:

MPLS Stack Entry Class = 1, C-Type = 1.

```
        0             1             2            3
+-------------+-------------+-------------+-------------+
|              Label                |EXP |S|     TTL     |
+-------------+-------------+-------------+-------------+
|                                                       |
|       // Remaining MPLS Stack Entries //              |
|                                                       |
+-------------+-------------+-------------+-------------+
```

Label: 20 bits

Exp: Experimental Use, 3 bits

S: Bottom of Stack, 1 bit

TTL: Time to Live, 8 bits

6.4 Extended Payload Object Class

An instance of the Extended Payload Object class represents some portion of the original datagram that could not be fit in the final field of the ICMP message body (i.e., octets beyond 128).

Syntax follows:

MPLS Stack Entry Class = 2, C-Type = 1.

```
       0             1             2            3
+-------------+-------------+-------------+-------------+
|                                                       |
|       // Additional bytes of original datagram //     |
|                                                       |
+-------------+-------------+-------------+-------------+
```

7. Backward Compatibility

ICMP extensions proposed in this document MUST be backward compatible with the syntax described in RFC-792. Extensions proposed in this memo MUST NOT change or deprecate any field defined in RFC-792.

The extensions defined herein are in keeping with the spirit, if not the letter of RFC-1812. In order to support IP-in-IP tunneling, RFC-1812 extends the final field of selected ICMP messages to include a greater portion of the original datagram. Unfortunately, it extends this field to a variable length without adding a length attribute.

This memo binds the length of that final field to an arbitrarily large value (128 octets). Fixing the length of that field facilitates extension of the ICMP message. An additional object is provided through which octets 129 and beyond can be appended to the ICMP message.

As few datagrams contain L3 or L4 header information beyond octet 128, it is unlikely that the extensions described herein will disable any applications that rely upon RFC-1812 style ICMP messages.

8. Security Considerations

This memo presents no security considerations beyond those already presented by current ICMP applications (e.g., traceroute).

9. References

[ARCH], Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", Internet Draft <draft-ietf-mpls-arch-06.txt>, August, 1999

[ENCODE], Rosen, E., Rekhter, Y., Tappan, D, Farinacci, D., Fedorkow, G., Li, T., Conta, A., "MPLS Stack Encoding", Internet Draft, <draft-ietf-mpls-label-encapse-07.txt>, September 1999.

[MPLSATM], Davie, B., Lawrence, J., McCloghrie, K., Rekhter, Y., Rosen, E., Swallow, G, "MPLS using LDP and ATM VC Switching", <draft-ietf- mpls-atm-04.txt>, June 2000.

[MPLSFRAME], Conta, A., Doolan, P., Malis, A., "Use of Label Switching on Frame Relay Networks", <draft-ietf-mpls-fr-06.txt>, June, 2000.

[RFC-792], Postel, J., "Internet Control Message Protocol", RFC 792, ISI, September 1981.

[RFC-1812], Baker, F., "Requirements for IP Version 4 Routers", RFC 1812, June 1995.

[RFC-2026], Bradner, S., "Internet Standards Process Revision 3", RFC 2026, Harvard University, October 1996.

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

10. Acknowledgments

Thanks to Yakov Rekhter and Curtis Villamizar for their contributions to this memo.

11. Author's Addresses

Ronald P. Bonica
WorldCom
22001 Loudoun County Pkwy
Ashburn, Virginia, 20147
Phone: 703 886 1681

Email: Ronald.p.bonica@wcom.com

Daniel C. Tappan
Cisco Systems
250 Apollo Drive
Chelmsford, Massachusetts, 01824
Email: tappan@cisco.com

Der-Hwa Gan
Juniper Networks
385 Ravendale Drive
Mountain View, California 94043
Email: dhg@juniper.net

Full Copyright Statement