

Network Working Group  
INTERNET DRAFT  
<[draft-ietf-mpls-framework-01.txt](#)>

R. Callon  
Ascend Communications  
P. Doolan  
Cisco Systems  
N. Feldman  
IBM Corp.  
A. Fredette  
Bay Networks  
G. Swallow  
Cisco Systems  
A. Viswanathan  
IBM Corp.  
July 30, 1997  
Expires Jan. 30, 1998

## A Framework for Multiprotocol Label Switching

### Status of this Memo

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as ``work in progress.''

To learn the current status of any Internet-Draft, please check the ``[1id-abstracts.txt](#)'' listing contained in the Internet-Drafts Shadow Directories on [ds.internic.net](#) (US East Coast), [nic.nordu.net](#) (Europe), [ftp.isi.edu](#) (US West Coast), or [munnari.oz.au](#) (Pacific Rim). Distribution of this memo is unlimited.

### Abstract

This document discusses technical issues and requirements for the Multiprotocol Label Switching working group. This is an initial draft document, which will evolve and expand over time. It is the intent of this document to produce a coherent description of all significant approaches which were and are being considered by the working group. Selection of specific approaches, making choices regarding engineering tradeoffs, and detailed protocol specification, are outside of the scope of this framework document.

Note that this document is at an early stage, and that most of the detailed technical discussion is only in a rough form. Additional

INTERNET DRAFT

A Framework for MPLS

July 30, 1997

text will be provided over time from a number of sources. A small amount of the text in this document may be redundant with the proposed protocol architecture for MPLS. This redundancy will be reduced over time, with the overall discussion of issues moved to be in this document, and the selection of specific approaches and specification of the protocol contained in the protocol architecture and other related documents.

## Acknowledgments

The ideas and text in this document have been collected from a number of sources and comments received. We would like to thank Jim Luciani, Andy Malis, Yakov Rekhter, Eric Rosen, and Vijay Srinivasan for their inputs and ideas.

## 1. Introduction and Requirements

### 1.1 Overview of MPLS

The primary goal of the MPLS working group is to standardize a base technology that integrates the label swapping forwarding paradigm with network layer routing. This base technology (label swapping) is expected to improve the price/performance of network layer routing, improve the scalability of the network layer, and provide greater flexibility in the delivery of (new) routing services (by allowing new routing services to be added without a change to the forwarding paradigm).

The initial MPLS effort will be focused on IPv4 and IPv6. However, the core technology will be extendible to multiple network layer protocols (e.g., IPX, Appletalk, DECnet, CLNP). MPLS is not confined to any specific link layer technology, it can work with any media over which Network Layer packets can be passed between network layer entities.

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 provides a simple "core" set of mechanisms which can be applied in several ways to provide a rich functionality. The core effort

includes:

a) Semantics assigned to a stream label:

- Labels are associated with specific streams of data;

Callon et al.

Expires January 30, 1998

[Page 2]

---

INTERNET DRAFT

A Framework for MPLS

July 30, 1997

b) Forwarding Methods:

- Forwarding is simplified by the use of short fixed length labels to identify streams
- Forwarding may require simple functions such as looking up a label in a table, swapping labels, and possibly decrementing and checking a TTL.
- In some cases MPLS may make direct use of underlying layer 2 forwarding, such as is provided by ATM or Frame Relay equipment.

c) Label Distribution Methods:

- Allow nodes to determine which labels to use for specific streams
- This may use some sort of control exchange, and/or be piggybacked on a routing protocol

The MPLS working group will define the procedures and protocols used to assign significance to the forwarding labels and to distribute that information between cooperating MPLS forwarders.

## 1.2 Requirements

- MPLS forwarding MUST simplify packet forwarding in order to do the following:
  - lower cost of high speed forwarding
  - improve forwarding performance
- MPLS core technologies MUST be general with respect to data link technologies (i.e., work over a very wide range of underlying data

links). Specific optimizations for particular media MAY be considered.

- MPLS core technologies MUST be compatible with a wide range of routing protocols, and MUST be capable of operating independently of the underlying routing protocols. It has been observed that considerable optimizations can be achieved in some cases by small enhancements of existing protocols. Such enhancements MAY be considered in the case of IETF standard routing protocols, and if appropriate, coordinated with the relevant working group(s).
- Routing protocols which are used in conjunction with MPLS might

be based on distributed computation. As such, during routing transients, these protocols may compute forwarding paths which potentially contain loops. MPLS MUST provide protocol mechanisms to either prevent the formation of loops and /or contain the amount of (networking) resources that can be consumed due to the presence of loops.

- MPLS forwarding MUST allow "aggregate forwarding" of user data; i.e., allow streams to be forwarded as a unit and ensure that an identified stream takes a single path, where a stream may consist of the aggregate of multiple flows of user data. MPLS SHOULD provide multiple levels of aggregation support (e.g., from individual end to end application flows at one extreme, to aggregates of all flows passing through a specified switch or router at the other extreme).
- MPLS MUST support operations, administration, and maintenance facilities at least as extensive as those supported in current IP networks. Current network management and diagnostic tools SHOULD continue to work in order to provide some backward compatibility. Where such tools are broken by MPLS, hooks MUST be supplied to allow equivalent functionality to be created.
- MPLS core technologies MUST work with both unicast and multicast streams.
- The MPLS core specifications MUST clearly state how MPLS operates in a hierarchical network.

