Network Working Group
Internet Draft
Expiration Date: February 1998

Eric C. Rosen
Cisco Systems, Inc.

Arun Viswanathan
IBM Corp.

Ross Callon
Ascend Communications, Inc.

August 1997

A Proposed Architecture for MPLS

draft-ietf-mpls-arch-00.txt

Status of this Memo

Abstract

This internet draft contains a draft protocol architecture for multiprotocol label switching (MPLS). The proposed 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

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 analyzes the packet header, and runs a network layer routing algorithm. The next hop for a packet is chosen 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 packet forwarding space into "forwarding equivalence classes (FECs)". The second maps these FECs to a next hop. Multiple network layer headers which get mapped into the same FEC are indistinguishable, as far as the forwarding decision is concerned. The set of packets belonging to the same FEC, traveling from a common node, will follow the same path and be forwarded in the

same manner (for example, by being placed in a common queue) towards the destination. This set of packets following the same path, belonging to the same FEC (and therefore being forwarded in a common manner) may be referred to as a "stream".

In IP forwarding, multiple packets are typically assigned to the same Stream by a particular router 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.

In MPLS, the mapping from packet headers to stream is performed just once, as the packet enters the network. The stream 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 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. 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. In MPLS, this can also be inferred from the label, so that no further header analysis is needed.

The fact that a packet is assigned to a Stream 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 Stream 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.

aggregate stream	synonym of "stream"
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	stream merge, 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 stream, usually of local significance.

label information base	the database of information containing label bindings
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 streams of data to be forwarded.
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 stream.
label switching router	an MPLS node which is capable of forwarding native L3 packets
merge point	the node at which multiple streams and switched paths are combined into a single stream sent over a single path.
Mlabel	abbreviation for MPLS label
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 placed in a short MPLS shim header used to identify streams
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
stream	an aggregate of one or more flows, treated as one aggregate for the purpose of forwarding in L2 and/or L3 nodes (e.g., may be described using a single label). In many cases a stream may be the aggregate of a very large number of flows. Synonymous with "aggregate stream".
stream merge	the merging of several smaller streams into a larger stream, such that for some or all of the path the larger stream can be referred to using a single label.
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	stream merge when it is specifically applied to VCs, specifically so as to allow multiple VCs to merge into one single VC
VP merge	stream merge when it is applied to VPs, specifically so as to allow multiple VPs to merge into one single VP. In this case the VCIs need to be unique. 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
STN	Stream to NHLFE Map
IGP	Interior Gateway Protocol
ILM	Incoming Label Map
IP	Internet Protocol
LIB	Label Information Base
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 stream. The label is based on the stream or forwarding equivalence class that a packet is assigned to. The label does not directly encode the network layer address, and is based on the network layer address only to the extent that the forwarding equivalence class is based on the address.

If Ru and Rd are neighboring LSRs, they may agree to use label L to represent Stream S for packets which are sent from Ru to Rd. That is, they can agree to a "mapping" between label L and Stream S for packets moving from Ru to Rd. As a result of such an agreement, L becomes Ru's "outgoing label" corresponding to Stream S for such packets; L becomes Rd's "incoming label" corresponding to Stream S for such packets.

Note that L does not necessarily correspond to Stream S for any packets other than those which are being sent from Ru to Rd. Also, L is not an inherently meaningful value and does not have any network-wide value; the particular value assigned to L gets its meaning solely from the agreement between Ru and Rd.

Sometimes it may be difficult or even impossible for Rd to tell that an arriving packet carrying label L comes from Ru, rather than from some other LSR. In such cases, Rd must make sure that the mapping from label to FEC is one-to-one. That is, in such cases, Rd must not agree with Ru1 to use L for one purpose, while also agreeing with some other LSR Ru2 to use L for a different purpose.

The scope of labels could be unique per interface, or unique per MPLS node, or unique in a network. If labels are unique within a network, no label swapping needs to be performed in the MPLS nodes in that domain. The packets are just label forwarded and not label swapped. The possible use of labels with network-wide scope is FFS.

2.2. Upstream and Downstream LSRs

Suppose Ru and Rd have agreed to map label L to Stream S, for packets sent from Ru to Rd. Then with respect to this mapping, Ru is the "upstream LSR", and Rd is the "downstream LSR".

The notion of upstream and downstream relate to agreements between nodes of the label values to be assigned for packets belonging to a particular Stream that might be traveling from an upstream node to a downstream node. This is independent of whether the routing protocol actually will cause any packets to be transmitted in that particular direction. Thus, Rd is the downstream LSR for a particular mapping for label L if it recognizes L-labeled packets from Ru as being in Stream S. This may be true even if routing does not actually forward packets for Stream S between nodes Rd and Ru, or if routing has made Ru downstream of Rd along the path which is actually used for packets in Stream S.

