Network Working Group
Internet Draft
Expiration Date: January 1999

Eric C. Rosen
Cisco Systems, Inc.

Arun Viswanathan
Lucent Technologies

Ross Callon
IronBridge Networks, Inc.

July 1998

Multiprotocol Label Switching Architecture

draft-ietf-mpls-arch-02.txt

Status of this Memo

Abstract

This internet draft specifies the architecture for multiprotocol label switching (MPLS). The architecture is based on other label switching approaches [2-11] as well as on the MPLS Framework document [1].

Table of Contents

1 Introduction to MPLS 4
1.1 Overview 4
1.2 Terminology 6
1.3 Acronyms and Abbreviations 9
1.4 Acknowledgments 10
2 Outline of Approach 10
2.1 Labels 11
2.2 Upstream and Downstream LSRs 12
2.3 Labeled Packet 12
2.4 Label Assignment and Distribution 12
2.5 Attributes of a Label Binding 12
2.6 Label Distribution Protocol (LDP) 13
2.7 Downstream vs. Downstream-on-Demand 13
2.8 Label Retention Mode 13
2.9 The Label Stack 14
2.10 The Next Hop Label Forwarding Entry (NHLFE) 14
2.11 Incoming Label Map (ILM) 15
2.12 FEC-to-NHLFE Map (FTN) 15
2.13 Label Swapping 16
2.14 Scope and Uniqueness of Labels 16
2.15 Label Switched Path (LSP), LSP Ingress, LSP Egress . 17
2.16 Penultimate Hop Popping 19
2.17 LSP Next Hop 20
2.18 Invalid Incoming Labels 21
2.19 LSP Control: Ordered versus Independent 21
2.20 Aggregation 22
2.21 Route Selection 24
2.22 Time-to-Live (TTL) 25
2.23 Loop Control 26
2.23.1 Loop Prevention 27
2.23.2 Interworking of Loop Control Options 29
2.24 Label Encodings 30
2.24.1 MPLS-specific Hardware and/or Software 31
2.24.2 ATM Switches as LSRs 31
2.24.3 Interoperability among Encoding Techniques 33
2.25 Label Merging 33
2.25.1 Non-merging LSRs 34
2.25.2 Labels for Merging and Non-Merging LSRs 35
2.25.3 Merge over ATM 36
2.25.3.1 Methods of Eliminating Cell Interleave 36
2.25.3.2 Interoperation: VC Merge, VP Merge, and Non-Merge .. 36
2.26 Tunnels and Hierarchy 37
2.26.1 Hop-by-Hop Routed Tunnel 38
2.26.2 Explicitly Routed Tunnel 38

2.26.3 LSP Tunnels 38
2.26.4 Hierarchy: LSP Tunnels within LSPs 39
2.26.5 LDP Peering and Hierarchy 39
2.27 LDP Transport 40
2.28 Multicast 41
3 Some Applications of MPLS 41
3.1 MPLS and Hop by Hop Routed Traffic 41
3.1.1 Labels for Address Prefixes 41
3.1.2 Distributing Labels for Address Prefixes 41
3.1.2.1 LDP Peers for a Particular Address Prefix 41
3.1.2.2 Distributing Labels 42
3.1.3 Using the Hop by Hop path as the LSP 43
3.1.4 LSP Egress and LSP Proxy Egress 43
3.1.5 The Implicit NULL Label 44
3.1.6 Option: Egress-Targeted Label Assignment 45
3.2 MPLS and Explicitly Routed LSPs 46
3.2.1 Explicitly Routed LSP Tunnels: Traffic Engineering . 46
3.3 Label Stacks and Implicit Peering 47
3.4 MPLS and Multi-Path Routing 48
3.5 LSP Trees as Multipoint-to-Point Entities 48
3.6 LSP Tunneling between BGP Border Routers 49
3.7 Other Uses of Hop-by-Hop Routed LSP Tunnels 50
3.8 MPLS and Multicast 51
4 LDP Procedures for Hop-by-Hop Routed Traffic 51
4.1 The Procedures for Advertising and Using labels 51
4.1.1 Downstream LSR: Distribution Procedure 52
4.1.1.1 PushUnconditional 52
4.1.1.2 PushConditional 53
4.1.1.3 PulledUnconditional 53
4.1.1.4 PulledConditional 54
4.1.2 Upstream LSR: Request Procedure 55
4.1.2.1 RequestNever 55
4.1.2.2 RequestWhenNeeded 55
4.1.2.3 RequestOnRequest 55
4.1.3 Upstream LSR: NotAvailable Procedure 56
4.1.3.1 RequestRetry 56
4.1.3.2 RequestNoRetry 56
4.1.4 Upstream LSR: Release Procedure 56
4.1.4.1 ReleaseOnChange 56
4.1.4.2 NoReleaseOnChange 57
4.1.5 Upstream LSR: labelUse Procedure 57
4.1.5.1 UseImmediate 57
4.1.5.2 UseIfLoopFree 57
4.1.5.3 UseIfLoopNotDetected 58
4.1.6 Downstream LSR: Withdraw Procedure 58
4.2 MPLS Schemes: Supported Combinations of Procedures . 59
4.2.1 TTL-capable LSP Segments 59
4.2.2 Using ATM Switches as LSRs 60

4.2.2.1 Without Label Merging 60
4.2.2.2 With Label Merging 61
4.2.3 Interoperability Considerations 62
5 Security Considerations 63
6 Authors' Addresses 63
7 References 64

1. Introduction to MPLS

1.1. Overview

In connectionless network layer protocols, as a packet travels from one router hop to the next, an independent forwarding decision is made at each hop. Each router runs a network layer routing algorithm. As a packet travels through the network, each router analyzes the packet header. The choice of next hop for a packet is based on the header analysis and the result of running the routing algorithm.

Packet headers contain considerably more information than is needed simply to choose the next hop. Choosing the next hop can therefore be thought of as the composition of two functions. The first function partitions the entire set of possible packets into a set of "Forwarding Equivalence Classes (FECs)". The second maps each FEC to a next hop. Insofar as the forwarding decision is concerned, different packets which get mapped into the same FEC are indistinguishable. All packets which belong to a particular FEC and which travel from a particular node will follow the same path.

In conventional IP forwarding, a particular router will typically consider two packets to be in the same FEC if there is some address prefix X in that router's routing tables such that X is the "longest match" for each packet's destination address. As the packet traverses the network, each hop in turn reexamines the packet and assigns it to a FEC.

In MPLS, the assignment of a particular packet to a particular FEC is done just once, as the packet enters the network. The FEC to which the packet is assigned is encoded with a short fixed length value known as a "label". When a packet is forwarded to its next hop, the label is sent along with it; that is, the packets are "labeled".

At subsequent hops, there is no further analysis of the packet's network layer header. Rather, the label is used as an index into a table which specifies the next hop, and a new label. The old label

is replaced with the new label, and the packet is forwarded to its next hop. If assignment to a FEC is based on a "longest match", this eliminates the need to perform a longest match computation for each packet at each hop; the computation can be performed just once.

Some routers analyze a packet's network layer header not merely to choose the packet's next hop, but also to determine a packet's "precedence" or "class of service", in order to apply different discard thresholds or scheduling disciplines to different packets. MPLS allows the precedence or class of service to be inferred from the label, so that no further header analysis is needed; in some cases MPLS provides a way to explicitly encode a class of service in the "label header".

The fact that a packet is assigned to a FEC just once, rather than at every hop, allows the use of sophisticated forwarding paradigms. A packet that enters the network at a particular router can be labeled differently than the same packet entering the network at a different router, and as a result forwarding decisions that depend on the ingress point ("policy routing") can be easily made. In fact, the policy used to assign a packet to a FEC need not have only the network layer header as input; it may use arbitrary information about the packet, and/or arbitrary policy information as input. Since this decouples forwarding from routing, it allows one to use MPLS to support a large variety of routing policies that are difficult or impossible to support with just conventional network layer forwarding.

Similarly, MPLS facilitates the use of explicit routing, without requiring that each IP packet carry the explicit route. Explicit routes may be useful to support policy routing and traffic engineering.

MPLS makes use of a routing approach whereby the normal mode of operation is that L3 routing (e.g., existing IP routing protocols and/or new IP routing protocols) is used by all nodes to determine the routed path.

MPLS stands for "Multiprotocol" Label Switching, multiprotocol because its techniques are applicable to ANY network layer protocol. In this document, however, we focus on the use of IP as the network layer protocol.

A router which supports MPLS is known as a "Label Switching Router", or LSR.

A general discussion of issues related to MPLS is presented in "A Framework for Multiprotocol Label Switching" [1].

1.2. Terminology

This section gives a general conceptual overview of the terms used in this document. Some of these terms are more precisely defined in later sections of the document.

DLCI	a label used in Frame Relay networks to identify frame relay circuits
flow	a single instance of an application to application flow of data (as in the RSVP and IFMP use of the term "flow")
forwarding equivalence class	a group of IP packets which are forwarded in the same manner (e.g., over the same path, with the same forwarding treatment)
frame merge	label merging, when it is applied to operation over frame based media, so that the potential problem of cell interleave is not an issue.
label	a short fixed length physically contiguous identifier which is used to identify a FEC, usually of local significance.
label merging	the replacement of multiple incoming labels for a particular FEC with a single outgoing label
label swap	the basic forwarding operation consisting of looking up an incoming label to determine the outgoing label, encapsulation, port, and other data handling information.
label swapping	a forwarding paradigm allowing streamlined forwarding of data by using labels to identify classes of data packets which are treated indistinguishably when forwarding.

label switched hop	the hop between two MPLS nodes, on which forwarding is done using labels.
label switched path	the path created by the concatenation of one or more label switched hops, allowing a packet to be forwarded by swapping labels from an MPLS node to another MPLS node.
layer 2	the protocol layer under layer 3 (which therefore offers the services used by layer 3). Forwarding, when done by the swapping of short fixed length labels, occurs at layer 2 regardless of whether the label being examined is an ATM VPI/VCI, a frame relay DLCI, or an MPLS label.
layer 3	the protocol layer at which IP and its associated routing protocols operate link layer synonymous with layer 2
loop detection	a method of dealing with loops in which loops are allowed to be set up, and data may be transmitted over the loop, but the loop is later detected and closed
loop prevention	a method of dealing with loops in which data is never transmitted over a loop
label stack	an ordered set of labels
loop survival	a method of dealing with loops in which data may be transmitted over a loop, but means are employed to limit the amount of network resources which may be consumed by the looping data
label switched path	The path through one or more LSRs at one level of the hierarchy followed by a packets in a particular FEC.
label switching router	an MPLS node which is capable of forwarding native L3 packets

merge point	a node at which label merging is done
MPLS core standards	the standards which describe the core MPLS technology
MPLS domain	a contiguous set of nodes which operate MPLS routing and forwarding and which are also in one Routing or Administrative Domain
MPLS edge node	an MPLS node that connects an MPLS domain with a node which is outside of the domain, either because it does not run MPLS, and/or because it is in a different domain. Note that if an LSR has a neighboring host which is not running MPLS, that that LSR is an MPLS edge node.
MPLS egress node	an MPLS edge node in its role in handling traffic as it leaves an MPLS domain
MPLS ingress node	an MPLS edge node in its role in handling traffic as it enters an MPLS domain
MPLS label	a label which is carried in a packet header, and which represents the packet's FEC
MPLS node	a node which is running MPLS. An MPLS node will be aware of MPLS control protocols, will operate one or more L3 routing protocols, and will be capable of forwarding packets based on labels. An MPLS node may optionally be also capable of forwarding native L3 packets.
MultiProtocol Label Switching	an IETF working group and the effort associated with the working group
network layer	synonymous with layer 3
stack	synonymous with label stack

switched path	synonymous with label switched path
virtual circuit	a circuit used by a connection-oriented layer 2 technology such as ATM or Frame Relay, requiring the maintenance of state information in layer 2 switches.
VC merge	label merging where the MPLS label is carried in the ATM VCI field (or combined VPI/VCI field), so as to allow multiple VCs to merge into one single VC
VP merge	label merging where the MPLS label is carried din the ATM VPI field, so as to allow multiple VPs to be merged into one single VP. In this case two cells would have the same VCI value only if they originated from the same node. This allows cells from different sources to be distinguished via the VCI.
VPI/VCI	a label used in ATM networks to identify circuits

1.3. Acronyms and Abbreviations

ATM	Asynchronous Transfer Mode
BGP	Border Gateway Protocol
DLCI	Data Link Circuit Identifier
FEC	Forwarding Equivalence Class
FTN	FEC to NHLFE Map
IGP	Interior Gateway Protocol
ILM	Incoming Label Map
IP	Internet Protocol
LDP	Label Distribution Protocol
L2	Layer 2

L3	Layer 3
LSP	Label Switched Path
LSR	Label Switching Router
MPLS	MultiProtocol Label Switching
MPT	Multipoint to Point Tree
NHLFE	Next Hop Label Forwarding Entry
SVC	Switched Virtual Circuit
SVP	Switched Virtual Path
TTL	Time-To-Live
VC	Virtual Circuit
VCI	Virtual Circuit Identifier
VP	Virtual Path
VPI	Virtual Path Identifier

1.4. Acknowledgments

The ideas and text in this document have been collected from a number of sources and comments received. We would like to thank Rick Boivie, Paul Doolan, Nancy Feldman, Yakov Rekhter, Vijay Srinivasan, and George Swallow for their inputs and ideas.