- Scalability issues MUST be considered and analyzed during the definition of MPLS. Very scaleable solutions MUST be sought.
- MPLS core technologies MUST be capable of working with  $O(n)$  streams to switch all best-effort traffic, where  $n$  is the number of nodes in a MPLS domain. MPLS protocol standards MUST be capable of taking advantage of hardware that supports stream merging where appropriate. Note that  $O(n\text{-squared})$  streams or VCs might also be appropriate for use in some cases.
- The core set of MPLS standards, along with existing Internet standards, MUST be a self-contained solution. For example, the proposed solution MUST NOT require specific hardware features that do not commonly exist on network equipment at the time that the standard is complete. However, the solution MAY make use of additional optional hardware features (e.g., to optimize performance).
- The MPLS protocol standards MUST support multipath routing and

Callon et al.

Expires January 30, 1998

[Page 4]

INTERNET DRAFT

A Framework for MPLS

July 30, 1997

forwarding.

- MPLS MUST be compatible with the IETF Integrated Services Model, including RSVP.
- It MUST be possible for MPLS switches to coexist with non MPLS switches in the same switched network. MPLS switches SHOULD NOT impose additional configuration on non-MPLS switches.
- MPLS MUST allow "ships in the night" operation with existing layer 2 switching protocols (e.g., ATM Forum Signaling) (i.e., MPLS must be capable of being used in the same network which is also simultaneously operating standard layer 2 protocols).
- The MPLS protocol MUST support both topology-driven and traffic/request-driven label assignments.

### [1.3](#) Terminology

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 L3 packets which are forwarded in the same manner (e.g., over the same path, with the same forwarding treatment). A forwarding equivalence class is therefore the set of L3 packets which could safely be mapped to the same label. Note that there may be reasons that packets from a single forwarding equivalence class may be mapped to multiple labels (e.g., when stream merge is not used).

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 locally significant identifier which is used to identify a stream

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 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. In the case that the multiple paths are not combined prior to the egress node, then the egress node becomes the merge point.

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

#### shortcut VC

a VC set up as a result of an NHRP query and response

#### 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

#### [1.4](#) Acronyms and Abbreviations

DLCI                      Data Link Circuit Identifier

FEC                        Forwarding Equivalence Class

ISP

Internet Service Provider

Callon et al.

Expires January 30, 1998

[Page 9]

---

INTERNET DRAFT

A Framework for MPLS

July 30, 1997

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
NHC	Next Hop (NHRP) Client
NHS	Next Hop (NHRP) Server
VC	Virtual Circuit
VCID	Virtual Circuit Identifier
VPI	Virtual Path Identifier

## [2. Discussion of Core MPLS Components](#)

### [2.1 The Basic Routing Approach](#)

Routing is accomplished through the use of standard L3 routing protocols, such as OSPF and BGP. The information maintained by the L3 routing protocols is then used to distribute labels to neighboring nodes that are used in the forwarding of packets as described below. In the case of ATM networks, the labels that are distributed are VPI/VCIDs and a separate protocol (i.e., PNNI) is not necessary for the establishment of VCs for IP forwarding.

The topological scope of a routing protocol (i.e. routing domain) and the scope of label switching MPLS-capable nodes may be different. For example, MPLS-knowledgeable and MPLS-ignorant nodes, all of which are OSPF routers, may be co-resident in an area. In the case that neighboring routers know MPLS, labels can be exchanged and used.

Neighboring MPLS routers may use configured PVCs or PVPs to tunnel through non-participating ATM or FR switches.

## [2.2](#) Labels

In addition to the single routing protocol approach discussed above, the other key concept in the basic MPLS approach is the use of short fixed length labels to simply user data forwarding.

### [2.2.1](#) Label Semantics

It is important that the MPLS solutions are clear about what semantics (i.e., what knowledge of the state of the network) is implicit in the use of labels for forwarding user data packets or cells.

At the simplest level, a label may be thought of as nothing more than a shorthand for the packet header, in order to index the forwarding decision that a router would make for the packet. In this context, the label is nothing more than a shorthand for an aggregate stream of user data.

This observation leads to one possible very simple interpretation that the "meaning" of the label is a strictly local issue between two neighboring nodes. With this interpretation: (i) MPLS could be employed between any two neighboring nodes for forwarding of data between those nodes, even if no other nodes in the network participate in MPLS; (ii) When MPLS is used between more than two nodes, then the operation between any two neighboring nodes could be interpreted as independent of the operation between any other pair of nodes. This approach has the advantage of semantic simplicity, and of being the closest to pure datagram forwarding. However this approach (like pure datagram forwarding) has the disadvantage that when a packet is forwarded it is not known whether the packet is being forwarded into a loop, into a black hole, or towards links which have

inadequate resources to handle the traffic flow. These disadvantages are necessary with pure datagram forwarding, but are optional design choices to be made when label switching is being used.

