Network Working Group
Request for Comments: 3032
Category: Standards Track

E. Rosen
D. Tappan
G. Fedorkow
Cisco Systems, Inc.
Y. Rekhter
Juniper Networks
D. Farinacci
T. Li
Procket Networks, Inc.
A. Conta
TranSwitch Corporation

January 2001

MPLS Label Stack Encoding

Status of this Memo

Copyright Notice

Abstract

"Multi-Protocol Label Switching (MPLS)" [1] requires a set of procedures for augmenting network layer packets with "label stacks", thereby turning them into "labeled packets". Routers which support MPLS are known as "Label Switching Routers", or "LSRs". In order to transmit a labeled packet on a particular data link, an LSR must support an encoding technique which, given a label stack and a network layer packet, produces a labeled packet. This document specifies the encoding to be used by an LSR in order to transmit labeled packets on Point-to-Point Protocol (PPP) data links, on LAN data links, and possibly on other data links as well. On some data links, the label at the top of the stack may be encoded in a different manner, but the techniques described here MUST be used to encode the remainder of the label stack. This document also specifies rules and procedures for processing the various fields of the label stack encoding.

Table of Contents

1. Introduction

"Multi-Protocol Label Switching (MPLS)" [1] requires a set of procedures for augmenting network layer packets with "label stacks", thereby turning them into "labeled packets". Routers which support MPLS are known as "Label Switching Routers", or "LSRs". In order to transmit a labeled packet on a particular data link, an LSR must support an encoding technique which, given a label stack and a network layer packet, produces a labeled packet.

This document specifies the encoding to be used by an LSR in order to transmit labeled packets on PPP data links and on LAN data links. The specified encoding may also be useful for other data links as well.

This document also specifies rules and procedures for processing the various fields of the label stack encoding. Since MPLS is independent of any particular network layer protocol, the majority of such procedures are also protocol-independent. A few, however, do differ for different protocols. In this document, we specify the protocol-independent procedures, and we specify the protocol-dependent procedures for IPv4 and IPv6.

LSRs that are implemented on certain switching devices (such as ATM switches) may use different encoding techniques for encoding the top one or two entries of the label stack. When the label stack has additional entries, however, the encoding technique described in this document MUST be used for the additional label stack entries.

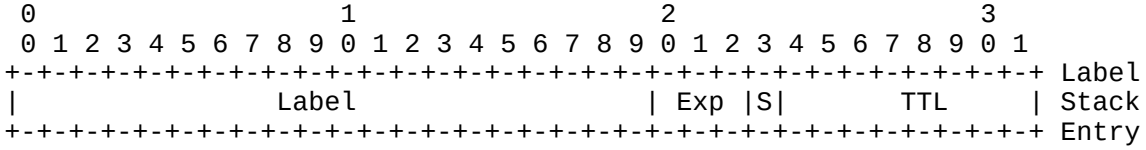
1.1. Specification of Requirements

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 [2].

2. The Label Stack

2.1. Encoding the Label Stack

The label stack is represented as a sequence of "label stack entries". Each label stack entry is represented by 4 octets. This is shown in Figure 1.



Label: Label Value, 20 bits
Exp: Experimental Use, 3 bits
S: Bottom of Stack, 1 bit
TTL: Time to Live, 8 bits

Figure 1

The label stack entries appear AFTER the data link layer headers, but BEFORE any network layer headers. The top of the label stack appears earliest in the packet, and the bottom appears latest. The network layer packet immediately follows the label stack entry which has the S bit set.

Each label stack entry is broken down into the following fields:

1. Bottom of Stack (S)

   This bit is set to one for the last entry in the label stack (i.e., for the bottom of the stack), and zero for all other label stack entries.

2. Time to Live (TTL)

   This eight-bit field is used to encode a time-to-live value. The processing of this field is described in section 2.4.

3. Experimental Use

   This three-bit field is reserved for experimental use.

4. Label Value

   This 20-bit field carries the actual value of the Label.

   When a labeled packet is received, the label value at the top of the stack is looked up. As a result of a successful lookup one learns:

   a) the next hop to which the packet is to be forwarded;

   b) the operation to be performed on the label stack before forwarding; this operation may be to replace the top label stack entry with another, or to pop an entry off the label stack, or to replace the top label stack entry and then to push one or more additional entries on the label stack.