2. Outline of Approach

In this section, we introduce some of the basic concepts of MPLS and describe the general approach to be used.

2.1. Labels

A label is a short, fixed length, locally significant identifier which is used to identify a FEC. The label which is put on a particular packet represents the Forwarding Equivalence Class to which that packet is assigned.

Most commonly, packets are assigned to FECS based on their destination network layer addresses. However, the label is never an encoding of the destination network layer address.

If Ru and Rd are LSRs, they may agree that when Ru transmits a packet to Rd, Ru will label with packet with label value L if and only if the packet is a member of a particular FEC F. That is, they can agree to a "binding" between label L and FEC F for packets moving from Ru to Rd. As a result of such an agreement, L becomes Ru's "outgoing label" representing FEC F, and L becomes Rd's "incoming label" representing FEC F.

Note that L does not necessarily represent FEC F for any packets other than those which are being sent from Ru to Rd. L is an arbitrary value whose binding to F is local to Ru and Rd.

When we speak above of packets "being sent" from Ru to to Rd, we do not imply either that the packet originated at Ru or that its destination is Rd. Rather, we mean to include packets which are "transit packets" at one or both of the LSRs.

Sometimes it may be difficult or even impossible for Rd to tell, of an arriving packet carrying label L, that the label L was placed in the packet by Ru, rather than by some other LSR. (This will typically be the case when Ru and Rd are not direct neighbors.) In such cases, Rd must make sure that the binding from label to FEC is one-to-one. That is, in such cases, Rd must not agree with Ru1 to bind L to FEC F1, while also agreeing with some other LSR Ru2 to bind L to a different FEC F2. It is the responsibility of each LSR to ensure that it can uniquely interpret its incoming labels.

2.2. Upstream and Downstream LSRs

Suppose Ru and Rd have agreed to bind label L to FEC F, for packets sent from Ru to Rd. Then with respect to this binding, Ru is the "upstream LSR", and Rd is the "downstream LSR".

To say that one node is upstream and one is downstream with respect to a given binding means only that a particular label represents a particular FEC in packets travelling from the upstream node to the downstream node. This is NOT meant to imply that packets in that FEC would actually be routed from the upstream node to the downstream node.

2.3. Labeled Packet

A "labeled packet" is a packet into which a label has been encoded. In some cases, the label resides in an encapsulation header which exists specifically for this purpose. In other cases, the label may reside in an existing data link or network layer header, as long as there is a field which is available for that purpose. The particular encoding technique to be used must be agreed to by both the entity which encodes the label and the entity which decodes the label.

2.4. Label Assignment and Distribution

In the MPLS architecture, the decision to bind a particular label L to a particular FEC F is made by the LSR which is DOWNSTREAM with respect to that binding. The downstream LSR then informs the upstream LSR of the binding. Thus labels are "downstream-assigned", and label bindings are distributed in the "downstream to upstream" direction.

2.5. Attributes of a Label Binding

A particular binding of label L to FEC F, distributed by Rd to Ru, may have associated "attributes". If Ru, acting as a downstream LSR, also distributes a binding of a label to FEC F, then under certain conditions, it may be required to also distribute the corresponding attribute that it received from Rd.

2.6. Label Distribution Protocol (LDP)

A Label Distribution Protocol (LDP) is a set of procedures by which one LSR informs another of the label/FEC bindings it has made. Two LSRs which use an LDP to exchange label/FEC binding information are known as "LDP Peers" with respect to the binding information they exchange. If two LSRs are LDP Peers, we will speak of there being an "LDP Adjacency" between them.

(N.B.: two LSRs may be LDP Peers with respect to some set of bindings, but not with respect to some other set of bindings.)

The LDP also encompasses any negotiations in which two LDP Peers need to engage in order to learn of each other's MPLS capabilities.

2.7. Downstream vs. Downstream-on-Demand

The MPLS architecture allows an LSR to explicitly request, from its next hop for a particular FEC, a label binding for that FEC. This is known as "downstream-on-demand" label distribution.

The MPLS architecture also allows an LSR to distribute bindings to LSRs that have not explicitly requested them. This is known as "downstream" label distribution.

Both of these label distribution techniques may be used in the same network at the same time. However, on any given LDP adjacency, the upstream LSR and the downstream LSR must agree on which technique is to be used.

2.8. Label Retention Mode

An LSR Ru may receive (or have received) a label binding for a particular FEC from an LSR Rd, even though Rd is not Ru's next hop (or is no longer Ru's next hop) for that FEC.

Ru then has the choice of whether to keep track of such bindings, or whether to discard such bindings. If Ru keeps track of such bindings, then it may immediately begin using the binding again if Rd eventually becomes its next hop for the FEC in question. If Ru discards such bindings, then if Rd later becomes the next hop, the binding will have to be reacquired.

If an LSR supports "Liberal Label Retention Mode", it maintains the bindings between a label and a FEC which are received from LSRs which are not its next hop for that FEC. If an LSR supports "Conservative

Label Retention Mode", it discards such bindings.

Liberal label retention mode allows for quicker adaptation to routing changes, especially if loop prevention (see section 2.23) is not being used. Conservative label retention mode though requires an LSR to maintain many fewer labels.

2.9. The Label Stack

So far, we have spoken as if a labeled packet carries only a single label. As we shall see, it is useful to have a more general model in which a labeled packet carries a number of labels, organized as a last-in, first-out stack. We refer to this as a "label stack".

IN MPLS, EVERY FORWARDING DECISION IS BASED EXCLUSIVELY ON THE LABEL AT THE TOP OF THE STACK.

Although, as we shall see, MPLS supports a hierarchy, the processing of a labeled packet is completely independent of the level of hierarchy. The processing is always based on the top label, without regard for the possibility that some number of other labels may have been "above it" in the past, or that some number of other labels may be below it at present.

An unlabeled packet can be thought of as a packet whose label stack is empty (i.e., whose label stack has depth 0).

If a packet's label stack is of depth m, we refer to the label at the bottom of the stack as the level 1 label, to the label above it (if such exists) as the level 2 label, and to the label at the top of the stack as the level m label.

The utility of the label stack will become clear when we introduce the notion of LSP Tunnel and the MPLS Hierarchy (section 2.26).

2.10. The Next Hop Label Forwarding Entry (NHLFE)

The "Next Hop Label Forwarding Entry" (NHLFE) is used when forwarding a labeled packet. It contains the following information:

1. the packet's next hop

2. the data link encapsulation to use when transmitting the packet

3. the way to encode the label stack when transmitting the packet

4. the operation to perform on the packet's label stack; this is one of the following operations:

   a) replace the label at the top of the label stack with a specified new label

   b) pop the label stack

   c) replace the label at the top of the label stack with a specified new label, and then push one or more specified new labels onto the label stack.

Note that at a given LSR, the packet's "next hop" might be that LSR itself. In this case, the LSR would need to pop the top level label, and then "forward" the resulting packet to itself. It would then make another forwarding decision, based on what remains after the label stacked is popped. This may still be a labeled packet, or it may be the native IP packet.

This implies that in some cases the LSR may need to operate on the IP header in order to forward the packet.

If the packet's "next hop" is the current LSR, then the label stack operation MUST be to "pop the stack".

2.11. Incoming Label Map (ILM)

The "Incoming Label Map" (ILM) is a mapping from incoming labels to NHLFEs. It is used when forwarding packets that arrive as labeled packets.

2.12. FEC-to-NHLFE Map (FTN)

The "FEC-to-NHLFE" (FTN) is a mapping from FECs to NHLFEs. It is used when forwarding packets that arrive unlabeled, but which are to be labeled before being forwarded.

2.13. Label Swapping

Label swapping is the use of the following procedures to forward a packet.

In order to forward a labeled packet, a LSR examines the label at the top of the label stack. It uses the ILM to map this label to an NHLFE. Using the information in the NHLFE, it determines where to forward the packet, and performs an operation on the packet's label stack. It then encodes the new label stack into the packet, and forwards the result.

In order to forward an unlabeled packet, a LSR analyzes the network layer header, to determine the packet's FEC. It then uses the FTN to map this to an NHLFE. Using the information in the NHLFE, it determines where to forward the packet, and performs an operation on the packet's label stack. (Popping the label stack would, of course, be illegal in this case.) It then encodes the new label stack into the packet, and forwards the result.

IT IS IMPORTANT TO NOTE THAT WHEN LABEL SWAPPING IS IN USE, THE NEXT HOP IS ALWAYS TAKEN FROM THE NHLFE; THIS MAY IN SOME CASES BE DIFFERENT FROM WHAT THE NEXT HOP WOULD BE IF MPLS WERE NOT IN USE.

2.14. Scope and Uniqueness of Labels

A given LSR Rd may bind label L1 to FEC F, and distribute that binding to LDP peer Ru1. Rd may also bind label L2 to FEC F, and distribute that binding to LDP peer Ru2. Whether or not L1 == L2 is not determined by the architecture; this is a local matter.

A given LSR Rd may bind label L to FEC F1, and distribute that binding to LDP peer Ru1. Rd may also bind label L to FEC F2, and distribute that binding to LDP peer Ru2. IF (AND ONLY IF) RD CAN TELL, WHEN IT RECEIVES A PACKET WHOSE TOP LABEL IS L, WHETHER THE LABEL WAS PUT THERE BY RU1 OR BY RU2, THEN THE ARCHITECTURE DOES NOT REQUIRE THAT F1 == F2.

In general, Rd can only tell whether it was Ru1 or Ru2 that put the particular label value L at the top of the label stack if the following conditions hold:

- Ru1 and Ru2 are the only LDP peers to which Rd distributed a binding of label value L, and

- Ru1 and Ru2 are each directly connected to Rd via a point-to-point interface.

When these conditions hold, an LSR may use labels that have "per interface" scope, i.e., which are only unique per interface. When these conditions do not hold, the labels must be unique over the LSR which has assigned them.

If a particular LSR Rd is attached to a particular LSR Ru over two point-to-point interfaces, then Rd may distribute to Rd a binding of label L to FEC F1, as well as a binding of label L to FEC F2, F1 != F2, if and only if each binding is valid only for packets which Ru sends to Rd over a particular one of the interfaces. In all other cases, Rd MUST NOT distribute to Ru bindings of the same label value to two different FECs.

This prohibition holds even if the bindings are regarded as being at different "levels of hierarchy". In MPLS, there is no notion of having a different label space for different levels of the hierarchy; when interpreting a label, the level of the label is irrelevant.

2.15. Label Switched Path (LSP), LSP Ingress, LSP Egress

A "Label Switched Path (LSP) of level m" for a particular packet P is a sequence of routers,

<R1, ..., Rn>

with the following properties:

1. R1, the "LSP Ingress", is an LSR which pushes a label onto P's label stack, resulting in a label stack of depth m;

2. For all i, 1<i<n, P has a label stack of depth m when received by LSR Ri;

3. At no time during P's transit from R1 to R[n-1] does its label stack ever have a depth of less than m;

4. For all i, 1<i<n: Ri transmits P to R[i+1] by means of MPLS, i.e., by using the label at the top of the label stack (the level m label) as an index into an ILM;

5. For all i, 1<i<n: if a system S receives and forwards P after P is transmitted by Ri but before P is received by R[i+1] (e.g., Ri and R[i+1] might be connected via a switched data link subnetwork, and S might be one of the data link switches), then

S's forwarding decision is not based on the level m label, or on the network layer header. This may be because:

a) the decision is not based on the label stack or the network layer header at all;

b) the decision is based on a label stack on which additional labels have been pushed (i.e., on a level m+k label, where k>0).

In other words, we can speak of the level m LSP for Packet P as the sequence of routers:

1. which begins with an LSR (an "LSP Ingress") that pushes on a level m label,

2. all of whose intermediate LSRs make their forwarding decision by label Switching on a level m label,

3. which ends (at an "LSP Egress") when a forwarding decision is made by label Switching on a level m-k label, where k>0, or when a forwarding decision is made by "ordinary", non-MPLS forwarding procedures.

A consequence (or perhaps a presupposition) of this is that whenever an LSR pushes a label onto an already labeled packet, it needs to make sure that the new label corresponds to a FEC whose LSP Egress is the LSR that assigned the label which is now second in the stack.

We will call a sequence of LSRs the "LSP for a particular FEC F" if it is an LSP of level m for a particular packet P when P's level m label is a label corresponding to FEC F.

Consider the set of nodes which may be LSP ingress nodes for FEC F. Then there is an LSP for FEC F which begins with each of those nodes. If a number of those LSPs have the same LSP egress, then one can consider the set of such LSPs to be a tree, whose root is the LSP egress. (Since data travels along this tree towards the root, this may be called a multipoint-to-point tree.) We can thus speak of the "LSP tree" for a particular FEC F.