2.3. Labeled Packet

A "labeled packet" is a packet into which a label has been encoded. The encoding can be done by means of an encapsulation which exists specifically for this purpose, or by placing the label in an available location in either of the data link or network layer headers. Of course, the encoding technique must be agreed to by the entity which encodes the label and the entity which decodes the label.

2.4. Label Assignment and Distribution; Attributes

For unicast traffic in the MPLS architecture, the decision to bind a particular label L to a particular Stream S is made by the LSR which is downstream with respect to that mapping. The downstream LSR then informs the upstream LSR of the mapping. Thus labels are "downstream-assigned", and are "distributed upstream".

A particular mapping of label L to Stream S, distributed by Rd to Ru, may have associated "attributes". If Ru, acting as a downstream LSR, also distributes a mapping of a label to Stream S, then under certain

conditions, it may be required to also distribute the corresponding attribute that it received from Rd.

2.5. Label Distribution Protocol (LDP)

A Label Distribution Protocol (LDP) is a set of procedures by which one LSR informs another of the label/Stream mappings it has made. Two LSRs which use an LDP to exchange label/Stream mapping information are known as "LDP Peers" with respect to the mapping information they exchange; 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 mappings, but not with respect to some other set of mappings.)

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

2.6. 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".

At a particular LSR, the decision as to how to forward a labeled packet is always based exclusively on the label at the top of the stack.

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 (sections 2.19.3 and 2.19.4).

2.7. 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 examine and operate on the encapsulated packet. This may be a lower level label, or 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".

2.8. 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.

2.9. Stream-to-NHLFE Map (STN)

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

2.10. 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 Stream. 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.

2.11. 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 LSRs,

<R1, ..., Rn>

with the following properties:

1. R1, the "LSP Ingress", 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 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 LSRs:

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.

Note that according to these definitions, 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. This is appropriate, since the level m label has served its function of getting the packet to Rn, and Rn's forwarding decision cannot be made until the level m label is popped. If the label stack is not popped by R[n-1], then Rn must do two label lookups; this is an overhead which is best avoided. However, some hardware switching engines may not be able to pop the label stack.

The penultimate node pops the label stack only if this is specifically requested by the egress node. Having the penultimate node pop the label stack has an implication on the assignment of labels: For any one node Rn, operating at level m in the MPLS

hierarchy, there may be some LSPs which terminate at that node (i.e., for which Rn is the egress node) and some other LSPs which continue beyond that node (i.e., for which Rn is an intermediate node). If the penultimate node R[n-1] pops the stack for those LSPs which terminate at Rn, then node R[n] will receive some packets for which the top of the stack is a level m label (i.e., packets destined for other egress nodes), and some packets for which the top of the stack is a level m-1 label (i.e., packets for which Rn is the egress). This implies that in order for node R[n-1] to pop the stack, node Rn must assign labels such that level m and level m-1 labels are distinguishable (i.e., use unique values across multiple levels of the MPLS hierarchy).

Note that if m = 1, the LSP Egress may receive an unlabeled packet, and in fact need not even be capable of supporting MPLS. In this case, assuming that we are using globally meaningful IP addresses, the confusion of labels at multiple levels is not possible. However, it is possible that the label may still be of value for the egress node. One example is that the label may be used to assign the packet to a particular Forwarding Equivalence Class (for example, to identify the packet as a high priority packet). Another example is that the label may assign the packet to a particular virtual private network (for example, the virtual private network may make use of local IP addresses, and the label may be necessary to disambiguate the addresses). Therefore even when there is only a single label value the stack is nonetheless popped only when requested by the egress node.

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

2.12. 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 Stream is the next hop as selected by the NHLFE entry indexed by a label which corresponds to that Stream.

2.13. Route Selection

Route selection refers to the method used for selecting the LSP for a particular stream. 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 the path for a stream. This is the normal mode today with existing datagram IP networks. A hop by hop routed LSP refers to an LSP whose route is selected using hop by hop routing.

An explicitly routed LSP is an LSP where, at a given LSR, the LSP next hop is not chosen by each local node, but rather is chosen by a single node (usually the ingress or egress node of the LSP). The sequence of LSRs followed by an explicit routing LSP may be chosen by configuration, or by a protocol selected 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 allowing policy routing and/or facilitating 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 implies that explicit routing with MPLS is relatively efficient (when compared with the efficiency of explicit routing for pure datagrams).

For any one LSP (at any one level of hierarchy), there are two possible options: (i) The entire LSP may be hop by hop routed from ingress to egress; (ii) The entire LSP may be explicit routed from ingress to egress. Intermediate cases do not make sense: In general, an LSP will be explicit routed specifically because there is a good reason to use an alternative to the hop by hop routed path. This implies that if some of the nodes along the path follow an explicit route but some of the nodes make use of hop by hop routing, then inconsistent routing will result and loops (or severely inefficient paths) may form.