   In addition to learning the next hop and the label stack operation, one may also learn the outgoing data link encapsulation, and possibly other information which is needed in order to properly forward the packet.

There are several reserved label values:

i. A value of 0 represents the "IPv4 Explicit NULL Label". This label value is only legal at the bottom of the label stack. It indicates that the label stack must be popped, and the forwarding of the packet must then be based on the IPv4 header.

ii. A value of 1 represents the "Router Alert Label". This label value is legal anywhere in the label stack except at the bottom. When a received packet contains this label value at the top of the label stack, it is delivered to a local software module for processing. The actual forwarding of the packet is determined by the label beneath it in the stack. However, if the packet is forwarded further, the Router Alert Label should be pushed back onto the label stack before forwarding. The use of this label is analogous to the use of the "Router Alert Option" in IP packets [5]. Since this label cannot occur at the bottom of the stack, it is not associated with a particular network layer protocol.

iii. A value of 2 represents the "IPv6 Explicit NULL Label". This label value is only legal at the bottom of the label stack. It indicates that the label stack must be popped, and the forwarding of the packet must then be based on the IPv6 header.

iv. A value of 3 represents the "Implicit NULL Label". This is a label that an LSR may assign and distribute, but which never actually appears in the encapsulation. When an LSR would otherwise replace the label at the top of the stack with a new label, but the new label is "Implicit NULL", the LSR will pop the stack instead of doing the replacement. Although this value may never appear in the encapsulation, it needs to be specified in the Label Distribution Protocol, so a value is reserved.

v. Values 4-15 are reserved.

2.2. Determining the Network Layer Protocol

When the last label is popped from a packet’s label stack (resulting in the stack being emptied), further processing of the packet is based on the packet’s network layer header. The LSR which pops the last label off the stack must therefore be able to identify the packet’s network layer protocol. However, the label stack does not contain any field which explicitly identifies the network layer

protocol. This means that the identity of the network layer protocol must be inferable from the value of the label which is popped from the bottom of the stack, possibly along with the contents of the network layer header itself.

Therefore, when the first label is pushed onto a network layer packet, either the label must be one which is used ONLY for packets of a particular network layer, or the label must be one which is used ONLY for a specified set of network layer protocols, where packets of the specified network layers can be distinguished by inspection of the network layer header. Furthermore, whenever that label is replaced by another label value during a packet's transit, the new value must also be one which meets the same criteria. If these conditions are not met, the LSR which pops the last label off a packet will not be able to identify the packet's network layer protocol.

Adherence to these conditions does not necessarily enable intermediate nodes to identify a packet's network layer protocol. Under ordinary conditions, this is not necessary, but there are error conditions under which it is desirable. For instance, if an intermediate LSR determines that a labeled packet is undeliverable, it may be desirable for that LSR to generate error messages which are specific to the packet's network layer. The only means the intermediate LSR has for identifying the network layer is inspection of the top label and the network layer header. So if intermediate nodes are to be able to generate protocol-specific error messages for labeled packets, all labels in the stack must meet the criteria specified above for labels which appear at the bottom of the stack.

If a packet cannot be forwarded for some reason (e.g., it exceeds the data link MTU), and either its network layer protocol cannot be identified, or there are no specified protocol-dependent rules for handling the error condition, then the packet MUST be silently discarded.

2.3. Generating ICMP Messages for Labeled IP Packets

Section 2.4 and section 3 discuss situations in which it is desirable to generate ICMP messages for labeled IP packets. In order for a particular LSR to be able to generate an ICMP packet and have that packet sent to the source of the IP packet, two conditions must hold:

1. it must be possible for that LSR to determine that a particular labeled packet is an IP packet;

2. it must be possible for that LSR to route to the packet's IP source address.

Condition 1 is discussed in section 2.2. The following two subsections discuss condition 2. However, there will be some cases in which condition 2 does not hold at all, and in these cases it will not be possible to generate the ICMP message.

2.3.1. Tunneling through a Transit Routing Domain

Suppose one is using MPLS to "tunnel" through a transit routing domain, where the external routes are not leaked into the domain's interior routers. For example, the interior routers may be running OSPF, and may only know how to reach destinations within that OSPF domain. The domain might contain several Autonomous System Border Routers (ASBRs), which talk BGP to each other. However, in this example the routes from BGP are not distributed into OSPF, and the LSRs which are not ASBRs do not run BGP.