2.16. Penultimate Hop Popping

Note that according to the definitions of section 2.15, if <R1, ..., Rn> is a level m LSP for packet P, P may be transmitted from R[n-1] to Rn with a label stack of depth m-1. That is, the label stack may be popped at the penultimate LSR of the LSP, rather than at the LSP Egress.

From an architectural perspective, this is perfectly appropriate. The purpose of the level m label is to get the packet to Rn. Once R[n-1] has decided to send the packet to Rn, the label no longer has any function, and need no longer be carried.

There is also a practical advantage to doing penultimate hop popping. If one does not do this, then when the LSP egress receives a packet, it first looks up the top label, and determines as a result of that lookup that it is indeed the LSP egress. Then it must pop the stack, and examine what remains of the packet. If there is another label on the stack, the egress will look this up and forward the packet based on this lookup. (In this case, the egress for the packet's level m LSP is also an intermediate node for its level m-1 LSP.) If there is no other label on the stack, then the packet is forwarded according to its network layer destination address. Note that this would require the egress to do TWO lookups, either two label lookups or a label lookup followed by an address lookup.

If, on the other hand, penultimate hop popping is used, then when the penultimate hop looks up the label, it determines:

- that it is the penultimate hop, and

- who the next hop is.

The penultimate node then pops the stack, and forwards the packet based on the information gained by looking up the label that was previously at the top of the stack. When the LSP egress receives the packet, the label which is now at the top of the stack will be the label which it needs to look up in order to make its own forwarding decision. Or, if the packet was only carrying a single label, the LSP egress will simply see the network layer packet, which is just what it needs to see in order to make its forwarding decision.

This technique allows the egress to do a single lookup, and also requires only a single lookup by the penultimate node.

The creation of the forwarding "fastpath" in a label switching product may be greatly aided if it is known that only a single lookup is ever required:

- the code may be simplified if it can assume that only a single lookup is ever needed

- the code can be based on a "time budget" that assumes that only a single lookup is ever needed.

In fact, when penultimate hop popping is done, the LSP Egress need not even be an LSR.

However, some hardware switching engines may not be able to pop the label stack, so this cannot be universally required. There may also be some situations in which penultimate hop popping is not desirable. Therefore the penultimate node pops the label stack only if this is specifically requested by the egress node, OR if the next node in the LSP does not support MPLS. (If the next node in the LSP does support MPLS, but does not make such a request, the penultimate node has no way of knowing that it in fact is the penultimate node.)

An LSR which is capable of popping the label stack at all MUST do penultimate hop popping when so requested by its downstream LDP peer.

Initial LDP negotiations MUST allow each LSR to determine whether its neighboring LSRS are capable of popping the label stack. A LSR MUST NOT request an LDP peer to pop the label stack unless it is capable of doing so.

It may be asked whether the egress node can always interpret the top label of a received packet properly if penultimate hop popping is used. As long as the uniqueness and scoping rules of section 2.14 are obeyed, it is always possible to interpret the top label of a received packet unambiguously.

2.17. LSP Next Hop

The LSP Next Hop for a particular labeled packet in a particular LSR is the LSR which is the next hop, as selected by the NHLFE entry used for forwarding that packet.

The LSP Next Hop for a particular FEC is the next hop as selected by the NHLFE entry indexed by a label which corresponds to that FEC.

Note that the LSP Next Hop may differ from the next hop which would be chosen by the network layer routing algorithm. We will use the term "L3 next hop" when we refer to the latter.

2.18. Invalid Incoming Labels

What should an LSR do if it receives a labeled packet with a particular incoming label, but has no binding for that label? It is tempting to think that the labels can just be removed, and the packet forwarded as an unlabeled IP packet. However, in some cases, doing so could cause a loop. If the upstream LSR thinks the label is bound to an explicit route, and the downstream LSR doesn't think the label is bound to anything, and if the hop by hop routing of the unlabeled IP packet brings the packet back to the upstream LSR, then a loop is formed.

It is also possible that the label was intended to represent a route which the cannot be inferred the IP header.

Therefore, when a labeled packet is received with an invalid incoming label, it MUST be discarded, UNLESS it is determined by some means (not within the scope of the current document) that forwarding it unlabeled cannot cause any harm.

2.19. LSP Control: Ordered versus Independent

Some FECs correspond to address prefixes which are distributed via a dynamic routing algorithm. The setup of the LSPs for these FECs can be done in one of two ways: Independent LSP Control or Ordered LSP Control.

In Independent LSP Control, each LSR, upon noting that it recognizes a particular FEC, makes an independent decision to bind a label to that FEC and to distribute that binding to its LDP peers. This corresponds to the way that conventional IP datagram routing works; each node makes an independent decision as to how to treat each packet, and relies on the routing algorithm to converge rapidly so as to ensure that each datagram is correctly delivered.

In Ordered LSP Control, an LSR only binds a label to a particular FEC if it is the egress LSR for that FEC, or if it has already received a label binding for that FEC from its next hop for that FEC.

If one wants to ensure that traffic in a particular FEC follows a path with some specified set of properties (e.g., that the traffic does not traverse any node twice, that a specified amount of resources are available to the traffic, that the traffic follows an explicitly specified path, etc.) ordered control must be used. With independent control, some LSRs may begin label switching a traffic in the FEC before the LSP is completely set up, and thus some traffic in the FEC may follow a path which does not have the specified set of

properties. Ordered control also needs to be used if the recognition of the FEC is a consequence of the setting up of the corresponding LSP.

Ordered LSP setup may be initiated either by the ingress or the egress.

Ordered control and independent control are fully interoperable. However, unless all LSRs in an LSP are using ordered control, the overall effect on network behavior is largely that of independent control, since one cannot be sure that an LSP is not used until it is fully set up.

This architecture allows the choice between independent control and ordered control to be a local matter. Since the two methods interwork, a given LSR need support only one or the other. Generally speaking, the choice of independent versus ordered control does not appear to have any effect on the LDP mechanisms which need to be defined.

2.20. Aggregation

One way of partitioning traffic into FECs is to create a separate FEC for each address prefix which appears in the routing table. However, within a particular MPLS domain, this may result in a set of FECs such that all traffic in all those FECs follows the same route. For example, a set of distinct address prefixes might all have the same egress node, and label swapping might be used only to get the the traffic to the egress node. In this case, within the MPLS domain, the union of those FECs is itself a FEC. This creates a choice: should a distinct label be bound to each component FEC, or should a single label be bound to the union, and that label applied to all traffic in the union?

The procedure of binding a single label to a union of FECs which is itself a FEC (within some domain), and of applying that label to all traffic in the union, is known as "aggregation". The MPLS architecture allows aggregation. Aggregation may reduce the number of labels which are needed to handle a particular set of packets, and may also reduce the amount of LDP control traffic needed.

Given a set of FECs which are "aggregatable" into a single FEC, it is possible to (a) aggregate them into a single FEC, (b) aggregate them into a set of FECs, or (c) not aggregate them at all. Thus we can speak of the "granularity" of aggregation, with (a) being the "coarsest granularity", and (c) being the "finest granularity".

When order control is used, each LSR should adopt, for a given set of FECs, the granularity used by its next hop for those FECs.

When independent control is used, it is possible that there will be two adjacent LSRs, Ru and Rd, which aggregate some set of FECs differently.

If Ru has finer granularity than Rd, this does not cause a problem. Ru distributes more labels for that set of FECs than Rd does. This means that when Ru needs to forward labeled packets in those FECs to Rd, it may need to map n labels into m labels, where n > m. As an option, Ru may withdraw the set of n labels that it has distributed, and then distribute a set of m labels, corresponding to Rd's level of granularity. This is not necessary to ensure correct operation, but it does result in a reduction of the number of labels distributed by Ru, and Ru is not gaining any particular advantage by distributing the larger number of labels. The decision whether to do this or not is a local matter.

If Ru has coarser granularity than Rd (i.e., Rd has distributed n labels for the set of FECs, while Ru has distributed m, where n > m), it has two choices:

- It may adopt Rd's finer level of granularity. This would require it to withdraw the m labels it has distributed, and distribute n labels. This is the preferred option.

- It may simply map its m labels into a subset of Rd's n labels, if it can determine that this will produce the same routing. For example, suppose that Ru applies a single label to all traffic that needs to pass through a certain egress LSR, whereas Rd binds a number of different labels to such traffic, depending on the individual destination addresses of the packets. If Ru knows the address of the egress router, and if Rd has bound a label to the FEC which is identified by that address, then Ru can simply apply that label.

In any event, every LSR needs to know (by configuration) what granularity to use for labels that it assigns. Where ordered control is used, this requires each node to know the granularity only for FECs which leave the MPLS network at that node. For independent control, best results may be obtained by ensuring that all LSRs are consistently configured to know the granularity for each FEC. However, in many cases this may be done by using a single level of granularity which applies to all FECs (such as "one label per IP prefix in the forwarding table", or "one label per egress node").

2.21. Route Selection

Route selection refers to the method used for selecting the LSP for a particular FEC. The proposed MPLS protocol architecture supports two options for Route Selection: (1) hop by hop routing, and (2) explicit routing.

Hop by hop routing allows each node to independently choose the next hop for each FEC. This is the usual mode today in existing IP networks. A "hop by hop routed LSP" is an LSP whose route is selected using hop by hop routing.

In an explicitly routed LSP, each LSR does not independently choose the next hop; rather, a single LSR, generally the LSP ingress or the LSP egress, specifies several (or all) of the LSRs in the LSP. If a single LSR specifies the entire LSP, the LSP is "strictly" explicitly routed. If a single LSR specifies only some of the LSP, the LSP is "loosely" explicitly routed.

The sequence of LSRs followed by an explicitly routed LSP may be chosen by configuration, or may be selected dynamically by a single node (for example, the egress node may make use of the topological information learned from a link state database in order to compute the entire path for the tree ending at that egress node).

Explicit routing may be useful for a number of purposes such as policy routing or traffic engineering. With MPLS the explicit route needs to be specified at the time that labels are assigned, but the explicit route does not have to be specified with each IP packet. This makes MPLS explicit routing much more efficient than the alternative of IP source routing.

When an LSP is explicitly routed (either loosely or strictly), it is essential that packets travelling along the LSP reach its end before they revert to hop by hop routing. Otherwise inconsistent routing and loops might form.

It is not necessary for a node to be able to create an explicit route. However, in order to ensure interoperability it is necessary to ensure that either (i) Every node knows how to use hop by hop routing; or (ii) Every node knows how to create and follow an explicit route. We propose that due to the common use of hop by hop routing in networks today, it is reasonable to make hop by hop routing the default that all nodes need to be able to use.

2.22. Time-to-Live (TTL)

In conventional IP forwarding, each packet carries a "Time To Live" (TTL) value in its header. Whenever a packet passes through a router, its TTL gets decremented by 1; if the TTL reaches 0 before the packet has reached its destination, the packet gets discarded.

This provides some level of protection against forwarding loops that may exist due to misconfigurations, or due to failure or slow convergence of the routing algorithm. TTL is sometimes used for other functions as well, such as multicast scoping, and supporting the "traceroute" command. This implies that there are two TTL-related issues that MPLS needs to deal with: (i) TTL as a way to suppress loops; (ii) TTL as a way to accomplish other functions, such as limiting the scope of a packet.

When a packet travels along an LSP, it SHOULD emerge with the same TTL value that it would have had if it had traversed the same sequence of routers without having been label switched. If the packet travels along a hierarchy of LSPs, the total number of LSR-hops traversed SHOULD be reflected in its TTL value when it emerges from the hierarchy of LSPs.

The way that TTL is handled may vary depending upon whether the MPLS label values are carried in an MPLS-specific "shim" header, or if the MPLS labels are carried in an L2 header, such as an ATM header or a frame relay header.

If the label values are encoded in a "shim" that sits between the data link and network layer headers, then this shim MUST have a TTL field that SHOULD be initially loaded from the network layer header TTL field, SHOULD be decremented at each LSR-hop, and SHOULD be copied into the network layer header TTL field when the packet emerges from its LSP.

If the label values are encoded in a data link layer header (e.g., the VPI/VCI field in ATM's AAL5 header), and the labeled packets are forwarded by an L2 switch (e.g., an ATM switch), and the data link layer (like ATM) does not itself have a TTL field, then it will not be possible to decrement a packet's TTL at each LSR-hop. An LSP segment which consists of a sequence of LSRs that cannot decrement a packet's TTL will be called a "non-TTL LSP segment".

When a packet emerges from a non-TTL LSP segment, it SHOULD however be given a TTL that reflects the number of LSR-hops it traversed. In the unicast case, this can be achieved by propagating a meaningful LSP length to ingress nodes, enabling the ingress to decrement the TTL value before forwarding packets into a non-TTL LSP segment.

Sometimes it can be determined, upon ingress to a non-TTL LSP segment, that a particular packet's TTL will expire before the packet reaches the egress of that non-TTL LSP segment. In this case, the LSR at the ingress to the non-TTL LSP segment must not label switch the packet. This means that special procedures must be developed to support traceroute functionality, for example, traceroute packets may be forwarded using conventional hop by hop forwarding.