For this reason, it is important that if an explicit route is specified for an LSP, then that route must be followed. Note that it is relatively simple to *follow* an explicit route which is specified in a LDP setup. We therefore propose that the LDP specification require that all MPLS nodes implement the ability to follow an explicit route if this is specified.

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.

2.14. 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 should have a TTL field that is initially loaded from the network layer header TTL field, is decremented at each LSR-hop, and is copied into the network layer header TTL field when the packet emerges from its LSP.

If the label values are encoded in an L2 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). This implies that unless the data link layer itself has a TTL field (unlike ATM), 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.

2.15. 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.

2.15.1. Loop Prevention

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 stream, it asks its new next hop for a label and the associated LSR ID list for that stream.

The new next hop responds with a label for the stream 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 stream, 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 stream, 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 stream 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 stream will immediately unsplice itself from the switched path for that stream, and will NOT pass the LSR Id list further upstream. The LSR can rejoin a switched path for the stream when it changes its next hop for that stream, 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.

2.15.2. 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 mapping which contains loop prevention information, it (a) accepts the label mapping, (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 mapping which does not contain any loop prevention information, then R passes the label mapping upstream with loop prevention information included as if R were the egress for the specified stream.

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 mapping 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 mapping information without waiting for other nodes to act.

2.16. Merging and Non-Merging LSRs

Merge allows multiple upstream LSPs to be merged into a single downstream LSP. When implemented by multiple nodes, this results in the traffic going to a particular egress nodes, based on one particular Stream, to follow a multipoint to point tree (MPT), with the MPT rooted at the egress node and associated with the Stream. This can have a significant effect on reducing the number of labels that need to be maintained by any one particular node.

If merge was not used at all it would be necessary for each node to provide the upstream neighbors with a label for each Stream for each upstream node which may be forwarding traffic over the link. This implies that the number of labels needed might not in general be known a priori. However, the use of merge allows a single label to be

used per Stream, therefore allowing label assignment to be done in a common way without regard for the number of upstream nodes which will be using the downstream LSP.

The proposed MPLS protocol architecture supports LSP merge, while allowing nodes which do not support LSP merge. This leads to the issue of ensuring correct interoperation between nodes which implement merge and those which do not. 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.

2.16.1. Stream Merge

Let us say that an LSR is capable of Stream Merge 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. This in effect takes two incoming streams and merges them into one. 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 Stream Merge 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.

An LSR which is capable of Stream Merge (a "Merging LSR") needs to maintain only one outgoing label for each FEC. AN LSR which is not capable of Stream Merge (a "Non-merging LSR") may need to maintain as many as N outgoing labels per FEC, where N is the number of LSRs in the network. Hence by supporting Stream Merge, an LSR can reduce its number of outgoing labels by a factor of O(N). Since each label in use requires the dedication of some amount of resources, this can be a significant savings.

2.16.2. 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

Stream Merge capability. In ATM, if one attempts to perform Stream Merge, 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 Stream Merge, 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.

2.16.3. Labels for Merging and Non-Merging LSRs

An upstream LSR which supports Stream Merge needs to be sent only one label per FEC. An upstream neighbor which does not support Stream Merge 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 Stream Merge, 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 Stream Merge, 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 merge, but have a limited number of upstream streams which may be merged into a single downstream streams. Suppose for example that due to some hardware limitation a node is capable of merging four upstream LSPs into a single downstream LSP. Suppose however, that this particular node has six upstream LSPs arriving at it for a particular Stream. In this case, this node may merge these into two downstream LSPs (corresponding to two labels that need to be obtained from the downstream neighbor). In this case, the normal operation of the LDP implies that the downstream neighbor will supply this node with a single label for the Stream. This node can then ask its downstream neighbor for one additional label for the Stream, implying that the node will thereby obtain the required two labels.

The interaction between explicit routing and merge is FFS.

2.16.4. Merge over ATM

2.16.4.1. 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

   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 FFS.

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.

2.16.4.2. 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).

2.17. LSP Control: Egress versus Local

There is a choice to be made regarding whether the initial setup of LSPs will be initiated by the egress node, or locally by each individual node.

When LSP control is done locally, then each node may at any time pass label bindings to its neighbors for each FEC recognized by that node. In the normal case that the neighboring nodes recognize the same FECs, then nodes may map incoming labels to outgoing labels as part of the normal label swapping forwarding method.

When LSP control is done by the egress, then initially only the

egress node passes label bindings to its neighbors corresponding to any FECs which leave the MPLS network at that egress node. Other nodes wait until they get a label from downstream for a particular FEC before passing a corresponding label for the same FEC to upstream nodes.

With local control, since each LSR is (at least initially) independently assigning labels to FECs, it is possible that different LSRs may make inconsistent decisions. For example, an upstream LSR may make a coarse decision (map multiple IP address prefixes to a single label) while its downstream neighbor makes a finer grain decision (map each individual IP address prefix to a separate label). With downstream label assignment this can be corrected by having LSRs withdraw labels that it has assigned which are inconsistent with downstream labels, and replace them with new consistent label assignments.