In this example, only an ASBR will know how to route to the source of some arbitrary packet. If an interior router needs to send an ICMP message to the source of an IP packet, it will not know how to route the ICMP message.

One solution is to have one or more of the ASBRs inject "default" into the IGP. (N.B.: this does NOT require that there be a "default" carried by BGP.) This would then ensure that any unlabeled packet which must leave the domain (such as an ICMP packet) gets sent to a router which has full routing information. The routers with full routing information will label the packets before sending them back through the transit domain, so the use of default routing within the transit domain does not cause any loops.

This solution only works for packets which have globally unique addresses, and for networks in which all the ASBRs have complete routing information. The next subsection describes a solution which works when these conditions do not hold.

2.3.2. Tunneling Private Addresses through a Public Backbone

In some cases where MPLS is used to tunnel through a routing domain, it may not be possible to route to the source address of a fragmented packet at all. This would be the case, for example, if the IP addresses carried in the packet were private (i.e., not globally unique) addresses, and MPLS were being used to tunnel those packets through a public backbone. Default routing to an ASBR will not work in this environment.

In this environment, in order to send an ICMP message to the source of a packet, one can copy the label stack from the original packet to the ICMP message, and then label switch the ICMP message. This will

cause the message to proceed in the direction of the original packet's destination, rather than its source. Unless the message is label switched all the way to the destination host, it will end up, unlabeled, in a router which does know how to route to the source of original packet, at which point the message will be sent in the proper direction.

This technique can be very useful if the ICMP message is a "Time Exceeded" message or a "Destination Unreachable because fragmentation needed and DF set" message.

When copying the label stack from the original packet to the ICMP message, the label values must be copied exactly, but the TTL values in the label stack should be set to the TTL value that is placed in the IP header of the ICMP message. This TTL value should be long enough to allow the circuitous route that the ICMP message will need to follow.

Note that if a packet's TTL expiration is due to the presence of a routing loop, then if this technique is used, the ICMP message may loop as well. Since an ICMP message is never sent as a result of receiving an ICMP message, and since many implementations throttle the rate at which ICMP messages can be generated, this is not expected to pose a problem.

2.4. Processing the Time to Live Field

2.4.1. Definitions

The "incoming TTL" of a labeled packet is defined to be the value of the TTL field of the top label stack entry when the packet is received.

The "outgoing TTL" of a labeled packet is defined to be the larger of:

a) one less than the incoming TTL,
b) zero.

2.4.2. Protocol-independent rules

If the outgoing TTL of a labeled packet is 0, then the labeled packet MUST NOT be further forwarded; nor may the label stack be stripped off and the packet forwarded as an unlabeled packet. The packet's lifetime in the network is considered to have expired.

Depending on the label value in the label stack entry, the packet MAY be simply discarded, or it may be passed to the appropriate "ordinary" network layer for error processing (e.g., for the generation of an ICMP error message, see section 2.3).

When a labeled packet is forwarded, the TTL field of the label stack entry at the top of the label stack MUST be set to the outgoing TTL value.

Note that the outgoing TTL value is a function solely of the incoming TTL value, and is independent of whether any labels are pushed or popped before forwarding. There is no significance to the value of the TTL field in any label stack entry which is not at the top of the stack.

2.4.3. IP-dependent rules

We define the "IP TTL" field to be the value of the IPv4 TTL field, or the value of the IPv6 Hop Limit field, whichever is applicable.

When an IP packet is first labeled, the TTL field of the label stack entry MUST BE set to the value of the IP TTL field. (If the IP TTL field needs to be decremented, as part of the IP processing, it is assumed that this has already been done.)

When a label is popped, and the resulting label stack is empty, then the value of the IP TTL field SHOULD BE replaced with the outgoing TTL value, as defined above. In IPv4 this also requires modification of the IP header checksum.

It is recognized that there may be situations where a network administration prefers to decrement the IPv4 TTL by one as it traverses an MPLS domain, instead of decrementing the IPv4 TTL by the number of LSP hops within the domain.

2.4.4. Translating Between Different Encapsulations

Sometimes an LSR may receive a labeled packet over, e.g., a label switching controlled ATM (LC-ATM) interface [9], and may need to send it out over a PPP or LAN link. Then the incoming packet will not be received using the encapsulation specified in this document, but the outgoing packet will be sent using the encapsulation specified in this document.