There are cases where it would be desirable to have additional knowledge implicit in the existence of the label. For example, one approach to avoiding loops (see section x.x below) involves signaling the label distribution along a path before packets are forwarded on that path. With this approach the fact that a node has a label to use for a particular IP packet would imply the knowledge that following the label (including label swapping at subsequent nodes) leads to a non-looping path which makes progress towards the destination (something which is usually, but not necessarily always true when using pure datagram routing). This would of course require some sort of label distribution/setup protocol which signals along the path being setup before the labels are available for packet forwarding.

However, there are also other consequences to having additional semantics associated with the label: specifically, procedures are needed to ensure that the semantics are correct. For example, if the fact that you have a label for a particular destination implies that there is a loop-free path, then when the path changes some procedures are required to ensure that it is still loop free. Another example of semantics which could be implicit in a label is the identity of the higher level protocol type which is encoded using that label value.

In either case, the specific value of a label to use for a stream is strictly a local issue; however the decision about whether to use the label may be based on some global (or at least wider scope) knowledge that, for example, the label-switched path is loop-free and/or has the appropriate resources.

A similar example occurs in ATM networks: With standard ATM a signaling protocol is used which both reserves resources in switches along the path, and which ensures that the path is loop-free and terminates at the correct node. Thus implicit in the fact that an ATM node has a VPI/VCI for forwarding a particular piece of data is the knowledge that the path has been set up successfully.

Another similar examples occurs with multipoint to point trees over ATM (see section xx below), where the multipoint to point tree uses a VP, and cell interleave at merge points in the tree is handled by

giving each source on the tree a distinct VCI within the VP. In this case, the fact that each source has a known VPI/VCI to use needs to (implicitly or explicitly) imply the knowledge that the VCI assigned to that source is unique within the context of the VP.

In general labels are used to optimize how the system works, not to control how the system works. For example, the routing protocol determines the path that a packet follows. The presence or absence of a label assignment should not effect the path of a L3 packet. Note however that the use of labels may make capabilities such as explicit routes, loadsharing, and multipath more efficient.

### 2.2.2 Label Granularity

Labels are used to create a simple forwarding paradigm. The essential element in assigning a label is that the device which will be using the label to forward packets will be forwarding all packets with the same label in the same way. If the packet is to be forwarded solely by looking at the label, then at a minimum, all packets with the same incoming label must be forwarded out the same port(s) with the same encapsulation(s), and with the same next hop label (if any).

The term "forwarding equivalence class" is used to refer to a set of L3 packets which are all forwarded in the same manner by a particular LSR (for example, the IP packets in a forwarding equivalence class may be destined for the same egress from an MPLS network, and may be associated with the same QoS class). A forwarding equivalence class is therefore the set of L3 packets which could safely be mapped to the same label. Note that there may be reasons that packets from a single forwarding equivalence class may be mapped to multiple labels (e.g., when stream merge is not used).

Note that the label could also mean "ignore this label and forward based on what is contained within," where within one might find a label (if a stack of labels is used) or a layer 3 packet.

For IP unicast traffic, the granularity of a label allows various levels of aggregation in a Label Information Base (LIB). At one end of the spectrum, a label could represent a host route (i.e. the full 32 bits of IP address). If a router forwards an entire CIDR prefix

in the same way, it may choose to use one label to represent that prefix. Similarly if the router is forwarding several (otherwise unrelated) CIDR prefixes in the same way it may choose to use the same label for this set of prefixes. For instance all CIDR prefixes which share the same BGP Next Hop could be assigned the same label. Taking this to the limit, an egress router may choose to advertise all of its prefixes with the same label.

By introducing the concept of an egress identifier, the distribution of labels associated with groups of CIDR prefixes can be simplified. For instance, an egress identifier might specify the BGP Next Hop, with all prefixes routed to that next hop receiving the label associated with that egress identifier. Another natural place to aggregate would be the MPLS egress router. This would work particularly well in conjunction with a link-state routing protocol, where the association between egress router and CIDR prefix is already distributed throughout an area.

For IP multicast, the natural binding of a label would be to a multicast tree, or rather to the branch of a tree which extends from a particular port. Thus for a shared tree, the label corresponds to the multicast group, (\*,G). For (S,G) state, the label would correspond to the source address and the multicast group.

A label can also have a granularity finer than a host route. That is, it could be associated with some combination of source and destination address or other information within the packet. This might for example be done on an administrative basis to aid in effecting policy. A label could also correspond to all packets which match a particular Integrated Services filter specification.

Labels can also represent explicit routes. This use is semantically equivalent to using an IP tunnel with a complete explicit route. This is discussed in more detail in [section 4.10](#).

### [2.2.3](#) Label Assignment

Essential to label switching is the notion of binding between a label and Network Layer routing (routes). A control component is responsible for creating label bindings, and then distributing the label binding information among label switches. Label assignment involves allocating a label, and then binding a label to a route.

Label assignment can be driven by control traffic or by data traffic. This is discussed in more detail in [section 3.4](#).