Even with egress control it is possible that the choice of egress node may change, or the egress may (based on a change in configuration) change its mind in terms of the granularity which is to be used. This implies the same mechanism will be necessary to allow changes in granularity to bubble up to upstream nodes. The choice of egress or local control may therefore effect the frequency with which this mechanism is used, but will not effect the need for a mechanism to achieve consistency of label granularity. Generally speaking, the choice of local versus egress control does not appear to have any effect on the LDP mechanisms which need to be defined.

Egress control and local control can interwork in a very straightforward manner (although some of the advantages ascribed to egress control may be lost, see appendices A and B). With either approach, (assuming downstream label assignment) the egress node will initially assign labels for particular FECs and will pass these labels to its neighbors. With either approach these label assignments will bubble upstream, with the upstream nodes choosing labels that are consistent with the labels that they receive from downstream. The difference between the two approaches is therefore primarily an issue of what each node does prior to obtaining a label assignment for a particular FEC from downstream nodes: Does it wait, or does it assign a preliminary label under the expectation that it will (probably) be correct?

Regardless of which method is used (local control or egress control) each node needs to know (possibly by configuration) what granularity to use for labels that it assigns. Where egress control is used, this requires each node to know the granularity only for streams which leave the MPLS network at that node. For local control, in order to avoid the need to withdraw inconsistent labels, each node in the

network would need to be configured consistently to know the granularity for each stream. However, in many cases this may be done by using a single level of granularity which applies to all streams (such as "one label per IP prefix in the forwarding table"). The choice between local control versus egress control could similarly be left as a configuration option.

Future versions of the MPLS architecture will need to choose between three options: (i) Requiring local control; (ii) Requiring egress control; or (iii) Allowing a choice of local control or egress control. Arguments for local versus egress control are contained in appendices A and B.

2.18. Granularity

When forwarding by label swapping, a stream of packets following a stream arriving from upstream may be mapped into an equal or coarser grain stream. However, a coarse grain stream (for example, containing packets destined for a short IP address prefix covering many subnets) cannot be mapped directly into a finer grain stream (for example, containing packets destined for a longer IP address prefix covering a single subnet). This implies that there needs to be some mechanism for ensuring consistency between the granularity of LSPs in an MPLS network.

The method used for ensuring compatibility of granularity may depend upon the method used for LSP control.

When LSP control is local, it is possible that a node may pass a coarse grain label to its upstream neighbor(s), and subsequently receive a finer grain label from its downstream neighbor. In this case the node has two options: (i) It may forward the corresponding packets using normal IP datagram forwarding (i.e., by examination of the IP header); (ii) It may withdraw the label mappings that it has passed to its upstream neighbors, and replace these with finer grain label mappings.

When LSP control is egress based, the label setup originates from the egress node and passes upstream. It is therefore straightforward with this approach to maintain equally-grained mappings along the route.

2.19. 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".

2.19.1. 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.

2.19.2. 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.

2.19.3. 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 becomes a Stream, and each LSR in the tunnel must assign a label to that Stream (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.

2.19.4. 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.

2.19.5. 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 mappings 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.

2.20. LDP Transport

LDP is used between nodes in an MPLS network to establish and maintain the label mappings. 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. This may potentially be accomplished either by using an existing reliable transport protocol such as TCP, or by specifying reliability mechanisms as part of LDP (for example, the reliability mechanisms which are defined in IDRP could potentially be "borrowed" for use with LSP). The precise means for accomplishing transport reliability with LSP are for further study, but will be specified by the MPLS Protocol Architecture before the architecture may be considered complete.

2.21. 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.

2.21.1. 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.

This section is not intended to rule out the use of alternative mechanisms in network environments where such alternatives may be appropriate.

2.21.2. 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, so multipoint-to-point VPs can be provided without any cell interleaving problems.

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. This does presuppose that it be possible to tell, when reassembling the ATM cells into packets, whether the generic encapsulation is also present.

2.21.3. 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.

2.22. 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 Stream are just those packets which match a given address prefix in R's routing table. In this case, a Stream 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 mapping 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 mapping 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 mapping 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 to indicate which label mappings must be distributed by a given LSR to which other LSRs, NOT to indicate the conditions under which the distribution is to be made. That is discussed in section 2.17.

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 advertises 10.2.153/22, 10.2.154/22, and 10.2/16 to R3. 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 Stream.

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 POP Label

The POP 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 mapping of the POP label 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 mapping between the POP label 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 mapping for X, and

2. when the LDP connection between Ru and Rd was opened, Ru indicated that it could support the POP label, 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 POP label mapping 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 the POP label 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 Streams 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 Streams; it is not necessary to have a distinct label for each Stream. 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 Streams

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 Stream? 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 Stream 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 mappings 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 mapping, 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 mapping 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 mappings 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 mapping 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 mappings for a particular address prefix are specified, they may have distinct attributes.