In this case, the value of the "incoming TTL" is determined by the procedures used for carrying labeled packets on, e.g., LC-ATM interfaces. TTL processing then proceeds as described above.

Sometimes an LSR may receive a labeled packet over a PPP or a LAN link, and may need to send it out, say, an LC-ATM interface. Then the incoming packet will be received using the encapsulation specified in this document, but the outgoing packet will not be sent using the encapsulation specified in this document. In this case, the procedure for carrying the value of the "outgoing TTL" is determined by the procedures used for carrying labeled packets on, e.g., LC-ATM interfaces.

3. Fragmentation and Path MTU Discovery

Just as it is possible to receive an unlabeled IP datagram which is too large to be transmitted on its output link, it is possible to receive a labeled packet which is too large to be transmitted on its output link.

It is also possible that a received packet (labeled or unlabeled) which was originally small enough to be transmitted on that link becomes too large by virtue of having one or more additional labels pushed onto its label stack. In label switching, a packet may grow in size if additional labels get pushed on. Thus if one receives a labeled packet with a 1500-byte frame payload, and pushes on an additional label, one needs to forward it as frame with a 1504-byte payload.

This section specifies the rules for processing labeled packets which are "too large". In particular, it provides rules which ensure that hosts implementing Path MTU Discovery [4], and hosts using IPv6 [7,8], will be able to generate IP datagrams that do not need fragmentation, even if those datagrams get labeled as they traverse the network.

In general, IPv4 hosts which do not implement Path MTU Discovery [4] send IP datagrams which contain no more than 576 bytes. Since the MTUs in use on most data links today are 1500 bytes or more, the probability that such datagrams will need to get fragmented, even if they get labeled, is very small.

Some hosts that do not implement Path MTU Discovery [4] will generate IP datagrams containing 1500 bytes, as long as the IP Source and Destination addresses are on the same subnet. These datagrams will not pass through routers, and hence will not get fragmented.

Unfortunately, some hosts will generate IP datagrams containing 1500 bytes, as long the IP Source and Destination addresses have the same classful network number. This is the one case in which there is any risk of fragmentation when such datagrams get labeled. (Even so,

fragmentation is not likely unless the packet must traverse an ethernet of some sort between the time it first gets labeled and the time it gets unlabeled.)

This document specifies procedures which allow one to configure the network so that large datagrams from hosts which do not implement Path MTU Discovery get fragmented just once, when they are first labeled. These procedures make it possible (assuming suitable configuration) to avoid any need to fragment packets which have already been labeled.

3.1. Terminology

With respect to a particular data link, we can use the following terms:

- Frame Payload:

  The contents of a data link frame, excluding any data link layer headers or trailers (e.g., MAC headers, LLC headers, 802.1Q headers, PPP header, frame check sequences, etc.).

  When a frame is carrying an unlabeled IP datagram, the Frame Payload is just the IP datagram itself. When a frame is carrying a labeled IP datagram, the Frame Payload consists of the label stack entries and the IP datagram.

- Conventional Maximum Frame Payload Size:

  The maximum Frame Payload size allowed by data link standards. For example, the Conventional Maximum Frame Payload Size for ethernet is 1500 bytes.

- True Maximum Frame Payload Size:

  The maximum size frame payload which can be sent and received properly by the interface hardware attached to the data link.

  On ethernet and 802.3 networks, it is believed that the True Maximum Frame Payload Size is 4-8 bytes larger than the Conventional Maximum Frame Payload Size (as long as neither an 802.1Q header nor an 802.1p header is present, and as long as neither can be added by a switch or bridge while a packet is in transit to its next hop). For example, it is believed that most ethernet equipment could correctly send and receive packets carrying a payload of 1504 or perhaps even 1508 bytes, at least, as long as the ethernet header does not have an 802.1Q or 802.1p field.

On PPP links, the True Maximum Frame Payload Size may be virtually unbounded.

- Effective Maximum Frame Payload Size for Labeled Packets:

  This is either the Conventional Maximum Frame Payload Size or the True Maximum Frame Payload Size, depending on the capabilities of the equipment on the data link and the size of the data link header being used.

- Initially Labeled IP Datagram:

  Suppose that an unlabeled IP datagram is received at a particular LSR, and that the the LSR pushes on a label before forwarding the datagram. Such a datagram will be called an Initially Labeled IP Datagram at that LSR.

- Previously Labeled IP Datagram:

  An IP datagram which had already been labeled before it was received by a particular LSR.