2.23. Loop Control

On a non-TTL LSP segment, by definition, TTL cannot be used to protect against forwarding loops. The importance of loop control may depend on the particular hardware being used to provide the LSR functions along the non-TTL LSP segment.

Suppose, for instance, that ATM switching hardware is being used to provide MPLS switching functions, with the label being carried in the VPI/VCI field. Since ATM switching hardware cannot decrement TTL, there is no protection against loops. If the ATM hardware is capable of providing fair access to the buffer pool for incoming cells carrying different VPI/VCI values, this looping may not have any deleterious effect on other traffic. If the ATM hardware cannot provide fair buffer access of this sort, however, then even transient loops may cause severe degradation of the LSR's total performance.

Even if fair buffer access can be provided, it is still worthwhile to have some means of detecting loops that last "longer than possible". In addition, even where TTL and/or per-VC fair queuing provides a means for surviving loops, it still may be desirable where practical to avoid setting up LSPs which loop.

The MPLS architecture will therefore provide a technique for ensuring that looping LSP segments can be detected, and a technique for ensuring that looping LSP segments are never created.

All LSRs will be required to support a common technique for loop detection. Support for the loop prevention technique is optional, though it is recommended in ATM-LSRs that have no other way to protect themselves against the effects of looping data packets. Use of the loop prevention technique, when supported, is optional.

The loop prevention technique presupposes the use of Ordered LSP Control. The loop detection technique, on the other hand, works with either Independent or Ordered LSP Control.

2.23.1. Loop Prevention

NOTE: The loop prevention technique described here is being reconsidered, and may be changed.

LSR's maintain for each of their LSP's an LSR id list. This list is a list of all the LSR's downstream from this LSR on a given LSP. The LSR id list is used to prevent the formation of switched path loops. The LSR ID list is propagated upstream from a node to its neighbor nodes. The LSR ID list is used to prevent loops as follows:

When a node, R, detects a change in the next hop for a given FEC, it asks its new next hop for a label and the associated LSR ID list for that FEC.

The new next hop responds with a label for the FEC and an associated LSR id list.

R looks in the LSR id list. If R determines that it, R, is in the list then we have a route loop. In this case, we do nothing and the old LSP will continue to be used until the route protocols break the loop. The means by which the old LSP is replaced by a new LSP after the route protocols breathe loop is described below.

If R is not in the LSR id list, R will start a "diffusion" computation [12]. The purpose of the diffusion computation is to prune the tree upstream of R so that we remove all LSR's from the tree that would be on a looping path if R were to switch over to the new LSP. After those LSR's are removed from the tree, it is safe for R to replace the old LSP with the new LSP (and the old LSP can be released).

The diffusion computation works as follows:

R adds its LSR id to the list and sends a query message to each of its "upstream" neighbors (i.e. to each of its neighbors that is not the new "downstream" next hop).

A node S that receives such a query will process the query as follows:

- If node R is not node S's next hop for the given FEC, node S will respond to node R will an "OK" message meaning that as far as node S is concerned it is safe for node R to switch over to the new LSP.

- If node R is node S's next hop for the FEC, node S will check to see if it, node S, is in the LSR id list that it received from node R. If it is, we have a route loop and S will respond with a "LOOP" message. R will unsplice the connection to S pruning S from the tree. The mechanism by which S will get a new LSP for the FEC after the route protocols break the loop is described below.

- If node S is not in the LSR id list, S will add its LSR id to the LSR id list and send a new query message further upstream. The diffusion computation will continue to propagate upstream along each of the paths in the tree upstream of S until either a loop is detected, in which case the node is pruned as described above or we get to a point where a node gets a response ("OK" or "LOOP") from each of its neighbors perhaps because none of those neighbors considers the node in question to be its downstream next hop. Once a node has received a response from each of its upstream neighbors, it returns an "OK" message to its downstream neighbor. When the original node, node R, gets a response from each of its neighbors, it is safe to replace the old LSP with the new one because all the paths that would loop have been pruned from the tree.

There are a couple of details to discuss:

- First, we need to do something about nodes that for one reason or another do not produce a timely response in response to a query message. If a node Y does not respond to a query from node X because of a failure of some kind, X will not be able to respond to its downstream neighbors (if any) or switch over to a new LSP if X is, like R above, the node that has detected the route change. This problem is handled by timing out the query message. If a node doesn't receive a response within a "reasonable" period of time, it "unsplices" its VC to the upstream neighbor that is not responding and proceeds as it would if it had received the "LOOP" message.

- We also need to be concerned about multiple concurrent routing updates. What happens, for example, when a node M receives a request for an LSP from an upstream neighbor, N, when M is in the middle of a diffusion computation i.e., it has sent a query upstream but hasn't received all the responses. Since a downstream node, node R is about to change from one LSP to another, M needs to pass to N an LSR id list corresponding to the union of the old and new LSP's if it is to avoid loops both before and after the transition. This is easily accomplished since M already has the LSR id list for the old LSP and it gets the LSR id list for the new LSP in the query message. After R

makes the switch from the old LSP to the new one, R sends a new establish message upstream with the LSR id list of (just) the new LSP. At this point, the nodes upstream of R know that R has switched over to the new LSP and that they can return the id list for (just) the new LSP in response to any new requests for LSP's. They can also grow the tree to include additional nodes that would not have been valid for the combined LSR id list.

- We also need to discuss how a node that doesn't have an LSP for a given stream at the end of a diffusion computation (because it would have been on a looping LSP) gets one after the routing protocols break the loop. If node L has been pruned from the tree and its local route protocol processing entity breaks the loop by changing L's next hop, L will request a new LSP from its new downstream neighbor which it will use once it executes the diffusion computation as described above. If the loop is broken by a route change at another point in the loop, i.e. at a point "downstream" of L, L will get a new LSP as the new LSP tree grows upstream from the point of the route change as discussed in the previous paragraph.

- Note that when a node is pruned from the tree, the switched path upstream of that node remains "connected". This is important since it allows the switched path to get "reconnected" to a downstream switched path after a route change with a minimal amount of unsplicing and resplicing once the appropriate diffusion computation(s) have taken place.

The LSR Id list can also be used to provide a "loop detection" capability. To use it in this manner, an LSR which sees that it is already in the LSR Id list for a particular FEC will immediately unsplice itself from the switched path for that FEC, and will NOT pass the LSR Id list further upstream. The LSR can rejoin a switched path for the FEC when it changes its next hop for that FEC, or when it receives a new LSR Id list from its current next hop, in which it is not contained. The diffusion computation would be omitted.

2.23.2. Interworking of Loop Control Options

The MPLS protocol architecture allows some nodes to be using loop prevention, while some other nodes are not (i.e., the choice of whether or not to use loop prevention may be a local decision). When this mix is used, it is not possible for a loop to form which includes only nodes which do loop prevention. However, it is possible for loops to form which contain a combination of some nodes which do loop prevention, and some nodes which do not.

There are at least four identified cases in which it makes sense to combine nodes which do loop prevention with nodes which do not: (i) For transition, in intermediate states while transitioning from all non-loop-prevention to all loop prevention, or vice versa; (ii) For interoperability, where one vendor implements loop prevention but another vendor does not; (iii) Where there is a mixed ATM and datagram media network, and where loop prevention is desired over the ATM portions of the network but not over the datagram portions; (iv) where some of the ATM switches can do fair access to the buffer pool on a per-VC basis, and some cannot, and loop prevention is desired over the ATM portions of the network which cannot.

Note that interworking is straightforward. If an LSR is not doing loop prevention, and it receives from a downstream LSR a label binding which contains loop prevention information, it (a) accepts the label binding, (b) does NOT pass the loop prevention information upstream, and (c) informs the downstream neighbor that the path is loop-free.

Similarly, if an LSR R which is doing loop prevention receives from a downstream LSR a label binding which does not contain any loop prevention information, then R passes the label binding upstream with loop prevention information included as if R were the egress for the specified FEC.

Optionally, a node is permitted to implement the ability of either doing or not doing loop prevention as options, and is permitted to choose which to use for any one particular LSP based on the information obtained from downstream nodes. When the label binding arrives from downstream, then the node may choose whether to use loop prevention so as to continue to use the same approach as was used in the information passed to it. Note that regardless of whether loop prevention is used the egress nodes (for any particular LSP) always initiates exchange of label binding information without waiting for other nodes to act.

2.24. Label Encodings

In order to transmit a label stack along with the packet whose label stack it is, it is necessary to define a concrete encoding of the label stack. The architecture supports several different encoding techniques; the choice of encoding technique depends on the particular kind of device being used to forward labeled packets.

2.24.1. MPLS-specific Hardware and/or Software

If one is using MPLS-specific hardware and/or software to forward labeled packets, the most obvious way to encode the label stack is to define a new protocol to be used as a "shim" between the data link layer and network layer headers. This shim would really be just an encapsulation of the network layer packet; it would be "protocol-independent" such that it could be used to encapsulate any network layer. Hence we will refer to it as the "generic MPLS encapsulation".

The generic MPLS encapsulation would in turn be encapsulated in a data link layer protocol.

The generic MPLS encapsulation should contain the following fields:

1. the label stack,

2. a Time-to-Live (TTL) field

3. a Class of Service (CoS) field

The TTL field permits MPLS to provide a TTL function similar to what is provided by IP.

The CoS field permits LSRs to apply various scheduling packet disciplines to labeled packets, without requiring separate labels for separate disciplines.

2.24.2. ATM Switches as LSRs

It will be noted that MPLS forwarding procedures are similar to those of legacy "label swapping" switches such as ATM switches. ATM switches use the input port and the incoming VPI/VCI value as the index into a "cross-connect" table, from which they obtain an output port and an outgoing VPI/VCI value. Therefore if one or more labels can be encoded directly into the fields which are accessed by these legacy switches, then the legacy switches can, with suitable software upgrades, be used as LSRs. We will refer to such devices as "ATM-LSRs".

There are three obvious ways to encode labels in the ATM cell header (presuming the use of AAL5):

1. SVC Encoding

   Use the VPI/VCI field to encode the label which is at the top of the label stack. This technique can be used in any network. With this encoding technique, each LSP is realized as an ATM SVC, and the LDP becomes the ATM "signaling" protocol. With this encoding technique, the ATM-LSRs cannot perform "push" or "pop" operations on the label stack.

2. SVP Encoding

   Use the VPI field to encode the label which is at the top of the label stack, and the VCI field to encode the second label on the stack, if one is present. This technique some advantages over the previous one, in that it permits the use of ATM "VP-switching". That is, the LSPs are realized as ATM SVPs, with LDP serving as the ATM signaling protocol.

   However, this technique cannot always be used. If the network includes an ATM Virtual Path through a non-MPLS ATM network, then the VPI field is not necessarily available for use by MPLS.

   When this encoding technique is used, the ATM-LSR at the egress of the VP effectively does a "pop" operation.

3. SVP Multipoint Encoding

   Use the VPI field to encode the label which is at the top of the label stack, use part of the VCI field to encode the second label on the stack, if one is present, and use the remainder of the VCI field to identify the LSP ingress. If this technique is used, conventional ATM VP-switching capabilities can be used to provide multipoint-to-point VPs. Cells from different packets will then carry different VCI values. As we shall see in section 2.25, this enables us to do label merging, without running into any cell interleaving problems, on ATM switches which can provide multipoint-to-point VPs, but which do not have the VC merge capability.

   This technique depends on the existence of a capability for assigning small unique values to each ATM switch.

If there are more labels on the stack than can be encoded in the ATM header, the ATM encodings must be combined with the generic encapsulation.

2.24.3. Interoperability among Encoding Techniques

If <R1, R2, R3> is a segment of a LSP, it is possible that R1 will use one encoding of the label stack when transmitting packet P to R2, but R2 will use a different encoding when transmitting a packet P to R3. In general, the MPLS architecture supports LSPs with different label stack encodings used on different hops. Therefore, when we discuss the procedures for processing a labeled packet, we speak in abstract terms of operating on the packet's label stack. When a labeled packet is received, the LSR must decode it to determine the current value of the label stack, then must operate on the label stack to determine the new value of the stack, and then encode the new value appropriately before transmitting the labeled packet to its next hop.

Unfortunately, ATM switches have no capability for translating from one encoding technique to another. The MPLS architecture therefore requires that whenever it is possible for two ATM switches to be successive LSRs along a level m LSP for some packet, that those two ATM switches use the same encoding technique.

Naturally there will be MPLS networks which contain a combination of ATM switches operating as LSRs, and other LSRs which operate using an MPLS shim header. In such networks there may be some LSRs which have ATM interfaces as well as "MPLS Shim" interfaces. This is one example of an LSR with different label stack encodings on different hops. Such an LSR may swap off an ATM encoded label stack on an incoming interface and replace it with an MPLS shim header encoded label stack on the outgoing interface.

2.25. Label Merging

Suppose that an LSR has bound multiple incoming labels to a particular FEC. When forwarding packets in that FEC, one would like to have a single outgoing label which is applied to all such packets. The fact that two different packets in the FEC arrived with different incoming labels is irrelevant; one would like to forward them with the same outgoing label. The capability to do so is known as "label merging".