3.5. LSPs may be 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 mapping to R2. R2 binds label L2 to X, and distributes this mapping 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", 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 2.21) 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 mapping 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 mapping 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 mappings 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 mappings 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 mapping 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 mapping 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

This section is FFS.

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
IBM Corp.
17 Skyline Drive
Hawthorne NY 10532
914-784-3273
E-mail: arunv@vnet.ibm.com

Ross Callon
Ascend Communications, Inc.
1 Robbins Road
Westford, MA 01886
508-952-7412
E-mail: rcallon@casc.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-01.txt>, July 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] "ARIS Support for LAN Media Switching", S. Blake, A. Ghanwani, W. Pace, V. Srinivasan, work in progress, Internet Draft <draft-blake-aris-lan-00.txt>, March 1997.

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

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

[7] "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.

[8] "Label Switching: Label Stack Encodings", Rosen, Rekhter, Tappan, Farinacci, Fedorkow, Li, work in progress, Internet Draft <draft-rosen-tag-stack-02.txt>, June, 1997.

[9] "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.

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

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

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

Appendix A Why Egress Control is Better

This section is written by Arun Viswanathan.

It is demonstrated here why egress control is a necessary and sufficient mechanism for the LDP, and therefore is the optimal method for setting up LSPs.

The necessary condition is established by citing counter examples that can be achieved *only* by egress control. It's also established why these typical scenarios are vital requirements for a multiprotocol LDP. The sufficiency part is established by proving

that egress control subsumes the local control.

Then finally, some discussions are made to mitigate concerns expressed against not having local control. It is shown that local control has clearly undesirable properties which may lead to severe scalability and robustness problems. It is also shown that in having both egress control and local control simultaneously in a network leads to interoperability problems and how local control abrogates the essential benefits of egress control.

A complete and self-contained case is presented here that clearly establishes that egress control is the preponderant mechanism for LDP, and it suffices to support egress control alone as the distribution paradigm.

A.1 Definition of an Egress

A node is identified as an "egress" for a Stream, if:

1) it's at a routing boundary for that Stream,
2) the next hop for that Stream is non-MPLS,
3) the Stream is directly attached or the node itself.

Nodes that satisfy conditions 1 or 2 for Streams, will by default start behaving as egress for those streams. Note that conditions 1 and 2 can be learned dynamically. For condition 3, nodes will not by default act as an egress for themselves or directly attached networks. If this condition is made the default, the LSPs setup by egress control will create LSPs that are identical to the LSPs created by local control.

A.2 Overview of Egress Control

When a node is an egress for a Stream, it originates a LSP setup message for that particular Stream. The setup message is sent to all MPLS neighbors, except the next hop neighbor. Each of these messages to the neighbors carry an appropriate label for that Stream. When a node in a MPLS domain receives a setup message from a neighbor for a particular Stream, it checks if that neighbor is the next hop for the given Stream. If so, it propagates the message to all its MPLS neighbors, except the next hop from which the message arrived. If not, the node may keep the label provided in the setup message for future use or negatively acknowledge the node that sent the message to release the label assignment. But it must not forward the setup message from the incorrect next hop to any of its neighbors. This flooding scheme is similar in mechanism to Reverse Path Multicast.

When a next hop for a Stream changes due to change in network

topology, or a new node joins the topology, the node is locally appended to the existing LSP, without requiring egress intervention. The node may either request the label mapping from the new next hop, or use the previously stored (but unused) label from that next hop. In the former case, the new next hop immediately responds with a label mapping for that Stream if it has its own downstream mapping for that Stream.

A.3 Why Egress Control is Necessary

There are some important situations in which egress control is necessary:

- Shutting off an LSP

  If for some reason a network administrator requires to "shut off" a LSP setup for a particular Stream, s/he can configure the egress node for that Stream for the desired result. Note that the requirement to shut off an LSP is a very fundamental one. If a destination has network layer reachability but no MPLS layer reachability (because of a problem in MPLS layer), shutting off an LSP provides the only means to reach that destination. This mode of operation can be used by LSRs in a network that aren't a sink for large amounts of data. These LSRs usually require an occasional telnet or network management traffic. It's important to provide the capability that such nodes in a network can be accessed through hop-by-hop connectivity avoiding the MPLS layer optimization. The reachability is more important than optimization in instances like this. The MPLS architecture MUST provide this capability.

  Note that this is only possible in local control when each node in an entire network is configured to shut off a LSP setup for a particular Stream. Such is neither desirable nor scalable.

- Egress Aggregation

  In some networks, due to the absence of routing summarization, aggregation may not be possible through routing information. However, with Egress control, it is possible to aggregate *all* Streams that exit the network through a common egress node with a single LSP. This is achieved easily because the egress simply can use the same label for all Streams.