3.2. Maximum Initially Labeled IP Datagram Size

Every LSR which is capable of

a) receiving an unlabeled IP datagram,
b) adding a label stack to the datagram, and
c) forwarding the resulting labeled packet,

SHOULD support a configuration parameter known as the "Maximum Initially Labeled IP Datagram Size", which can be set to a non-negative value.

If this configuration parameter is set to zero, it has no effect.

If it is set to a positive value, it is used in the following way. If:

a) an unlabeled IP datagram is received, and
b) that datagram does not have the DF bit set in its IP header, and
c) that datagram needs to be labeled before being forwarded, and
d) the size of the datagram (before labeling) exceeds the value of the parameter,

then

a) the datagram must be broken into fragments, each of whose size is no greater than the value of the parameter, and

b) each fragment must be labeled and then forwarded.

For example, if this configuration parameter is set to a value of 1488, then any unlabeled IP datagram containing more than 1488 bytes will be fragmented before being labeled. Each fragment will be capable of being carried on a 1500-byte data link, without further fragmentation, even if as many as three labels are pushed onto its label stack.

In other words, setting this parameter to a non-zero value allows one to eliminate all fragmentation of Previously Labeled IP Datagrams, but it may cause some unnecessary fragmentation of Initially Labeled IP Datagrams.

Note that the setting of this parameter does not affect the processing of IP datagrams that have the DF bit set; hence the result of Path MTU discovery is unaffected by the setting of this parameter.

3.3. When are Labeled IP Datagrams Too Big?

A labeled IP datagram whose size exceeds the Conventional Maximum Frame Payload Size of the data link over which it is to be forwarded MAY be considered to be "too big".

A labeled IP datagram whose size exceeds the True Maximum Frame Payload Size of the data link over which it is to be forwarded MUST be considered to be "too big".

A labeled IP datagram which is not "too big" MUST be transmitted without fragmentation.

3.4. Processing Labeled IPv4 Datagrams which are Too Big

If a labeled IPv4 datagram is "too big", and the DF bit is not set in its IP header, then the LSR MAY silently discard the datagram.

Note that discarding such datagrams is a sensible procedure only if the "Maximum Initially Labeled IP Datagram Size" is set to a non-zero value in every LSR in the network which is capable of adding a label stack to an unlabeled IP datagram.

If the LSR chooses not to discard a labeled IPv4 datagram which is too big, or if the DF bit is set in that datagram, then it MUST execute the following algorithm:

1. Strip off the label stack entries to obtain the IP datagram.

2. Let N be the number of bytes in the label stack (i.e, 4 times the number of label stack entries).

3. If the IP datagram does NOT have the "Don't Fragment" bit set in its IP header:

   a. convert it into fragments, each of which MUST be at least N bytes less than the Effective Maximum Frame Payload Size.

   b. Prepend each fragment with the same label header that would have been on the original datagram had fragmentation not been necessary.

   c. Forward the fragments

4. If the IP datagram has the "Don't Fragment" bit set in its IP header:

   a. the datagram MUST NOT be forwarded

   b. Create an ICMP Destination Unreachable Message:

      i. set its Code field [3] to "Fragmentation Required and DF Set",

      ii. set its Next-Hop MTU field [4] to the difference between the Effective Maximum Frame Payload Size and the value of N

   c. If possible, transmit the ICMP Destination Unreachable Message to the source of the of the discarded datagram.

3.5. Processing Labeled IPv6 Datagrams which are Too Big

To process a labeled IPv6 datagram which is too big, an LSR MUST execute the following algorithm:

1. Strip off the label stack entries to obtain the IP datagram.

2. Let N be the number of bytes in the label stack (i.e., 4 times the number of label stack entries).

3. If the IP datagram contains more than 1280 bytes (not counting the label stack entries), or if it does not contain a fragment header, then:

a. Create an ICMP Packet Too Big Message, and set its Next-Hop MTU field to the difference between the Effective Maximum Frame Payload Size and the value of N

b. If possible, transmit the ICMP Packet Too Big Message to the source of the datagram.

c. discard the labeled IPv6 datagram.

4. If the IP datagram is not larger than 1280 octets, and it contains a fragment header, then

a. Convert it into fragments, each of which MUST be at least N bytes less than the Effective Maximum Frame Payload Size.

b. Prepend each fragment with the same label header that would have been on the original datagram had fragmentation not been necessary.

c. Forward the fragments.