Let us say that an LSR is capable of label merging if it can receive two packets from different incoming interfaces, and/or with different labels, and send both packets out the same outgoing interface with the same label. Once the packets are transmitted, the information that they arrived from different interfaces and/or with different incoming labels is lost.

Let us say that an LSR is not capable of label merging if, for any two packets which arrive from different interfaces, or with different labels, the packets must either be transmitted out different interfaces, or must have different labels.

Label merging would be a requirement of the MPLS architecture, if not for the fact that ATM-LSRs using the SVC or SVP Encodings cannot perform label merging. This is discussed in more detail in the next section.

If a particular LSR cannot perform label merging, then if two packets in the same FEC arrive with different incoming labels, they must be forwarded with different outgoing labels. With label merging, the number of outgoing labels per FEC need only be 1; without label merging, the number of outgoing labels per FEC could be as large as the number of nodes in the network.

With label merging, the number of incoming labels per FEC that a particular LSR needs is never be larger than the number of LDP adjacencies. Without label merging, the number of incoming labels per FEC that a particular LSR needs is as large as the number of upstream nodes which forward traffic in the FEC to the LSR in question. In fact, it is difficult for an LSR to even determine how many such incoming labels it must support for a particular FEC.

The MPLS architecture accommodates both merging and non-merging LSRs, but allows for the fact that there may be LSRs which do not support label merging. This leads to the issue of ensuring correct interoperation between merging LSRs and non-merging LSRs. The issue is somewhat different in the case of datagram media versus the case of ATM. The different media types will therefore be discussed separately.

2.25.1. Non-merging LSRs

The MPLS forwarding procedures is very similar to the forwarding procedures used by such technologies as ATM and Frame Relay. That is, a unit of data arrives, a label (VPI/VCI or DLCI) is looked up in a "cross-connect table", on the basis of that lookup an output port is chosen, and the label value is rewritten. In fact, it is possible to use such technologies for MPLS forwarding; LDP can be used as the "signalling protocol" for setting up the cross-connect tables.

Unfortunately, these technologies do not necessarily support the label merging capability. In ATM, if one attempts to perform label merging, the result may be the interleaving of cells from various packets. If cells from different packets get interleaved, it is

impossible to reassemble the packets. Some Frame Relay switches use cell switching on their backplanes. These switches may also be incapable of supporting label merging, for the same reason -- cells of different packets may get interleaved, and there is then no way to reassemble the packets.

We propose to support two solutions to this problem. First, MPLS will contain procedures which allow the use of non-merging LSRs. Second, MPLS will support procedures which allow certain ATM switches to function as merging LSRs.

Since MPLS supports both merging and non-merging LSRs, MPLS also contains procedures to ensure correct interoperation between them.

2.25.2. Labels for Merging and Non-Merging LSRs

An upstream LSR which supports label merging needs to be sent only one label per FEC. An upstream neighbor which does not support label merging needs to be sent multiple labels per FEC. However, there is no way of knowing a priori how many labels it needs. This will depend on how many LSRs are upstream of it with respect to the FEC in question.

In the MPLS architecture, if a particular upstream neighbor does not support label merging, it is not sent any labels for a particular FEC unless it explicitly asks for a label for that FEC. The upstream neighbor may make multiple such requests, and is given a new label each time. When a downstream neighbor receives such a request from upstream, and the downstream neighbor does not itself support label merging, then it must in turn ask its downstream neighbor for another label for the FEC in question.

It is possible that there may be some nodes which support label merging, but can only merge a limited number of incoming labels into a single outgoing label. Suppose for example that due to some hardware limitation a node is capable of merging four incoming labels into a single outgoing label. Suppose however, that this particular node has six incoming labels arriving at it for a particular FEC. In this case, this node may merge these into two outgoing labels.

Whether label merging is applicable to explicitly routed LSPs is for further study.

2.25.3. Merge over ATM

2.25.3.1. Methods of Eliminating Cell Interleave

There are several methods that can be used to eliminate the cell interleaving problem in ATM, thereby allowing ATM switches to support stream merge: :

1. VP merge, using the SVP Multipoint Encoding

   When VP merge is used, multiple virtual paths are merged into a virtual path, but packets from different sources are distinguished by using different VCs within the VP.

2. VC merge

   When VC merge is used, switches are required to buffer cells from one packet until the entire packet is received (this may be determined by looking for the AAL5 end of frame indicator).

VP merge has the advantage that it is compatible with a higher percentage of existing ATM switch implementations. This makes it more likely that VP merge can be used in existing networks. Unlike VC merge, VP merge does not incur any delays at the merge points and also does not impose any buffer requirements. However, it has the disadvantage that it requires coordination of the VCI space within each VP. There are a number of ways that this can be accomplished. Selection of one or more methods is for further study.

This tradeoff between compatibility with existing equipment versus protocol complexity and scalability implies that it is desirable for the MPLS protocol to support both VP merge and VC merge. In order to do so each ATM switch participating in MPLS needs to know whether its immediate ATM neighbors perform VP merge, VC merge, or no merge.

2.25.3.2. Interoperation: VC Merge, VP Merge, and Non-Merge

The interoperation of the various forms of merging over ATM is most easily described by first describing the interoperation of VC merge with non-merge.

In the case where VC merge and non-merge nodes are interconnected the forwarding of cells is based in all cases on a VC (i.e., the concatenation of the VPI and VCI). For each node, if an upstream neighbor is doing VC merge then that upstream neighbor requires only a single VPI/VCI for a particular stream (this is analogous to the requirement for a single label in the case of operation over frame

media). If the upstream neighbor is not doing merge, then the neighbor will require a single VPI/VCI per stream for itself, plus enough VPI/VCIs to pass to its upstream neighbors. The number required will be determined by allowing the upstream nodes to request additional VPI/VCIs from their downstream neighbors (this is again analogous to the method used with frame merge).

A similar method is possible to support nodes which perform VP merge. In this case the VP merge node, rather than requesting a single VPI/VCI or a number of VPI/VCIs from its downstream neighbor, instead may request a single VP (identified by a VPI) but several VCIs within the VP. Furthermore, suppose that a non-merge node is downstream from two different VP merge nodes. This node may need to request one VPI/VCI (for traffic originating from itself) plus two VPs (one for each upstream node), each associated with a specified set of VCIs (as requested from the upstream node).

In order to support all of VP merge, VC merge, and non-merge, it is therefore necessary to allow upstream nodes to request a combination of zero or more VC identifiers (consisting of a VPI/VCI), plus zero or more VPs (identified by VPIs) each containing a specified number of VCs (identified by a set of VCIs which are significant within a VP). VP merge nodes would therefore request one VP, with a contained VCI for traffic that it originates (if appropriate) plus a VCI for each VC requested from above (regardless of whether or not the VC is part of a containing VP). VC merge node would request only a single VPI/VCI (since they can merge all upstream traffic into a single VC). Non-merge nodes would pass on any requests that they get from above, plus request a VPI/VCI for traffic that they originate (if appropriate).

2.26. Tunnels and Hierarchy

Sometimes a router Ru takes explicit action to cause a particular packet to be delivered to another router Rd, even though Ru and Rd are not consecutive routers on the Hop-by-hop path for that packet, and Rd is not the packet's ultimate destination. For example, this may be done by encapsulating the packet inside a network layer packet whose destination address is the address of Rd itself. This creates a "tunnel" from Ru to Rd. We refer to any packet so handled as a "Tunneled Packet".

2.26.1. Hop-by-Hop Routed Tunnel

If a Tunneled Packet follows the Hop-by-hop path from Ru to Rd, we say that it is in an "Hop-by-Hop Routed Tunnel" whose "transmit endpoint" is Ru and whose "receive endpoint" is Rd.

2.26.2. Explicitly Routed Tunnel

If a Tunneled Packet travels from Ru to Rd over a path other than the Hop-by-hop path, we say that it is in an "Explicitly Routed Tunnel" whose "transmit endpoint" is Ru and whose "receive endpoint" is Rd. For example, we might send a packet through an Explicitly Routed Tunnel by encapsulating it in a packet which is source routed.

2.26.3. LSP Tunnels

It is possible to implement a tunnel as a LSP, and use label switching rather than network layer encapsulation to cause the packet to travel through the tunnel. The tunnel would be a LSP <R1, ..., Rn>, where R1 is the transmit endpoint of the tunnel, and Rn is the receive endpoint of the tunnel. This is called a "LSP Tunnel".

The set of packets which are to be sent though the LSP tunnel constitutes a FEC, and each LSR in the tunnel must assign a label to that FEC (i.e., must assign a label to the tunnel). The criteria for assigning a particular packet to an LSP tunnel is a local matter at the tunnel's transmit endpoint. To put a packet into an LSP tunnel, the transmit endpoint pushes a label for the tunnel onto the label stack and sends the labeled packet to the next hop in the tunnel.

If it is not necessary for the tunnel's receive endpoint to be able to determine which packets it receives through the tunnel, as discussed earlier, the label stack may be popped at the penultimate LSR in the tunnel.

A "Hop-by-Hop Routed LSP Tunnel" is a Tunnel that is implemented as an hop-by-hop routed LSP between the transmit endpoint and the receive endpoint.

An "Explicitly Routed LSP Tunnel" is a LSP Tunnel that is also an Explicitly Routed LSP.

2.26.4. Hierarchy: LSP Tunnels within LSPs

Consider a LSP <R1, R2, R3, R4>. Let us suppose that R1 receives unlabeled packet P, and pushes on its label stack the label to cause it to follow this path, and that this is in fact the Hop-by-hop path. However, let us further suppose that R2 and R3 are not directly connected, but are "neighbors" by virtue of being the endpoints of an LSP tunnel. So the actual sequence of LSRs traversed by P is <R1, R2, R21, R22, R23, R3, R4>.

When P travels from R1 to R2, it will have a label stack of depth 1. R2, switching on the label, determines that P must enter the tunnel. R2 first replaces the Incoming label with a label that is meaningful to R3. Then it pushes on a new label. This level 2 label has a value which is meaningful to R21. Switching is done on the level 2 label by R21, R22, R23. R23, which is the penultimate hop in the R2-R3 tunnel, pops the label stack before forwarding the packet to R3. When R3 sees packet P, P has only a level 1 label, having now exited the tunnel. Since R3 is the penultimate hop in P's level 1 LSP, it pops the label stack, and R4 receives P unlabeled.

The label stack mechanism allows LSP tunneling to nest to any depth.

2.26.5. LDP Peering and Hierarchy

Suppose that packet P travels along a Level 1 LSP <R1, R2, R3, R4>, and when going from R2 to R3 travels along a Level 2 LSP <R2, R21, R22, R3>. From the perspective of the Level 2 LSP, R2's LDP peer is R21. From the perspective of the Level 1 LSP, R2's LDP peers are R1 and R3. One can have LDP peers at each layer of hierarchy. We will see in sections 3.6 and 3.7 some ways to make use of this hierarchy. Note that in this example, R2 and R21 must be IGP neighbors, but R2 and R3 need not be.

When two LSRs are IGP neighbors, we will refer to them as "Local LDP Peers". When two LSRs may be LDP peers, but are not IGP neighbors, we will refer to them as "Remote LDP Peers". In the above example, R2 and R21 are local LDP peers, but R2 and R3 are remote LDP peers.

The MPLS architecture supports two ways to distribute labels at different layers of the hierarchy: Explicit Peering and Implicit Peering.

One performs label Distribution with one's Local LDP Peers by opening LDP connections to them. One can perform label Distribution with one's Remote LDP Peers in one of two ways:

1. Explicit Peering

   In explicit peering, one sets up LDP connections between Remote LDP Peers, exactly as one would do for Local LDP Peers. This technique is most useful when the number of Remote LDP Peers is small, or the number of higher level label bindings is large, or the Remote LDP Peers are in distinct routing areas or domains. Of course, one needs to know which labels to distribute to which peers; this is addressed in section 3.1.2.

   Examples of the use of explicit peering is found in sections 3.2.1 and 3.6.

2. Implicit Peering

   In Implicit Peering, one does not have LDP connections to one's remote LDP peers, but only to one's local LDP peers. To distribute higher level labels to ones remote LDP peers, one encodes the higher level labels as an attribute of the lower level labels, and distributes the lower level label, along with this attribute, to the local LDP peers. The local LDP peers then propagate the information to their peers. This process continues till the information reaches remote LDP peers. Note that the intermediary nodes may also be remote LDP peers.

   This technique is most useful when the number of Remote LDP Peers is large. Implicit peering does not require a n-square peering mesh to distribute labels to the remote LDP peers because the information is piggybacked through the local LDP peering. However, implicit peering requires the intermediate nodes to store information that they might not be directly interested in.

   An example of the use of implicit peering is found in section 3.3.

2.27. LDP Transport

LDP is used between nodes in an MPLS network to establish and maintain the label bindings. In order for LDP to operate correctly, LDP information needs to be transmitted reliably, and the LDP messages pertaining to a particular FEC need to be transmitted in sequence. Flow control is also required, as is the capability to carry multiple LDP messages in a single datagram.