  Such is simply not possible with the Local control; with local knowledge LSRs cannot map several Streams to a single label because it is unknown if Streams will diverge at some subsequent downstream node.

The egress aggregation works for both distance vector protocols and link state protocols; it is protocol independent. Note that when using VP switching in conjunction with some distance vector protocols it becomes very essential that such aggregation be possible, as there are many vendor switches that don't have VC merging capability, and have limited VP switching capability. The egress control provides such vendors with a level-playing field to compete with MPLS products. Moreover, this capability can be very useful in enterprise networks; where several legacy LANs at a site can be aggregated to the egress LSR at that site. Furthermore, this approach can drastically reduce signalling and LSP state maintenance overheads in the entire network.

- Loop Prevention

  The loop-prevention mechanism only works from the egress node for multipoint-to-point LSPs, since the loop prevention mechanism requires the list of LSR nodes through which the setup message has already traversed in order to identify and prevent LSP loops.

  A loop prevention scheme is not possible through local control.

- De-aggregation

  Egress control provides the capability to de-aggregate one or more Streams from an aggregated Stream. For example, if a network is aggregating all CIDRs of an EBGP node into a single LSP, with egress control, a specific CIDR from this bundle can be given its own dedicated LSP. This enables one to apply special policies to specific CIDRs when required.

  In the local control this can be achieved only by configuring every node in the network with specific de-aggregation information and the associated policy. This approach can lead severe scalability problems.

- Unique Labels

  As is known, when using VP merging, all ingresses must have unique VCI values to prevent cell interleaving. With egress control, it is possible to distribute unique VCI values to the ingress nodes, avoiding the need to configure each ingress node. The egress node can pick a unique VCI for each ingress node. Another benefit of egress control is that each egress can be configured with a unique label value in the case of egress aggregation (as described above). Since the label value is unique, the same label value can be used on all the segments of a LSP. This enables one to identify anywhere in a network each LSP

that is associated with a certain egress node, thus easing network debugging.

This again, is not possible in the local control because of the lack of a single coordinating node.

A.4 Examples that work better through egress control

Local control needs to propagate attributes that come from the downstream node to all upstream nodes. This behavior itself can be LIKENED to the egress control. Nevertheless, the local control can achieve these only in a severely inefficient manner. Since each node only knows of local information, it creates and distributes an LSP with incorrect attributes. As each node learns of new downstream attributes, a correction is made as the attributes are propagated upstream again. This can lead to a worst case of O(n-squared) setup messages to create a single LSP, where n is the number of nodes in a LSP.

In the egress control, the attribute distribution is achieved during initial LSP setup, with a single message from the egress to ingresses.

- TTL/Traceroute

  The ingress requires a proper LSP hop-count value to decrement TTL in packets that use a particular LSP, in environments such as ATM which do not have a TTL equivalent. This simulates the TTL decrement which exists in an IP network, and also enables scoping utilities, such as traceroute, to work as they do today in IP networks. In egress control, the LSP hop-count is known at the ingress as a by-product of the LSP setup message, since an LSP setup message traverses from egress to ingress, and increments the hop-count at each node along the path.

- MTU

  When the MTU at the egress node is smaller than the MTU at some of the ingress nodes, packets originated at those ingress nodes will be dropped when they reach the egress node. Hosts not using MTU discovery have no means to recover from this. However, similar to the hop-count, the minimum LSP MTU can be propagated to the ingresses via egress control LSP setup messages, enabling the ingress to do fragmentation when required.

- Implicit Peering

  Implicit peering is the mechanism through which higher level stack labels are communicated to the ingress nodes. These label values are piggybacked in the LSP setup messages. This works best with egress control; when the egress creates the setup message, it can piggyback the stack labels at the same time.

- ToS/COS Based LSPs

  When certain LSPs require higher or lower precedence or priority through a network, the single egress node for that LSP can be configured with the required priority and this can be communicated in the egress control LSP setup message. In the local control, each and every node in the network must be configured per LSP to achieve the same result.

The local control initially distributes labels to its neighbors willy-nilly, and then waits for attributes to come through egress control. Thus, local control is completely dependent on egress control to provide complete functional operation to LSPs. Otherwise, local control requires that attributes be configured through the entire network for each Stream. This is the most compelling argument that local control is *not sufficient*; or conversely, egress control is necessary. This demonstrates egress control subsumes the local control. Moreover, distribution of labels without associated attributes may not be appropriate and may lead to undesired results.

A.5 Egress Control is Sufficient

The argument for sufficiency is proved by demonstrating that required LSPs can be created with egress control, and this is not the case with local control.

The egress control can create an LSP for every route entry made by the routing protocols:

1. A route can be learned from another routing domain, in which case the LSR at the routing domain will act as an egress for the route and originate an LSP setup for that route.