Reassembly of the fragments will be done at the destination host.

3.6. Implications with respect to Path MTU Discovery

The procedures described above for handling datagrams which have the DF bit set, but which are "too large", have an impact on the Path MTU Discovery procedures of RFC 1191 [4]. Hosts which implement these procedures will discover an MTU which is small enough to allow n labels to be pushed on the datagrams, without need for fragmentation, where n is the number of labels that actually get pushed on along the path currently in use.

In other words, datagrams from hosts that use Path MTU Discovery will never need to be fragmented due to the need to put on a label header, or to add new labels to an existing label header. (Also, datagrams from hosts that use Path MTU Discovery generally have the DF bit set, and so will never get fragmented anyway.)

Note that Path MTU Discovery will only work properly if, at the point where a labeled IP Datagram’s fragmentation needs to occur, it is possible to cause an ICMP Destination Unreachable message to be routed to the packet’s source address. See section 2.3.

If it is not possible to forward an ICMP message from within an MPLS "tunnel" to a packet's source address, but the network configuration makes it possible for the LSR at the transmitting end of the tunnel to receive packets that must go through the tunnel, but are too large to pass through the tunnel unfragmented, then:

- The LSR at the transmitting end of the tunnel MUST be able to determine the MTU of the tunnel as a whole. It MAY do this by sending packets through the tunnel to the tunnel's receiving endpoint, and performing Path MTU Discovery with those packets.

- Any time the transmitting endpoint of the tunnel needs to send a packet into the tunnel, and that packet has the DF bit set, and it exceeds the tunnel MTU, the transmitting endpoint of the tunnel MUST send the ICMP Destination Unreachable message to the source, with code "Fragmentation Required and DF Set", and the Next-Hop MTU Field set as described above.

4. Transporting Labeled Packets over PPP

The Point-to-Point Protocol (PPP) [6] provides a standard method for transporting multi-protocol datagrams over point-to-point links. PPP defines an extensible Link Control Protocol, and proposes a family of Network Control Protocols for establishing and configuring different network-layer protocols.

This section defines the Network Control Protocol for establishing and configuring label Switching over PPP.

4.1. Introduction

PPP has three main components:

1. A method for encapsulating multi-protocol datagrams.

2. A Link Control Protocol (LCP) for establishing, configuring, and testing the data-link connection.

3. A family of Network Control Protocols for establishing and configuring different network-layer protocols.

In order to establish communications over a point-to-point link, each end of the PPP link must first send LCP packets to configure and test the data link. After the link has been established and optional facilities have been negotiated as needed by the LCP, PPP must send "MPLS Control Protocol" packets to enable the transmission of labeled packets. Once the "MPLS Control Protocol" has reached the Opened state, labeled packets can be sent over the link.

The link will remain configured for communications until explicit LCP or MPLS Control Protocol packets close the link down, or until some external event occurs (an inactivity timer expires or network administrator intervention).

4.2. A PPP Network Control Protocol for MPLS

The MPLS Control Protocol (MPLSCP) is responsible for enabling and disabling the use of label switching on a PPP link. It uses the same packet exchange mechanism as the Link Control Protocol (LCP). MPLSCP packets may not be exchanged until PPP has reached the Network-Layer Protocol phase. MPLSCP packets received before this phase is reached should be silently discarded.

The MPLS Control Protocol is exactly the same as the Link Control Protocol [6] with the following exceptions:

1. Frame Modifications

   The packet may utilize any modifications to the basic frame format which have been negotiated during the Link Establishment phase.

2. Data Link Layer Protocol Field

   Exactly one MPLSCP packet is encapsulated in the PPP Information field, where the PPP Protocol field indicates type hex 8281 (MPLS).

3. Code field

   Only Codes 1 through 7 (Configure-Request, Configure-Ack, Configure-Nak, Configure-Reject, Terminate-Request, Terminate-Ack and Code-Reject) are used. Other Codes should be treated as unrecognized and should result in Code-Rejects.

4. Timeouts

   MPLSCP packets may not be exchanged until PPP has reached the Network-Layer Protocol phase. An implementation should be prepared to wait for Authentication and Link Quality Determination to finish before timing out waiting for a Configure-Ack or other response. It is suggested that an implementation give up only after user intervention or a configurable amount of time.

5. Configuration Option Types

None.