These goals will be met by using TCP as the underlying transport for LDP.

(The use of multicast techniques to distribute label bindings is for further study.)

2.28. Multicast

This section is for further study

3. Some Applications of MPLS

3.1. MPLS and Hop by Hop Routed Traffic

One use of MPLS is to simplify the process of forwarding packets using hop by hop routing.

3.1.1. Labels for Address Prefixes

In general, router R determines the next hop for packet P by finding the address prefix X in its routing table which is the longest match for P's destination address. That is, the packets in a given FEC are just those packets which match a given address prefix in R's routing table. In this case, a FEC can be identified with an address prefix.

If packet P must traverse a sequence of routers, and at each router in the sequence P matches the same address prefix, MPLS simplifies the forwarding process by enabling all routers but the first to avoid executing the best match algorithm; they need only look up the label.

3.1.2. Distributing Labels for Address Prefixes

3.1.2.1. LDP Peers for a Particular Address Prefix

LSRs R1 and R2 are considered to be LDP Peers for address prefix X if and only if one of the following conditions holds:

1. R1's route to X is a route which it learned about via a particular instance of a particular IGP, and R2 is a neighbor of R1 in that instance of that IGP

2. R1's route to X is a route which it learned about by some instance of routing algorithm A1, and that route is redistributed into an instance of routing algorithm A2, and R2 is a neighbor of R1 in that instance of A2

3. R1 is the receive endpoint of an LSP Tunnel that is within another LSP, and R2 is a transmit endpoint of that tunnel, and R1 and R2 are participants in a common instance of an IGP, and are in the same IGP area (if the IGP in question has areas), and R1's route to X was learned via that IGP instance, or is redistributed by R1 into that IGP instance

4. R1's route to X is a route which it learned about via BGP, and R2 is a BGP peer of R1

In general, these rules ensure that if the route to a particular address prefix is distributed via an IGP, the LDP peers for that address prefix are the IGP neighbors. If the route to a particular address prefix is distributed via BGP, the LDP peers for that address prefix are the BGP peers. In other cases of LSP tunneling, the tunnel endpoints are LDP peers.

3.1.2.2. Distributing Labels

In order to use MPLS for the forwarding of normally routed traffic, each LSR MUST:

1. bind one or more labels to each address prefix that appears in its routing table;

2. for each such address prefix X, use an LDP to distribute the binding of a label to X to each of its LDP Peers for X.

There is also one circumstance in which an LSR must distribute a label binding for an address prefix, even if it is not the LSR which bound that label to that address prefix:

3. If R1 uses BGP to distribute a route to X, naming some other LSR R2 as the BGP Next Hop to X, and if R1 knows that R2 has assigned label L to X, then R1 must distribute the binding between T and X to any BGP peer to which it distributes that route.

These rules ensure that labels corresponding to address prefixes which correspond to BGP routes are distributed to IGP neighbors if and only if the BGP routes are distributed into the IGP. Otherwise, the labels bound to BGP routes are distributed only to the other BGP speakers.

These rules are intended only to indicate which label bindings must be distributed by a given LSR to which other LSRs.

3.1.3. Using the Hop by Hop path as the LSP

If the hop-by-hop path that packet P needs to follow is <R1, ..., Rn>, then <R1, ..., Rn> can be an LSP as long as:

1. there is a single address prefix X, such that, for all i, 1<=i<n, X is the longest match in Ri's routing table for P's destination address;

2. for all i, 1<i<n, Ri has assigned a label to X and distributed that label to R[i-1].

Note that a packet's LSP can extend only until it encounters a router whose forwarding tables have a longer best match address prefix for the packet's destination address. At that point, the LSP must end and the best match algorithm must be performed again.

Suppose, for example, that packet P, with destination address 10.2.153.178 needs to go from R1 to R2 to R3. Suppose also that R2 advertises address prefix 10.2/16 to R1, but R3 advertises 10.2.153/22, 10.2.154/22, and 10.2/16 to R2. That is, R2 is advertising an "aggregated route" to R1. In this situation, packet P can be label Switched until it reaches R2, but since R2 has performed route aggregation, it must execute the best match algorithm to find P's FEC.

3.1.4. LSP Egress and LSP Proxy Egress

An LSR R is considered to be an "LSP Egress" LSR for address prefix X if and only if one of the following conditions holds:

1. R1 has an address Y, such that X is the address prefix in R1's routing table which is the longest match for Y, or

2. R contains in its routing tables one or more address prefixes Y such that X is a proper initial substring of Y, but R's "LSP previous hops" for X do not contain any such address prefixes Y; that is, R2 is a "deaggregation point" for address prefix X.

An LSR R1 is considered to be an "LSP Proxy Egress" LSR for address prefix X if and only if:

1. R1's next hop for X is R2 R1 and R2 are not LDP Peers with respect to X (perhaps because R2 does not support MPLS), or

2. R1 has been configured to act as an LSP Proxy Egress for X

The definition of LSP allows for the LSP Egress to be a node which does not support MPLS; in this case the penultimate node in the LSP is the Proxy Egress.

3.1.5. The Implicit NULL Label

The Implicit NULL label is a label with special semantics which an LSR can bind to an address prefix. If LSR Ru, by consulting its ILM, sees that labeled packet P must be forwarded next to Rd, but that Rd has distributed a binding of Implicit NULL to the corresponding address prefix, then instead of replacing the value of the label on top of the label stack, Ru pops the label stack, and then forwards the resulting packet to Rd.

LSR Rd distributes a binding between Implicit NULL and an address prefix X to LSR Ru if and only if:

1. the rules of Section 3.1.2 indicate that Rd distributes to Ru a label binding for X, and

2. when the LDP connection between Ru and Rd was opened, Ru indicated that it could support the Implicit NULL label (i.e., that it can pop the label stack), and

3. Rd is an LSP Egress (not proxy egress) for X.

This causes the penultimate LSR on a LSP to pop the label stack. This is quite appropriate; if the LSP Egress is an MPLS Egress for X, then if the penultimate LSR does not pop the label stack, the LSP Egress will need to look up the label, pop the label stack, and then look up the next label (or look up the L3 address, if no more labels are present). By having the penultimate LSR pop the label stack, the LSP Egress is saved the work of having to look up two labels in order to make its forwarding decision.

However, if the penultimate LSR is an ATM switch, it may not have the capability to pop the label stack. Hence a binding of Implicit NULL may be distributed only to LSRs which can support that function.

If the penultimate LSR in an LSP for address prefix X is an LSP Proxy Egress, it acts just as if the LSP Egress had distributed a binding of Implicit NULL for X.

3.1.6. Option: Egress-Targeted Label Assignment

There are situations in which an LSP Ingress, Ri, knows that packets of several different FECs must all follow the same LSP, terminating at, say, LSP Egress Re. In this case, proper routing can be achieved by using a single label can be used for all such FECs; it is not necessary to have a distinct label for each FEC. If (and only if) the following conditions hold:

1. the address of LSR Re is itself in the routing table as a "host route", and

2. there is some way for Ri to determine that Re is the LSP egress for all packets in a particular set of FECs

Then Ri may bind a single label to all FECS in the set. This is known as "Egress-Targeted Label Assignment."

How can LSR Ri determine that an LSR Re is the LSP Egress for all packets in a particular FEC? There are a couple of possible ways:

- If the network is running a link state routing algorithm, and all nodes in the area support MPLS, then the routing algorithm provides Ri with enough information to determine the routers through which packets in that FEC must leave the routing domain or area.

- It is possible to use LDP to pass information about which address prefixes are "attached" to which egress LSRs. This method has the advantage of not depending on the presence of link state routing.

If egress-targeted label assignment is used, the number of labels that need to be supported throughout the network may be greatly reduced. This may be significant if one is using legacy switching hardware to do MPLS, and the switching hardware can support only a limited number of labels.

One possible approach would be to configure the network to use egress-targeted label assignment by default, but to configure particular LSRs to NOT use egress-targeted label assignment for one or more of the address prefixes for which it is an LSP egress. We impose the following rule:

- If a particular LSR is NOT an LSP Egress for some set of address prefixes, then it should assign labels to the address prefixes in the same way as is done by its LSP next hop for those address prefixes. That is, suppose Rd is Ru's LSP next hop for address

prefixes X1 and X2. If Rd assigns the same label to X1 and X2, Ru should as well. If Rd assigns different labels to X1 and X2, then Ru should as well.

For example, suppose one wants to make egress-targeted label assignment the default, but to assign distinct labels to those address prefixes for which there are multiple possible LSP egresses (i.e., for those address prefixes which are multi-homed.) One can configure all LSRs to use egress-targeted label assignment, and then configure a handful of LSRs to assign distinct labels to those address prefixes which are multi-homed. For a particular multi-homed address prefix X, one would only need to configure this in LSRs which are either LSP Egresses or LSP Proxy Egresses for X.

It is important to note that if Ru and Rd are adjacent LSRs in an LSP for X1 and X2, forwarding will still be done correctly if Ru assigns distinct labels to X1 and X2 while Rd assigns just one label to the both of them. This just means that R1 will map different incoming labels to the same outgoing label, an ordinary occurrence.

Similarly, if Rd assigns distinct labels to X1 and X2, but Ru assigns to them both the label corresponding to the address of their LSP Egress or Proxy Egress, forwarding will still be done correctly. Ru will just map the incoming label to the label which Rd has assigned to the address of that LSP Egress.

3.2. MPLS and Explicitly Routed LSPs

There are a number of reasons why it may be desirable to use explicit routing instead of hop by hop routing. For example, this allows routes to be based on administrative policies, and allows the routes that LSPs take to be carefully designed to allow traffic engineering (i.e., to allow intentional management of the loading of the bandwidth through the nodes and links in the network).

3.2.1. Explicitly Routed LSP Tunnels: Traffic Engineering

In some situations, the network administrators may desire to forward certain classes of traffic along certain pre-specified paths, where these paths differ from the Hop-by-hop path that the traffic would ordinarily follow. This is known as Traffic Engineering.

MPLS allows this to be easily done by means of Explicitly Routed LSP Tunnels. All that is needed is:

1. A means of selecting the packets that are to be sent into the Explicitly Routed LSP Tunnel;

2. A means of setting up the Explicitly Routed LSP Tunnel;

3. A means of ensuring that packets sent into the Tunnel will not loop from the receive endpoint back to the transmit endpoint.

If the transmit endpoint of the tunnel wishes to put a labeled packet into the tunnel, it must first replace the label value at the top of the stack with a label value that was distributed to it by the tunnel's receive endpoint. Then it must push on the label which corresponds to the tunnel itself, as distributed to it by the next hop along the tunnel. To allow this, the tunnel endpoints should be explicit LDP peers. The label bindings they need to exchange are of no interest to the LSRs along the tunnel.

3.3. Label Stacks and Implicit Peering

Suppose a particular LSR Re is an LSP proxy egress for 10 address prefixes, and it reaches each address prefix through a distinct interface.

One could assign a single label to all 10 address prefixes. Then Re is an LSP egress for all 10 address prefixes. This ensures that packets for all 10 address prefixes get delivered to Re. However, Re would then have to look up the network layer address of each such packet in order to choose the proper interface to send the packet on.

Alternatively, one could assign a distinct label to each interface. Then Re is an LSP proxy egress for the 10 address prefixes. This eliminates the need for Re to look up the network layer addresses in order to forward the packets. However, it can result in the use of a large number of labels.

An alternative would be to bind all 10 address prefixes to the same level 1 label (which is also bound to the address of the LSR itself), and then to bind each address prefix to a distinct level 2 label. The level 2 label would be treated as an attribute of the level 1 label binding, which we call the "Stack Attribute". We impose the following rules:

- When LSR Ru initially labels an untagged packet, if the longest match for the packet's destination address is X, and R's LSP next hop for X is Rd, and Rd has distributed to R1 a binding of label L1 X, along with a stack attribute of L2, then

1. Ru must push L2 and then L1 onto the packet's label stack, and then forward the packet to Rd;

2. When Ru distributes label bindings for X to its LDP peers, it must include L2 as the stack attribute.

3. Whenever the stack attribute changes (possibly as a result of a change in Ru's LSP next hop for X), Ru must distribute the new stack attribute.

Note that although the label value bound to X may be different at each hop along the LSP, the stack attribute value is passed unchanged, and is set by the LSP proxy egress.

Thus the LSP proxy egress for X becomes an "implicit peer" with each other LSR in the routing area or domain. In this case, explicit peering would be too unwieldy, because the number of peers would become too large.

3.4. MPLS and Multi-Path Routing

If an LSR supports multiple routes for a particular stream, then it may assign multiple labels to the stream, one for each route. Thus the reception of a second label binding from a particular neighbor for a particular address prefix should be taken as meaning that either label can be used to represent that address prefix.

If multiple label bindings for a particular address prefix are specified, they may have distinct attributes.