Control traffic driven label assignment has several advantages, as compared to data traffic driven label Assignment. For one thing, it minimizes the amount of additional control traffic needed to distribute label binding information, as label binding information is distributed only in response to control traffic, independent of data traffic. It also makes the overall scheme independent of and insensitive to the data traffic profile/pattern. Control traffic driven creation of label binding improves forwarding latency, as labels are assigned before data traffic arrives, rather than being assigned as data traffic arrives. It also simplifies the overall system behavior, as the control plane is controlled solely by control traffic, rather than by a mix of control and data traffic.

There are however situations where data traffic driven label assignment is necessary. A particular case may occur with ATM without VP or VC merge. In this case in order to set up a full mesh of VCs would require  $n$ -squared VCs. However, in very large networks this may be infeasible. Instead VCs may be setup where required for forwarding data traffic. In this case it is generally not possible to know a priori how many such streams may occur.

Label withdrawal is required with both control-driven and data-driven label assignment. Label withdrawal is primarily a matter of garbage collection, that is collecting up unused labels so that they may be reassigned. Generally speaking, a label should be withdrawn when the conditions that allowed it to be assigned are no longer true. For example, if a label is imbued with extra semantics such as loop-freeness, then the label must be withdrawn when those extra semantics cease to hold.

In certain cases, notably multicast, it may be necessary to share a label space between multiple entities. If these sharing arrangements are altered by the coming and going of neighbors, then labels which

are no longer controlled by an entity must be withdrawn and a new label assigned.

#### [2.2.4](#) Label Stack and Forwarding Operations



The basic forwarding operation consists of looking up the incoming label to determine the outgoing label, encapsulation, port, and any additional information which may pertain to the stream such as a particular queue or other QoS related treatment. We refer to this operation as a label swap.

When a packet first enters an MPLS domain, the packet is forwarded by normal layer 3 forwarding operations with the exception that the outgoing encapsulation will now include a label. We refer to this operation as a label push. When a packet leaves an MPLS domain, the label is removed. We refer to this as a label pop.

In some situations, carrying a stack of labels is useful. For instance both IGP and BGP label could be used to allow routers in the interior of an AS to be free of BGP information. In this scenario, the "IGP" label is used to steer the packet through the AS and the "BGP" label is used to switch between ASes.

With a label stack, the set of label operations remains the same, except that at some points one might push or pop multiple labels, or pop & swap, or swap & push.

### [2.3](#) Encapsulation

Label-based forwarding makes use of various pieces of information, including a label or stack of labels, and possibly additional information such as a TTL field. In some cases this information may be encoded using an MPLS header, in other cases this information may be encoded in L2 headers. Note that there may be multiple types of MPLS headers. For example, the header used over one media type may be different than is used over a different media type. Similarly, in some cases the information that MPLS makes use of may be encoded in an ATM header. We will use the term "MPLS encapsulation" to refer to whatever form is used to encapsulate the label information and other information used for label based forwarding. The term "MPLS header" will be used where this information is carried in some sort of MPLS-specific header (i.e., when the MPLS information cannot all be carried in a L2 header). Whether there is one or multiple forms of possible MPLS headers is also outside of the scope of this document.

The exact contents of the MPLS encapsulation is outside of the scope of this document. Some fields, such as the label, are obviously needed. Some others might or might not be standardized, based on

further study. An encapsulation scheme may make use of the following fields:

- label
- TTL
- class of service
- stack indicator
- next header type indicator
- checksum

It is desirable to have a very short encapsulation header. For example, a four byte encapsulation header adds to the convenience of building a hardware implementation that forwards based on the encapsulation header. But at the same time it is tricky assigning such a limited number of bits to carry the above listed information in an MPLS header. Hence careful consideration must be given to the information chosen for an MPLS header.

A TTL value in the MPLS header may be useful in the same manner as it is in IP. Specifically, TTL may be used to terminate packets caught in a routing loop, and for other related uses such as traceroute. The TTL mechanism is a simple and proven method of handling such events. Another use of TTL is to expire packets in a network by limiting their "time to live" and eliminating stale packets that may cause problems for some of the higher layer protocols. When used over link layers which do not provide a TTL field, alternate mechanisms will be needed to replace the uses of the TTL field.

A provision for a class of service (COS) field in the MPLS header allows multiple service classes within the same label. However, when more sophisticated QoS is associated with a label, the COS may not have any significance. Alternatively, the COS (like QoS) can be left out of the header, and instead propagated with the label assignment, but this entails that a separate label be assigned to each required class of service. Nevertheless, the COS mechanism provides a simple method of segregating flows within a label.

As previously mentioned, the encapsulation header can be used to derive benefits of tunneling (or stacking).

The MPLS header must provide a way to indicate that multiple MPLS headers are stacked (i.e., the "stack indicator"). For this purpose a single bit in the MPLS header will suffice. In addition, there are also some benefits to indicating the type of the protocol header following the MPLS header (i.e., the "next header type indicator"). One option would be to combine the stack indicator and next header type indicator into a single value (i.e., the next header type

indicator could be allowed to take the value "MPLS header"). Another

option is to have the next header type indicator be implicit in the label value (such that this information would be propagated along with the label).

There is no compelling reason to support a checksum field in the MPLS header. A CRC mechanism at the L2 layer should be sufficient to ensure the integrity of the MPLS header.