4.3. Sending Labeled Packets

Before any labeled packets may be communicated, PPP must reach the Network-Layer Protocol phase, and the MPLS Control Protocol must reach the Opened state.

Exactly one labeled packet is encapsulated in the PPP Information field, where the PPP Protocol field indicates either type hex 0281 (MPLS Unicast) or type hex 0283 (MPLS Multicast). The maximum length of a labeled packet transmitted over a PPP link is the same as the maximum length of the Information field of a PPP encapsulated packet.

The format of the Information field itself is as defined in section 2.

Note that two codepoints are defined for labeled packets; one for multicast and one for unicast. Once the MPLSCP has reached the Opened state, both label switched multicasts and label switched unicasts can be sent over the PPP link.

4.4. Label Switching Control Protocol Configuration Options

There are no configuration options.

5. Transporting Labeled Packets over LAN Media

Exactly one labeled packet is carried in each frame.

The label stack entries immediately precede the network layer header, and follow any data link layer headers, including, e.g., any 802.1Q headers that may exist.

The ethertype value 8847 hex is used to indicate that a frame is carrying an MPLS unicast packet.

The ethertype value 8848 hex is used to indicate that a frame is carrying an MPLS multicast packet.

These ethertype values can be used with either the ethernet encapsulation or the 802.3 LLC/SNAP encapsulation to carry labeled packets. The procedure for choosing which of these two encapsulations to use is beyond the scope of this document.

6. IANA Considerations

Label values 0-15 inclusive have special meaning, as specified in this document, or as further assigned by IANA.

In this document, label values 0-3 are specified in section 2.1.

Label values 4-15 may be assigned by IANA, based on IETF Consensus.

7. Security Considerations

The MPLS encapsulation that is specified herein does not raise any security issues that are not already present in either the MPLS architecture [1] or in the architecture of the network layer protocol contained within the encapsulation.

There are two security considerations inherited from the MPLS architecture which may be pointed out here:

- Some routers may implement security procedures which depend on the network layer header being in a fixed place relative to the data link layer header. These procedures will not work when the MPLS encapsulation is used, because that encapsulation is of a variable size.

- An MPLS label has its meaning by virtue of an agreement between the LSR that puts the label in the label stack (the "label writer"), and the LSR that interprets that label (the "label reader"). However, the label stack does not provide any means of determining who the label writer was for any particular label. If labeled packets are accepted from untrusted sources, the result may be that packets are routed in an illegitimate manner.

8. Intellectual Property

The IETF has been notified of intellectual property rights claimed in regard to some or all of the specification contained in this document. For more information consult the online list of claimed rights.

9. Authors' Addresses

Eric C. Rosen
Cisco Systems, Inc.
250 Apollo Drive
Chelmsford, MA, 01824

EMail: erosen@cisco.com

Dan Tappan
Cisco Systems, Inc.
250 Apollo Drive
Chelmsford, MA, 01824

EMail: tappan@cisco.com

Yakov Rekhter
Juniper Networks
1194 N. Mathilda Avenue
Sunnyvale, CA 94089

EMail: yakov@juniper.net

Guy Fedorkow
Cisco Systems, Inc.
250 Apollo Drive
Chelmsford, MA, 01824

EMail: fedorkow@cisco.com

Dino Farinacci
Procket Networks, Inc.
3910 Freedom Circle, Ste. 102A
Santa Clara, CA 95054

EMail: dino@procket.com

Tony Li
Procket Networks, Inc.
3910 Freedom Circle, Ste. 102A
Santa Clara, CA 95054

EMail: tli@procket.com

Alex Conta
TranSwitch Corporation
3 Enterprise Drive
Shelton, CT, 06484

EMail: aconta@txc.com

10. References

[1] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, January 2001.

[2] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[3] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, September 1981.

[4] Mogul, J. and S. Deering, "Path MTU Discovery", RFC 1191, November 1990.

[5] Katz, D., "IP Router Alert Option", RFC 2113, February 1997.

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

[7] Conta, A. and S. Deering, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 1885, December 1995.

[8] McCann, J., Deering, S. and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, August 1996.

[9] Davie, B., Lawrence, J., McCloghrie, K., Rekhter, Y., Rosen, E. and G. Swallow, "MPLS Using LDP and ATM VC Switching", RFC 3035, January 2001.

11. Full Copyright Statement

Acknowledgement

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