3.5. LSP Trees as Multipoint-to-Point Entities

Consider the case of packets P1 and P2, each of which has a destination address whose longest match, throughout a particular routing domain, is address prefix X. Suppose that the Hop-by-hop path for P1 is <R1, R2, R3>, and the Hop-by-hop path for P2 is <R4, R2, R3>. Let's suppose that R3 binds label L3 to X, and distributes this binding to R2. R2 binds label L2 to X, and distributes this binding to both R1 and R4. When R2 receives packet P1, its incoming label will be L2. R2 will overwrite L2 with L3, and send P1 to R3. When R2 receives packet P2, its incoming label will also be L2. R2 again overwrites L2 with L3, and send P2 on to R3.

Note then that when P1 and P2 are traveling from R2 to R3, they carry the same label, and as far as MPLS is concerned, they cannot be distinguished. Thus instead of talking about two distinct LSPs, <R1,

R2, R3> and <R4, R2, R3>, we might talk of a single "Multipoint-to-Point LSP Tree", which we might denote as <{R1, R4}, R2, R3>.

This creates a difficulty when we attempt to use conventional ATM switches as LSRs. Since conventional ATM switches do not support multipoint-to-point connections, there must be procedures to ensure that each LSP is realized as a point-to-point VC. However, if ATM switches which do support multipoint-to-point VCs are in use, then the LSPs can be most efficiently realized as multipoint-to-point VCs. Alternatively, if the SVP Multipoint Encoding (section 2.24.2) can be used, the LSPs can be realized as multipoint-to-point SVPs.

3.6. LSP Tunneling between BGP Border Routers

Consider the case of an Autonomous System, A, which carries transit traffic between other Autonomous Systems. Autonomous System A will have a number of BGP Border Routers, and a mesh of BGP connections among them, over which BGP routes are distributed. In many such cases, it is desirable to avoid distributing the BGP routes to routers which are not BGP Border Routers. If this can be avoided, the "route distribution load" on those routers is significantly reduced. However, there must be some means of ensuring that the transit traffic will be delivered from Border Router to Border Router by the interior routers.

This can easily be done by means of LSP Tunnels. Suppose that BGP routes are distributed only to BGP Border Routers, and not to the interior routers that lie along the Hop-by-hop path from Border Router to Border Router. LSP Tunnels can then be used as follows:

1. Each BGP Border Router distributes, to every other BGP Border Router in the same Autonomous System, a label for each address prefix that it distributes to that router via BGP.

2. The IGP for the Autonomous System maintains a host route for each BGP Border Router. Each interior router distributes its labels for these host routes to each of its IGP neighbors.

3. Suppose that:

    a) BGP Border Router B1 receives an unlabeled packet P,

    b) address prefix X in B1's routing table is the longest match for the destination address of P,

c) the route to X is a BGP route,

d) the BGP Next Hop for X is B2,

e) B2 has bound label L1 to X, and has distributed this binding to B1,

f) the IGP next hop for the address of B2 is I1,

g) the address of B2 is in B1's and I1's IGP routing tables as a host route, and

h) I1 has bound label L2 to the address of B2, and distributed this binding to B1.

Then before sending packet P to I1, B1 must create a label stack for P, then push on label L1, and then push on label L2.

4. Suppose that BGP Border Router B1 receives a labeled Packet P, where the label on the top of the label stack corresponds to an address prefix, X, to which the route is a BGP route, and that conditions 3b, 3c, 3d, and 3e all hold. Then before sending packet P to I1, B1 must replace the label at the top of the label stack with L1, and then push on label L2.

With these procedures, a given packet P follows a level 1 LSP all of whose members are BGP Border Routers, and between each pair of BGP Border Routers in the level 1 LSP, it follows a level 2 LSP.

These procedures effectively create a Hop-by-Hop Routed LSP Tunnel between the BGP Border Routers.

Since the BGP border routers are exchanging label bindings for address prefixes that are not even known to the IGP routing, the BGP routers should become explicit LDP peers with each other.

3.7. Other Uses of Hop-by-Hop Routed LSP Tunnels

The use of Hop-by-Hop Routed LSP Tunnels is not restricted to tunnels between BGP Next Hops. Any situation in which one might otherwise have used an encapsulation tunnel is one in which it is appropriate to use a Hop-by-Hop Routed LSP Tunnel. Instead of encapsulating the packet with a new header whose destination address is the address of the tunnel's receive endpoint, the label corresponding to the address prefix which is the longest match for the address of the tunnel's receive endpoint is pushed on the packet's label stack. The packet which is sent into the tunnel may or may not already be labeled.

If the transmit endpoint of the tunnel wishes to put a labeled packet into the tunnel, it must first replace the label value at the top of the stack with a label value that was distributed to it by the tunnel's receive endpoint. Then it must push on the label which corresponds to the tunnel itself, as distributed to it by the next hop along the tunnel. To allow this, the tunnel endpoints should be explicit LDP peers. The label bindings they need to exchange are of no interest to the LSRs along the tunnel.

3.8. MPLS and Multicast

Multicast routing proceeds by constructing multicast trees. The tree along which a particular multicast packet must get forwarded depends in general on the packet's source address and its destination address. Whenever a particular LSR is a node in a particular multicast tree, it binds a label to that tree. It then distributes that binding to its parent on the multicast tree. (If the node in question is on a LAN, and has siblings on that LAN, it must also distribute the binding to its siblings. This allows the parent to use a single label value when multicasting to all children on the LAN.)

When a multicast labeled packet arrives, the NHLFE corresponding to the label indicates the set of output interfaces for that packet, as well as the outgoing label. If the same label encoding technique is used on all the outgoing interfaces, the very same packet can be sent to all the children.

4. LDP Procedures for Hop-by-Hop Routed Traffic

4.1. The Procedures for Advertising and Using labels

In this section, we consider only label bindings that are used for traffic to be label switched along its hop-by-hop routed path. In these cases, the label in question will correspond to an address prefix in the routing table.

There are a number of different procedures that may be used to distribute label bindings. One such procedure is executed by the downstream LSR, and the others by the upstream LSR.

The downstream LSR must perform:

- The Distribution Procedure, and

- the Withdrawal Procedure.

The upstream LSR must perform:

- The Request Procedure, and

- the NotAvailable Procedure, and

- the Release Procedure, and

- the labelUse Procedure.

The MPLS architecture supports several variants of each procedure.

However, the MPLS architecture does not support all possible combinations of all possible variants. The set of supported combinations will be described in section 4.2, where the interoperability between different combinations will also be discussed.

4.1.1. Downstream LSR: Distribution Procedure

The Distribution Procedure is used by a downstream LSR to determine when it should distribute a label binding for a particular address prefix to its LDP peers. The architecture supports four different distribution procedures.

Irrespective of the particular procedure that is used, if a label binding for a particular address prefix has been distributed by a downstream LSR Rd to an upstream LSR Ru, and if at any time the attributes (as defined above) of that binding change, then Rd must inform Ru of the new attributes.

If an LSR is maintaining multiple routes to a particular address prefix, it is a local matter as to whether that LSR binds multiple labels to the address prefix (one per route), and hence distributes multiple bindings.

4.1.1.1. PushUnconditional

Let Rd be an LSR. Suppose that:

1. X is an address prefix in Rd's routing table

2. Ru is an LDP Peer of Rd with respect to X

Whenever these conditions hold, Rd must bind a label to X and distribute that binding to Ru. It is the responsibility of Rd to keep track of the bindings which it has distributed to Ru, and to make sure that Ru always has these bindings.

This procedure would be used by LSRs which are performing downstream label assignment in the Independent LSP Control Mode.

4.1.1.2. PushConditional

Let Rd be an LSR. Suppose that:

1. X is an address prefix in Rd's routing table

2. Ru is an LDP Peer of Rd with respect to X

3. Rd is either an LSP Egress or an LSP Proxy Egress for X, or Rd's L3 next hop for X is Rn, where Rn is distinct from Ru, and Rn has bound a label to X and distributed that binding to Rd.

Then as soon as these conditions all hold, Rd should bind a label to X and distribute that binding to Ru.

Whereas PushUnconditional causes the distribution of label bindings for all address prefixes in the routing table, PushConditional causes the distribution of label bindings only for those address prefixes for which one has received label bindings from one's LSP next hop, or for which one does not have an MPLS-capable L3 next hop.

This procedure would be used by LSRs which are performing downstream label assignment in the Ordered LSP Control Mode.

4.1.1.3. PulledUnconditional

Let Rd be an LSR. Suppose that:

1. X is an address prefix in Rd's routing table

2. Ru is a label distribution peer of Rd with respect to X

3. Ru has explicitly requested that Rd bind a label to X and distribute the binding to Ru

Then Rd should bind a label to X and distribute that binding to Ru. Note that if X is not in Rd's routing table, or if Rd is not an LDP peer of Ru with respect to X, then Rd must inform Ru that it cannot provide a binding at this time.

If Rd has already distributed a binding for address prefix X to Ru, and it receives a new request from Ru for a binding for address prefix X, it will bind a second label, and distribute the new binding to Ru. The first label binding remains in effect.

This procedure would be used by LSRs performing downstream-on-demand label distribution using the Independent LSP Control Mode.

4.1.1.4. PulledConditional

Let Rd be an LSR. Suppose that:

1. X is an address prefix in Rd's routing table
2. Ru is a label distribution peer of Rd with respect to X
3. Ru has explicitly requested that Rd bind a label to X and distribute the binding to Ru
4. Rd is either an LSP Egress or an LSP Proxy Egress for X, or Rd's L3 next hop for X is Rn, where Rn is distinct from Ru, and Rn has bound a label to X and distributed that binding to Rd

Then as soon as these conditions all hold, Rd should bind a label to X and distribute that binding to Ru. Note that if X is not in Rd's routing table, or if Rd is not a label distribution peer of Ru with respect to X, then Rd must inform Ru that it cannot provide a binding at this time.

However, if the only condition that fails to hold is that Rn has not yet provided a label to Rd, then Rd must defer any response to Ru until such time as it has receiving a binding from Rn.

If Rd has distributed a label binding for address prefix X to Ru, and at some later time, any attribute of the label binding changes, then Rd must redistribute the label binding to Ru, with the new attribute. It must do this even though Ru does not issue a new Request.

This procedure would be used by LSRs that are performing downstream-on-demand label allocation in the Ordered LSP Control Mode.

In section 4.2, we will discuss how to choose the particular procedure to be used at any given time, and how to ensure interoperability among LSRs that choose different procedures.

4.1.2. Upstream LSR: Request Procedure

The Request Procedure is used by the upstream LSR for an address prefix to determine when to explicitly request that the downstream LSR bind a label to that prefix and distribute the binding. There are three possible procedures that can be used.

4.1.2.1. RequestNever

Never make a request. This is useful if the downstream LSR uses the PushConditional procedure or the PushUnconditional procedure, but is not useful if the downstream LSR uses the PulledUnconditional procedure or the the PulledConditional procedures.

This procedure would be used by an LSR when downstream label distribution and Liberal Label Retention Mode are being used.

4.1.2.2. RequestWhenNeeded

Make a request whenever the L3 next hop to the address prefix changes, and one doesn't already have a label binding from that next hop for the given address prefix.

This procedure would be used by an LSR whenever Conservative Label Retention Mode is being used.

4.1.2.3. RequestOnRequest

Issue a request whenever a request is received, in addition to issuing a request when needed (as described in section 4.1.2.2). If Rd receives such a request from Ru, for an address prefix for which Rd has already distributed Ru a label, Rd shall assign a new (distinct) label, bind it to X, and distribute that binding. (Whether Rd can distribute this binding to Ru immediately or not depends on the Distribution Procedure being used.)

This procedure would be used by an LSR which doing downstream-on-

demand label distribution, but is not doing label merging, e.g., an ATM-LSR which is not capable of VC merge.

4.1.3. Upstream LSR: NotAvailable Procedure

If Ru and Rd are respectively upstream and downstream label distribution peers for address prefix X, and Rd is Ru's L3 next hop for X, and Ru requests a binding for X from Rd, but Rd replies that it cannot provide a binding at this time, then the NotAvailable procedure determines how Ru responds. There are two possible procedures governing Ru's behavior:

4.1.3.1. RequestRetry

Ru should issue the request again at a later time. That is, the requester is responsible for trying again later to obtain the needed binding. This procedure would be used when downstream-on-demand label distribution is used.

4.1.3.2. RequestNoRetry

Ru should never reissue the request, instead assuming that Rd will provide the binding automatically when it is available. This is useful if Rd uses the PushUnconditional procedure or the PushConditional procedure, i.e., if downstream label distribution is used.

4.1.4. Upstream LSR: Release Procedure

Suppose that Rd is an LSR which has bound a label to address prefix X, and has distributed that binding to LSR Ru. If Rd does not happen to be Ru's L3 next hop for address prefix X, or has ceased to be Ru's L3 next hop for address prefix X, then Rd will not be using the label. The Release Procedure determines how Ru acts in this case. There are two possible procedures governing Ru's behavior:

4.1.4.1. ReleaseOnChange

Ru should release the binding, and inform Rd that it has done so. This procedure would be used to implement Conservative Label Retention Mode.

4.1.4.2. NoReleaseOnChange

Ru should maintain the binding, so that it can use it again immediately if Rd later becomes Ru's L3 next hop for X. This procedure would be used to implement Liberal Label Retention Mode.