2. A route can be a locally attached network or the LSR itself may be a host route. In this case, the LSR to which such a route is attached originates an LSP setup message.

3. An LSR with a non-MPLS next-hop behaves as an egress for all those route whose next-hop is the non-MPLS neighbor.

These three above methods can create an LSP for each route entry in a network. Moreover, policy specific LSPs, as described previously, can *only* be achieved with egress control. Thus, egress control is necessary and sufficient for creating LSPs. QED.

A.6 Discussions

A.6.1 Is Local control faster than Egress control?

During topology changes, such as links going down, coming up, change in link cost, etc, there is no difference in setup latency between Egress Control and Local control. This is due to the fact that the node (Ru) which undergoes a change in next-hop for a Stream immediately requests a label assignment from the new next hop node (Rd). The new next hop node then immediately supplies the label mapping for the requested Stream. As explained in the Egress Control Method section, the node Ru may already have stored label assignments from the node Rd, in which case node Ru can immediately splice itself to the multipoint-to-point tree. Hence, new nodes are spliced into existing LSPs locally. In the scenario where a network initially learns of a new route, although the Local control may setup LSPs faster than the Egress control, this difference in latency has no perceived advantage. Since routing itself may take several seconds to propagate and converge on the new route information, the potential latency of egress control is small as compared to the routing protocol propagation time, and the initial setup time at route propagation time is unimportant since these are long lived LSPs.

Moreover, the hurried distribution of labels in local control may not carry much meaning because:

4. The associated attributes are not applied or propagated to the ingress.

5. While the ingress may believe it has an LSP, in reality the packets may be blackholed in the middle of the network if the full LSP is not established.

6. Policy based LSPs, which can only be achieved via egress control as described above, may undo an un-used label assignment established by local control.

A.6.2 Scalability and Robustness

It has been alleged that the egress control does not have the

scalability and robustness properties required by distributed processing. However, the egress uses a root distribution paradigm commonly used by many other standard routing protocols. For example, in the case of OSPF, LSAs are flooded through a domain originating at the "egress", where the difference being that the flooding in the case of OSPF is contained through a sequence number and in the Egress control it is contained by the next hop validation. In the case of PIM (and some other multicast protocols), the distribution mechanism is in fact exactly similar. Even in BGP with route reflection, updates originate at the root and traverse a tree structure to reach the peers, as opposed to a n-square mesh. The commonality is the distribution paradigm, in which the distribution originates at the root of a tree and traverses the branches till it reaches all the leaves. None of the above mentioned protocols have scalability or robustness problems because of the distribution paradigm.

The ONLY concern expressed against to counter Egress control is that if the setup message does not propagate upstream from a certain node, then the sub-tree upstream of that node will not be added into the LSP. It's a reasonable concern, but further analysis shows that it's not a realistic problem. The impact of this problem compared to the impact of a similar problem in local control are exactly the same when LSRs employed in a MPLS domain have little or no forwarding capabilities (for example, ATM LSRs), since in both cases, packets are blackholed. In fact, in the egress control the packets for afflicted LSPs will be dropped right at the ingress, while with local control the packets will be dropped at the point of breakage, causing packets to unnecessarily traverse part way through the network. When reasonable forwarding capability exists in the MPLS domain, with the egress control the packets may be forwarded hop-by-hop till the point where the LSP setup ended. Whereas in case of local control, the packets will label switched till the point of breakage and hop-by-hop forwarded till the LSP segment resumes. Since egress control has advantages when there is no forwarding capability, and local control is has advantages when there is forwarding capability, there is an equal tradeoff between them, and thus, neither is superior or inferior in this regard. This latter case is simply a loss in optimization, since the network has reasonable forwarding capabilities. Hence the robustness issue is not a problem in either types of networks. As mentioned before, the local control is dependent on egress control for distributing attributes. The attribute distribution could then also face the same problem of stalled propagation, which would lead to erroneous LSP setup. So, the local control can also be seen as afflicted with this problem, if it exists.

Moreover, if stalled propagation were truly a problem, there are other schemes in MPLS that would face the same issue. For example,

the label distribution through PIM, Explicit Route setup, and RSVP would also not work, and therefore should be withdrawn :-).

Note that exhaustion of label space cannot stall the propagation of messages to the upstream nodes. Appropriate indications can be given to the upstream nodes in the setup message that no label allocation was made because of exhaustion of label space, so that correct action can be taken at the upstream nodes, and yet the LSP setup would continue.

A.6.3 Conclusion

The attempt here is not to deride the local control, but since one method subsumes the features and properties of the other, then why support both and complicate implementation, interoperability and maintenance? In fact RFC1925 says, "In protocol design, perfection has been reached not when there is nothing left to add, but when there is nothing left to take away". A usual diplomatic resolution for such controversy is to make accommodations for both. We feel that it's a poor choice of architecture to support both. That is why we feel strongly that this must be evaluated by the MPLS WG.