### 3. Observations, Issues and Assumptions

#### 3.1 Layer 2 versus Layer 3 Forwarding

MPLS uses L2 forwarding as a way to provide simple and fast packet forwarding capability. One primary reason for the simplicity of L2 layer forwarding comes from its short, fixed length labels. A node forwarding at L3 must parse a (relatively) large header, and perform a longest-prefix match to determine a forwarding path. However, when a node performs L2 label swapping, and labels are assigned properly, it can do a direct index lookup into its forwarding (or in this case, label-swapping) table with the short header. It is arguably simpler to build label swapping hardware than it is to build L3 forwarding hardware because the label swapping function is less complex.

The relative performance of L2 and L3 forwarding may differ considerably between nodes. Some nodes may illustrate an order of magnitude difference. Other nodes (for example, nodes with more extensive L3 forwarding hardware) may have identical performance at L2 and L3. However, some nodes may not be capable of doing a L3 forwarding at all (e.g. ATM), or have such limited capacity as to be unusable at L3. In this situation, traffic must be blackholed if no switched path exists.

On nodes in which L3 forwarding is slower than L2 forwarding, pushing traffic to L3 when no L2 path is available may cause congestion. In some cases this could cause data loss (since L3 may be unable to keep up with the increased traffic). However, if data is discarded, then in general this will cause TCP to backoff, which would allow control traffic, traceroute and other network management tools to continue to work.

The MPLS protocol MUST not make assumptions about the forwarding capabilities of an MPLS node. Thus, MPLS must propose solutions that can leverage the benefits of a node that is capable of L3 forwarding, but must not mandate the node be capable of such.

#### Why We Will Still Need L3 Forwarding:

MPLS will not, and is not intended to, replace L3 forwarding. There is absolutely a need for some systems to continue to forward IP packets using normal Layer 3 IP forwarding. L3 forwarding will be needed for a variety of reasons, including:

- For scaling; to forward on a finer granularity than the labels can provide
- For security; to allow packet filtering at firewalls.
- For forwarding at the initial router (when hosts don't do MPLS)

Consider a campus network which is serving a small company. Suppose that this company makes use of the Internet, for example as a method of communicating with customers. A customer on the other side of the world has an IP packet to be forwarded to a particular system within the company. It is not reasonable to expect that the customer will have a label to use to forward the packet to that specific system. Rather, the label used for the "first hop" forwarding might be sufficient to get the packet considerably closer to the destination. However, the granularity of the labels cannot be to every host worldwide. Similarly, routing used within one routing domain cannot know about every host worldwide. This implies that in many cases the labels assigned to a particular packet will be sufficient to get the packet close to the destination, but that at some points along the path of the packet the IP header will need to be examined to determine a finer granularity for forwarding that packet. This is particularly likely to occur at domain boundaries.

A similar point occurs at the last router prior to the destination host. In general, the number of hosts attached to a network is likely to be great enough that it is not feasible to assign a separate label to every host. Rather, as least for routing within the destination

routing domain (or the destination area if there is a hierarchical routing protocol in use) a label may be assigned which is sufficient to get the packet to the last hop router. However, the last hop router will need to examine the IP header (and particularly the destination IP address) in order to forward the packet to the correct destination host.

Packet filtering at firewalls is an important part of the operation of the Internet. While the current state of Internet security may be considerably less advanced than may be desired, nonetheless some security (as is provided by firewalls) is much better than no security. We expect that packet filtering will continue to be important for the foreseeable future. Packet filtering requires examination of the contents of the packet, including the IP header. This implies that at firewalls the packet cannot be forwarded simply

by considering the label associated with the packet. Note that this is also likely to occur at domain boundaries.

Finally, it is very likely that many hosts will not implement MPLS. Rather, the host will simply forward an IP packet to its first hop router. This first hop router will need to examine the IP header prior to forwarding the packet (with or without a label).

### [3.2](#) Scaling Issues

MPLS scalability is provided by two of the principles of routing. The first is that forwarding follows an inverted tree rooted at a destination. The second is that the number of destinations is reduced by routing aggregation.

The very nature of IP forwarding is a merged multipoint-to-point tree. Thus, since MPLS mirrors the IP network layer, an MPLS node that is capable of merging is capable of creating  $O(n)$  switched paths which provide network reachability to all "n" destinations. The meaning of "n" depends on the granularity of the switched paths. One obvious choice of "n" is the number of CIDR prefixes existing in the forwarding table (this scales the same as today's routing). However, the value of "n" may be reduced considerably by choosing switched paths of further aggregation. For example, by creating switched paths to each possible egress node, "n" may represent the number of egress nodes in a network. This choice creates "n" switched paths, such that

each path is shared by all CIDR prefixes that are routed through the same egress node. This selection greatly improves scalability, since it minimizes "n", but at the same time maintains the same switching performance of CIDR aggregation. (See [section 2.2.2](#) for a description of all of the levels of granularity provided by MPLS).

The MPLS technology must scale at least as well as existing technology. For example, if the MPLS technology were to support ONLY host-to-host switched path connectivity, then the number of switched-paths would be much higher than the number of routing table entries.

There are several ways in which merging can be done in order to allow  $O(n)$  switches paths to connect  $n$  nodes. The merging approach used has an impact on the amount of state information, buffering, delay characteristics, and the means of control required to coordinate the trees. These issues are discussed in more detail in [section 4.2](#).