4.1.5. Upstream LSR: labelUse Procedure

Suppose Ru is an LSR which has received label binding L for address prefix X from LSR Rd, and Ru is upstream of Rd with respect to X, and in fact Rd is Ru's L3 next hop for X.

Ru will make use of the binding if Rd is Ru's L3 next hop for X. If, at the time the binding is received by Ru, Rd is NOT Ru's L3 next hop for X, Ru does not make any use of the binding at that time. Ru may however start using the binding at some later time, if Rd becomes Ru's L3 next hop for X.

The labelUse Procedure determines just how Ru makes use of Rd's binding.

There are three procedures which Ru may use:

4.1.5.1. UseImmediate

Ru may put the binding into use immediately. At any time when Ru has a binding for X from Rd, and Rd is Ru's L3 next hop for X, Rd will also be Ru's LSP next hop for X. This procedure is used when neither loop prevention nor loop detection are in use.

4.1.5.2. UseIfLoopFree

Ru will use the binding only if it determines that by doing so, it will not cause a forwarding loop.

If Ru has a binding for X from Rd, and Rd is (or becomes) Ru's L3 next hop for X, but Rd is NOT Ru's current LSP next hop for X, Ru does NOT immediately make Rd its LSP next hop. Rather, it initiates a loop prevention algorithm. If, upon the completion of this algorithm, Rd is still the L3 next hop for X, Ru will make Rd the LSP next hop for X, and use L as the outgoing label.

This procedure is used when loop prevention is in use.

The loop prevention algorithm to be used is still under

consideration.

4.1.5.3. UseIfLoopNotDetected

This procedure is the same as UseImmediate, unless Ru has detected a loop in the LSP. If a loop has been detected, Ru will discard packets that would otherwise have been labeled with L and sent to Rd.

This procedure is used when loop detection, but not loop prevention, is in use.

This will continue until the next hop for X changes, or until the loop is no longer detected.

4.1.6. Downstream LSR: Withdraw Procedure

In this case, there is only a single procedure.

When LSR Rd decides to break the binding between label L and address prefix X, then this unbinding must be distributed to all LSRs to which the binding was distributed.

It is desirable, though not required, that the unbinding of L from X be distributed by Rd to a LSR Ru before Rd distributes to Ru any new binding of L to any other address prefix Y, where X != Y. If Ru learns of the new binding of L to Y before it learns of the unbinding of L from X, and if packets matching both X and Y are forwarded by Ru to Rd, then for a period of time, Ru will label both packets matching X and packets matching Y with label L.

The distribution and withdrawal of label bindings is done via a label distribution protocol, or LDP. LDP is a two-party protocol. If LSR R1 has received label bindings from LSR R2 via an instance of an LDP, and that instance of that protocol is closed by either end (whether as a result of failure or as a matter of normal operation), then all bindings learned over that instance of the protocol must be considered to have been withdrawn.

As long as the relevant LDP connection remains open, label bindings that are withdrawn must always be withdrawn explicitly. If a second label is bound to an address prefix, the result is not to implicitly withdraw the first label, but to bind both labels; this is needed to support multi-path routing. If a second address prefix is bound to a label, the result is not to implicitly withdraw the binding of that label to the first address prefix, but to use that label for both address prefixes.

4.2. MPLS Schemes: Supported Combinations of Procedures

Consider two LSRs, Ru and Rd, which are label distribution peers with respect to some set of address prefixes, where Ru is the upstream peer and Rd is the downstream peer.

The MPLS scheme which governs the interaction of Ru and Rd can be described as a quintuple of procedures: <Distribution Procedure, Request Procedure, NotAvailable Procedure, Release Procedure, labelUse Procedure>. (Since there is only one Withdraw Procedure, it need not be mentioned.) A "*" appearing in one of the positions is a wild-card, meaning that any procedure in that category may be present; an "N/A" appearing in a particular position indicates that no procedure in that category is needed.

Only the MPLS schemes which are specified below are supported by the MPLS Architecture. Other schemes may be added in the future, if a need for them is shown.

4.2.1. TTL-capable LSP Segments

If Ru and Rd are MPLS peers, and both are capable of decrementing a TTL field in the MPLS header, then the MPLS scheme in use between Ru and Rd must be one of the following:

1. <PushUnconditional, RequestNever, N/A, NoReleaseOnChange, UseImmediate>

   This is downstream label distribution with independent control, liberal label retention mode, and no loop detection.

2. <PushUnconditional, RequestNever, N/A, NoReleaseOnChange, UseIfLoopNotDetected>

   This is downstream label distribution with independent control, liberal label retention, and loop detection.

3. <PushConditional, RequestWhenNeeded, RequestNoRetry, ReleaseOnChange, *>

   This is downstream label distribution with ordered control and conservative label retention mode. Loop prevention and loop detection are optional.

4. <PushConditional, RequestNever, N/A, NoReleaseOnChange, *>

   This is downstream label distribution with ordered control and liberal label retention mode. Loop prevention and loop detection are optional.

4.2.2. Using ATM Switches as LSRs

The procedures for using ATM switches as LSRs depends on whether the ATM switches can realize LSP trees as multipoint-to-point VCs or VPs.

Most ATM switches existing today do NOT have a multipoint-to-point VC-switching capability. Their cross-connect tables could easily be programmed to move cells from multiple incoming VCs to a single outgoing VC, but the result would be that cells from different packets get interleaved.

Some ATM switches do support a multipoint-to-point VC-switching capability. These switches will queue up all the incoming cells from an incoming VC until a packet boundary is reached. Then they will transmit the entire sequence of cells on the outgoing VC, without allowing cells from any other packet to be interleaved.

Many ATM switches do support a multipoint-to-point VP-switching capability, which can be used if the Multipoint SVP label encoding is used.

4.2.2.1. Without Label Merging

Suppose that R1, R2, R3, and R4 are ATM switches which do not support label merging, but are being used as LSRs. Suppose further that the L3 hop-by-hop path for address prefix X is <R1, R2, R3, R4>, and that packets destined for X can enter the network at any of these LSRs. Since there is no multipoint-to-point capability, the LSPs must be realized as point-to-point VCs, which means that there needs to be three such VCs for address prefix X: <R1, R2, R3, R4>, <R2, R3, R4>, and <R3, R4>.

Therefore, if R1 and R2 are MPLS peers, and either is an LSR which is implemented using conventional ATM switching hardware (i.e., no cell interleave suppression), the MPLS scheme in use between R1 and R2 must be one of the following:

1. <PulledUnconditional, RequestOnRequest, RequestRetry, ReleaseOnChange, UseImmediate>

   This is downstream-on-demand label distribution with independent control and conservative label retention mode, without loop prevention or detection.

2. <PulledUnconditional, RequestOnRequest, RequestRetry, ReleaseOnChange, UseIfLoopNotDetected>

   This is downstream-on-demand label distribution with independent control and conservative label retention mode, with loop detection.

3. <PulledConditional, RequestOnRequest, RequestNoRetry, ReleaseOnChange, *>

   This is downstream-on-demand label distribution with ordered control (initiated by the ingress), conservative label retention mode, and optional loop detection or loop prevention.

   The use of the RequestOnRequest procedure will cause R4 to distribute three labels for X to R3; R3 will distribute 2 labels for X to R2, and R2 will distribute one label for X to R1.

4.2.2.2. With Label Merging

If R1 and R2 are MPLS peers, at least one of which is an ATM-LSR which supports label merging, then the MPLS scheme in use between R1 and R2 must be one of the following:

1. <PulledConditional, RequestOnRequest, RequestNoRetry, ReleaseOnChange, *>

   This is downstream-on-demand label distribution with

   <PushConditional, RequestWhenNeeded, RequestNoRetry, *, *>

   <PushUnconditional, RequestNever, N/A, NoReleaseOnChange, UseImmediate>

   The first of these is an ordered control scheme. The second is is the "downstream" variant of independent control. The third is the "conservative downstream-on-demand" variant of independent control.

4.2.3. Interoperability Considerations

It is easy to see that certain quintuples do NOT yield viable MPLS schemes. For example:

- <PulledUnconditional, RequestNever, *, *, *>
  <PulledConditional, RequestNever, *, *, *>

  In these MPLS schemes, the downstream LSR Rd distributes label bindings to upstream LSR Ru only upon request from Ru, but Ru never makes any such requests. Obviously, these schemes are not viable, since they will not result in the proper distribution of label bindings.

- <*, RequestNever, *, *, ReleaseOnChange>

  In these MPLS schemes, Rd releases bindings when it isn't using them, but it never asks for them again, even if it later has a need for them. These schemes thus do not ensure that label bindings get properly distributed.

In this section, we specify rules to prevent a pair of LDP peers from adopting procedures which lead to infeasible MPLS Schemes. These rules require the exchange of information between LDP peers during the initialization of the LDP connection between them.

1. Each must state whether it is an ATM-LSR, and if so, whether it has cell interleave suppression (i.e., VC merging).

2. If Rd is an ATM switch without cell interleave suppression, it must state whether it intends to use the PulledUnconditional procedure or the Pulledconditional procedure. If the former, Ru MUST use the RequestRetry procedure; if the latter, Ru MUST use the RequestNoRetry procedure.

3. If Ru is an ATM switch without cell interleave suppression, it must state whether it intends to use the RequestRetry or the RequestNoRetry procedure. If Rd is an ATM switch without cell interleave suppression, Rd is not bound by this, and in fact Ru MUST adopt Rd's preferences. However, if Rd is NOT an ATM switch without cell interleave suppression, then if Ru chooses RequestRetry, Rd must use PulledUnconditional, and if Ru chooses RequestNoRetry, Rd MUST use PulledConditional.

4. If Rd is an ATM switch with cell interleave suppression, it must specify whether it prefers to use PushConditional, PushUnconditional, or PulledConditional. If Ru is not an ATM switch without cell interleave suppression, it must then use

RequestWhenNeeded and RequestNoRetry, or else RequestNever and NoReleaseOnChange, respectively.

5. If Ru is an ATM switch with cell interleave suppression, it must specify whether it prefers to use RequestWhenNeeded and RequestNoRetry, or else RequestNever and NoReleaseOnChange. If Rd is NOT an ATM switch with cell interleave suppression, it must then use either PushConditional or PushUnconditional, respectively.

5. Security Considerations

Security considerations are not discussed in this version of this draft.

6. Authors' Addresses

Eric C. Rosen
Cisco Systems, Inc.
250 Apollo Drive
Chelmsford, MA, 01824
E-mail: erosen@cisco.com

Arun Viswanathan
Lucent Technologies
101 Crawford Corner Rd., #4D-537
Holmdel, NJ 07733
732-332-5163
E-mail: arunv@dnrc.bell-labs.com

Ross Callon
IronBridge Networks
55 Hayden Avenue,
Lexington, MA 02173
+1-781-402-8017
E-mail: rcallon@ironbridgenetworks.com

7. References

[1] "A Framework for Multiprotocol Label Switching", R.Callon, P.Doolan, N.Feldman, A.Fredette, G.Swallow, and A.Viswanathan, work in progress, Internet Draft <draft-ietf-mpls-framework-02.txt>, November 1997.

[2] "ARIS: Aggregate Route-Based IP Switching", A. Viswanathan, N. Feldman, R. Boivie, R. Woundy, work in progress, Internet Draft <draft-viswanathan-aris-overview-00.txt>, March 1997.

[3] "ARIS Specification", N. Feldman, A. Viswanathan, work in progress, Internet Draft <draft-feldman-aris-spec-00.txt>, March 1997.

[4] "Tag Switching Architecture - Overview", Rekhter, Davie, Katz, Rosen, Swallow, Farinacci, work in progress, Internet Draft <draft-rekhter-tagswitch-arch-00.txt>, January, 1997.

[5] "Tag distribution Protocol", Doolan, Davie, Katz, Rekhter, Rosen, work in progress, Internet Draft <draft-doolan-tdp-spec-01.txt>, May, 1997.

[6] "Use of Tag Switching with ATM", Davie, Doolan, Lawrence, McGloghrie, Rekhter, Rosen, Swallow, work in progress, Internet Draft <draft-davie-tag-switching-atm-01.txt>, January, 1997.

[7] "Label Switching: Label Stack Encodings", Rosen, Rekhter, Tappan, Farinacci, Fedorkow, Li, Conta, work in progress, Internet Draft <draft-ietf-mpls-label-encaps-01.txt>, February, 1998.

[8] "Partitioning Tag Space among Multicast Routers on a Common Subnet", Farinacci, work in progress, internet draft <draft-farinacci-multicast-tag-part-00.txt>, December, 1996.

[9] "Multicast Tag Binding and Distribution using PIM", Farinacci, Rekhter, work in progress, internet draft <draft-farinacci-multicast-tagsw-00.txt>, December, 1996.

[10] "Toshiba's Router Architecture Extensions for ATM: Overview", Katsube, Nagami, Esaki, RFC 2098, February, 1997.

[11] "Loop-Free Routing Using Diffusing Computations", J.J. Garcia-Luna-Aceves, IEEE/ACM Transactions on Networking, Vol. 1, No. 1, February 1993.