In a way, controlling the network behavior as to which LSP are formed, which Streams map to which LSPs, and the associated attributes, can be compared to applying policies at the edges of an AS. This is precisely what the egress control provides, a rich and varied policy control at the egress node of LSPs.

Appendix B Why Local Control is Better

This section is written by Eric Rosen.

The remaining area of dispute between advocates of "local control" and advocates of "egress control" is relatively small. In particular, there is agreement on the following points:

1. If LSR R1's next hop for address prefix X is LSR R2, and R2 is in a different area or in a different routing domain than R1, then R1 may assign and distribute a label for X, even if R2 has not done so.

   This means that even under egress control, the border routers in one autonomous system do not have to wait, before distributing labels, for any downstream routers which are in other autonomous systems.

2. If LSR R1's next hop for address prefix X is LSR R2, but R1 receives a label mapping for X from LSR R3, then R1 may remember R3's mapping. If, at some later time, R3 becomes R1's next hop for S, then (if R1 is not using loop prevention) R1 may immediately begin using R3 as the LSP next hop for S, using the remembered mapping from R3.

3. Attributes which are passed upstream from the egress may change over time, as a result of reconfiguration of the egress, or of other events. This means that even if egress control is used, LSRs must be able to accept attribute changes on existing LSPs; attributes are not fixed when the LSP is first constructed, nor does a change in attributes require a new LSP to be constructed.

The dispute is centered on the situation in which the following conditions hold:

- LSR R1's next hop for address prefix X is within the same administrative domain as R1, and

- R1's next hop for X has not distributed to R1 a label for X, and

- R1 has not yet distributed to its neighbors any labels for X.

With local control, R1 is permitted to distribute a label for X to its neighbors; with egress control it is not.

From an implementation perspective, the difference then between egress control and local control is relatively small. Egress control simply creates an additional state in the label distribution process, and prohibits label distribution in that state.

From the perspective of network behavior, however, this difference is a bit more significant:

- Egress control adds latency to the initial construction of an LSP, because the path must be set up serially, node by node from the egress. With local control, all LSRs along the path may perform their setup activities in parallel.

- Egress control adds additional interdependencies among nodes, as there is something that one node cannot do until some other node does something else first, which it cannot do until some other node does something first, etc. This is problematical for a number of reasons.

- In robust system design, one tries to avoid such interdependencies, since they always bring along robustness and scalability problems.

- In some situations, it is advantageous for a node to use MPLS, even if some node downstream is not functioning properly and hence not assigning labels as it should.

These disadvantages might be tolerable if there is some significant problem which can be solved by egress control, but not by local control. So it is worth looking to see if there is such a problem.

There are a number of situations in which it may be desirable for an LSP Ingress node to know certain attributes of the LSP, e.g., the number of hops in the LSP. It is sometimes claimed that obtaining such information requires the use of egress control. However, this is not true. Any attribute of an LSP is liable to change after the LSP exists. Procedures to detect and communicate the change must exist. These procedures CANNOT be tied to the initial construction of the LSP, since they must execute after the LSP has already been constructed. The ability to pass control information upstream along a path towards an ingress node does not presuppose anything about the procedures used to construct the path.

The fundamental issue separating the advocates of egress control from the advocates of local control is really a network management issue. To advocates of egress control, setting up an LSP for a particular address prefix is analogous to setting up a PVC in an ATM network. When setting up a PVC, one goes to one of the PVC endpoints and enters certain configuration information. Similarly, one might think that to set up an LSP for a particular address prefix, one goes to the LSR which is the egress for that address prefix, and enters configuration information. This allows the network administrator complete control of which address prefixes are assigned LSPs and which are not. And if this is one's management model, egress control does simplify the configuration issues.

On the other hand, if one's model is that the LSPs get set up automatically by the network, as a result of the operation of the routing algorithm, then egress control is of no utility at all. When one hears the claim that "egress control allow you to control your network from a few nodes", what is really being claimed is "egress control simplifies the job of manually configuring all the LSPs in your network". Of course, if you don't intend to manually configure all the LSPs in your network, this is irrelevant.

So before an egress control scheme is adopted, one should ask whether complete manual configuration of the set of address prefixes which

get assigned LSPs is necessary. That is, is this capability needed to solve a real problem?

It is sometimes claimed that egress control is needed if one wants to conserve labels by assigning a single label to all address prefixes which have the same egress. This is not true. If the network is running a link state routing algorithm, each LSR already knows which address prefixes have a common egress, and hence can assign a common label. If the network is running a distance vector routing protocol, information about which address prefixes have a common egress can be made to "bubble up" from the egress, using LDP, even if local control is used.

It is only in the case where the number of available labels is so small that their use must be manually administered that egress control has an advantage. It may be arguable that egress control should be an option that can be used for the special cases in which it provides value. In most cases, there is no reason to have it at all.