There are some cases in which  $O(n^2)$  switched paths may be used (for example, by setting up a full mesh of point to point streams). As label space and the amount of state information that can be supported may be limited, it will not be possible to support  $O(n^2)$

squared) switched paths in very large networks. However, in some cases the use of  $n^2$  paths may even be a advantage (for example, to allow load-splitting of individual streams).

MPLS must be designed to scale for  $O(n)$ .  $O(n)$  scaling allows MPLS domains to scale to a very large scale. In addition, if best effort service can be supported with  $O(n)$  scaling, this conserves resources (such as label space and state information) which can be used for supporting advanced services such as QoS. However, since some switches may not support merging, and some small networks may not require the scaling benefits of  $O(n)$ , provisions must also be provided for a non-merging,  $O(n^2)$  solution.

Note: A precise and complete description of scaling would consider that there are multiple dimensions of scaling, and multiple resources whose usage may be considered. Possible dimensions of scaling include: (i) the total number of streams which exist in an MPLS domain (with associated labels assigned to them); (ii) the total number of "label swapping pairs" which may be stored in the nodes of

the network (ie, entries of the form "for incoming label 'x', use outgoing label 'y'"); (iii) the number of labels which need to be assigned for use over a particular link; (iv) The amount of state information which needs to be maintained by any one node. We do not intend to perform a complete analysis of all possible scaling issues, and understand that our use of the terms " $O(n)$ " and " $O(n\text{-squared})$ " is approximate only.

### 3.3 Types of Streams

Switched paths in the MPLS network can be of different types:

- point-to-point
- multipoint-to-point
- point-to-multipoint
- multipoint-to-multipoint

Two of the factors that determine which type of switched path is used are (i) The capability of the switches employed in a network; (ii) The purpose of the creation of a switched path; that is, the types of flows to be carried in the switched path. These two factor also determine the scalability of a network in terms of the number of switched paths in use for transporting data through a network.

The point-to-point switched path can be used to connect all ingress nodes to all the egress nodes to carry unicast traffic. In this case, since an ingress node has point-to-point connections to all the egress nodes, the number of connections in use for transporting traffic is of  $O(n\text{-squared})$ , where  $n$  is the number of edges MPLS

devices. For small networks the full mesh connection approach may suffice and not pose any scalability problems. However, in large enterprise backbone or ISP networks, this will not scale well.

Point-to-point switched paths may be used on a host-to-host or application to application basis (e.g., a switched path per RSVP flow). The dedicated point-to-point switched path transports the unicast data from the ingress to the egress node of the MPLS network. This approach may be used for providing QoS services or for best-effort traffic.

A multipoint-to-point switched path connects all ingress nodes to an

single egress node. At a given intermediate node in the multipoint-to-point switched path, L2 data units from several upstream links are "merged" into a single label on a downstream link. Since each egress node is reachable via a single multipoint-to-point switched path, the number of switched paths required to transport best-effort traffic through a MPLS network is  $O(n)$ , where  $n$  is the number of egress nodes.

The point-to-multipoint switched path is used for distributing multicast traffic. This switched path tree mirrors the multicast distribution tree as determined by the multicast routing protocols. Typically a switch capable of point-to-multipoint connection replicates an L2 data unit from the incoming (parent) interface to all the outgoing (child) interfaces. Standard ATM switches support such functionality in the form of point-to-multipoint VCs or VPs.

A multipoint-to-multipoint switched path may be used to combine multicast traffic from multiple sources into a single multicast distribution tree. The advantage of this is that the multipoint-to-multipoint switched path is shared by multiple sources. Conceptually, a form of multipoint-to-multipoint can be thought of as follows: Suppose that you have a point to multipoint VC from each node to all other nodes. Suppose that any point where two or more VCs happen to merge, you merge them into a single VC or VP. This would require either coordination of VCI spaces (so that each source has a unique VCI within a VP) or VC merge capabilities. The applicability of similar concepts to MPLS is FFS.

#### [3.4](#) Data Driven versus Control Traffic Driven Label Assignment

A fundamental concept in MPLS is the association of labels and network layer routing. Each LSR must assign labels, and distribute them to its forwarding peers, for traffic which it intends to forward by label swapping. In the various contributions that have been made so far to the MPLS WG we identify three broad strategies for label assignment; (i) those driven by topology based control traffic

[TAG][ARIS][IP navigator]; (ii) Those driven by request based control traffic [RSVP]; and (iii) those driven by data traffic [CSR][Ipsilon].

We also note that in actual practice combinations of these methods



may be employed. One example is that topology based methods for best effort traffic plus request based methods for support of RSVP.

#### 3.4.1 Topology Driven Label Assignment

In this scheme labels are assigned in response to normal processing of routing protocol control traffic. Examples of such control protocols are OSPF and BGP. As an LSR processes OSPF or BGP updates it can, as it makes or changes entries in its forwarding tables, assign labels to those entries.

Among the properties of this scheme are:

- The computational load of assignment and distribution and the bandwidth consumed by label distribution are bounded by the size of the network.
- Labels are in the general case preassigned. If a route exists then a label has been assigned to it (and distributed). Traffic may be label swapped immediately it arrives, there is no label setup latency at forwarding time.
- Requires LSRs to be able to process control traffic load only.
- Labels assigned in response to the operation of routing protocols can have a granularity equivalent to that of the routes advertised by the protocol. Labels can, by this means, cover (highly) aggregated routes.

#### 3.4.2 Request Driven Label Assignment

In this scheme labels are assigned in response to normal processing of request based control traffic. Examples of such control protocols are RSVP. As an LSR processes RSVP messages it can, as it makes or changes entries in its forwarding tables, assign labels to those entries.

Among the properties of this scheme are:

- The computational load of assignment and distribution and the bandwidth consumed by label distribution are bounded by the amount of control traffic in the system.

- Labels are in the general case preassigned. If a route exists then a label has been assigned to it (and distributed). Traffic may be label swapped immediately it arrives, there is no label setup latency at forwarding time.
- Requires LSRs to be able to process control traffic load only.
- Depending upon the number of flows supported, this approach may require a larger number of labels to be assigned compared with topology driven assignment.
- This approach requires applications to make use of request paradigm in order to get a label assigned to their flow.

### 3.4.3 Traffic Driven Label Assignment

In this scheme the arrival of data at an LSR "triggers" label assignment and distribution. Traffic driven approach has the following characteristics.

- Label assignment and distribution costs are a function of traffic patterns. In an LSR with limited label space that is using a traffic driven approach to amortize its labels over a larger number of flows the overhead due to label assignment and distribution grows as a function of the number of flows and as a function of their "persistence". Short lived but recurring flows may impose a heavy control burden.
- There is a latency associated with the appearance of a "flow" and the assignment of a label to it. The documented approaches to this problem suggest L3 forwarding during this setup phase, this has the potential for packet reordering (note that packet reordering may occur with any scheme when the network topology changes, but traffic driven label assignment introduces another cause for reordering).
- Flow driven label assignment requires high performance packet classification capabilities.
- Traffic driven label assignment may be useful to reduce label consumption (assuming that flows are not close to full mesh).
- If you want flows to hosts, due to limits on label space, then traffic based label consumption is probably necessary due to the large number of hosts which may occur in a network.
- If you want to assign specific network resources to specific labels, to be used for support of application flows, then

INTERNET DRAFT

A Framework for MPLS

July 30, 1997

again the fine grain associated with labels may require data based label assignment.

### [3.5](#) The Need for Dealing with Looping

Routing protocols which are used in conjunction with MPLS will in many cases be based on distributed computation. As such, during routing transients, these protocols may compute forwarding paths which contain loops. For this reason MPLS will be designed with mechanisms to either prevent the formation of loops and /or contain the amount of resources that can be consumed due to the presence of loops.

Note that there are a number of different alternative mechanisms which have been proposed (see [section 4.3](#)). Some of these prevent the formation of layer 2 forwarding loops, others allow loops to form but minimize their impact in one way or another (e.g., by discarding packets which loop, or by detecting and closing the loop after a period of time). Generally speaking, there are tradeoffs to be made between the amount of looping which might occur, and other considerations such as the time to convergence after a change in the paths computed by the routing algorithm.

We are not proposing any changes to normal layer 3 operation, and specifically are not trying to eliminate the possibility of looping at layer 3. Transient loops will continue to be possible in IP networks. Note that IP has a means to limit the damage done by looping packets, based on decrementing the IP TTL field as the packet is forwarded, and discarding packets whose TTL has expired. Dynamic routing protocols used with IP are also designed to minimize the amount of time during which loops exist.

The question that MPLS has to deal with is what to do at L2. In some cases L2 may make use of the same method that is used as L3. However, other options are available at L2, and in some cases (specifically when operating over ATM or Frame Relay hardware) the method of decrementing a TTL field (or any similar field) is not available.

There are basically two problems caused by packet looping: The most obvious problem is that packets are not delivered to the correct destination. The other result of looping is congestion. Even with TTL decrementing and packet discard, there may still be a significant

amount of time that packets travel through a loop. This can adversely affect other packets which are not looping: Congestion due to the looping packets can cause non-looping packets to be delayed and/or discarded.

Looping is particularly serious in (at least) three cases: One is

when forwarding over ATM. Since ATM does not have a TTL field to decrement, there is no way to discard ATM cells which are looping over ATM subnetworks. Standard ATM PNNI routing and signaling solves this problem by making use of call setup procedures which ensure that ATM VCs will never be setup in a loop [PNNI]. However, when MPLS is used over ATM subnets, the native ATM routing and signaling procedures may not be used for the full L2 path. This leads to the possibility that MPLS over ATM might in principle allow packets to loop indefinitely, or until L3 routing stabilizes. Methods are needed to prevent this problem.

Another case in which looping can be particularly unpleasant is for multicast traffic. With multicast, it is possible that the packet may be delivered successfully to some destinations even though copies intended for other destinations are looping. This leads to the possibility that huge numbers of identical packets could be delivered to some destinations. Also, since multicast implies that packets are duplicated at some points in their path, the congestion resulting from looping packets may be particularly severe.

Another unpleasant complication of looping occurs if the congestion caused by the loop interferes with the routing protocol. It is possible for the congestion caused by looping to cause routing protocol control packets to be discarded, with the result that the routing protocol becomes unstable. For example this could lengthen the duration of the loop.

In normal operation of IP networks the impact of congestion is limited by the fact that TCP backs off (i.e., transmits substantially less traffic) in response to lost packets. Where the congestion is caused by looping, the combination of TTL and the resulting discard of looping packets, plus the reduction in offered traffic, can limit the resulting impact on the network. TCP backoff however does not solve the problem if the looping packets are not discarded (for example, if the loop is over an ATM subnetwork where TTL is not

used).

The severity of the problem caused by looping may depend upon implementation details. Suppose, for instance, that ATM switching hardware is being used to provide MPLS switching functions. If the ATM hardware has per-VC queuing, and if it is capable of providing fair access to the buffer pool for incoming cells based on the incoming VC (so that no one incoming VC is allowed to grab a disproportionate number of buffers), this looping might not have a significant 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 node's total performance.

Given that MPLS is a relatively new approach, it is possible that looping may have consequences which are not fully understood (such as looping of LDP control information in cases where stream merge is not used).

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.

Methods for dealing with loops are discussed in [section 4.3](#).

### [3.6](#) Operations and Management

Operations and management of networks is critically important. This implies that MPLS must support operations, administration, and maintenance facilities at least as extensive as those supported in current IP networks.

In most ways this is a relatively simple requirement to meet. Given that all MPLS nodes run normal IP routing protocols, it is straightforward to expect them to participate in normal IP network management protocols.

There is one issue which has been identified and which needs to be addressed by the MPLS effort: There is an issue with regard to operation of Traceroute over MPLS networks. Note that other O&M

issues may be identified in the future.

Traceroute is a very commonly used network management tool. Traceroute is based on use of the TTL field: A station trying to determine the route from itself to a specified address transmits multiple IP packets, with the TTL field set to 1 in the first packet, 2 in the second packet, etc.. This causes each router along the path to send back an ICMP error report for TTL exceeded. This in turn allows the station to determine the set of routers along the route. For example, this can be used to determine where a problem exists (if no router responds past some point, the last router which responds can become the starting point for a search to determine the cause of the problem).

When MPLS is operating over ATM or Frame Relay networks there is no TTL field to decrement (and ATM and Frame Relay forwarding hardware does not decrement TTL). This implies that it is not straightforward to have Traceroute operate in this environment.

There is the question of whether we \*want\* all routers along a path

to be visible via traceroute. For example, an ISP probably doesn't want to expose the interior of their network to a customer. However, the issue of whether a network's policy will allow the interior of the network to be visible should be independent of whether it is possible for some users to see the interior of the network. Thus while there clearly should be the possibility of using policy mechanisms to block traceroute from being used to see the interior of the network, this does not imply that it is okay to develop protocol mechanisms which break traceroute from working.

There is also the question of whether the interior of a MPLS network is analogous to a normal IP network, or whether it is closer to the interior of a layer 2 network (for example, an ATM subnet). Clearly IP traceroute cannot be used to expose the interior of an ATM subnet. When a packet is crossing an ATM subnetwork (for example, between an ingress and an egress router which are attached to the ATM subnet) traceroute can be used to determine the router to router path, but not the path through the ATM switches which comprise the ATM subnet. Note here that MPLS forms a sort of "in between" special case: Routing is based on normal IP routing protocols, the equivalent of call setup (label binding/exchange) is based on MPLS-specific

protocols, but forwarding is based on normal L2 ATM forwarding. MPLS therefore supersedes the normal ATM-based methods that would be used to eliminate loops and/or trace paths through the ATM subnet.

It is generally agreed that Traceroute is a relatively "ugly" tool, and that a better tool for tracing the route of a packet would be preferable. However, no better tool has yet been designed or even proposed. Also, however ugly Traceroute may be, it is nonetheless very useful, widely deployed, and widely used. In general, it is highly preferable to define, implement, and deploy a new tool, and to determine through experience that the new tool is sufficient, before breaking a tool which is as widely used as traceroute.

Methods that may be used to either allow traceroute to be used in an MPLS network, or to replace traceroute, are discussed in [section 4.14](#).

#### [4. Technical Approaches](#)

We believe that [section 4](#) is probably less complete than other sections. Additional subsections are likely to be needed as a result of additional discussions in the MPLS working group.

#### [4.1 Label Distribution](#)

A fundamental requirement in MPLS is that an LSR forwarding label switched traffic to another LSR apply a label to that traffic which is meaningful to the other (receiving the traffic) LSR. LSR's could learn about each other's labels in a variety of ways. We call the general topic "label distribution".

##### [4.1.1 Explicit Label Distribution](#)

Explicit label distribution anticipates the specification by MPLS of a standard protocol for label distribution. Two of the possible approaches [TDP] [ARIS] that are oriented toward topology driven label distribution. One other approach [FANP], in contrast, makes use

of traffic driven label distribution.

We expect that the label distribution protocol (LDP) which emerges from the MPLS WG is likely to inherit elements from one or more of the possible approaches.

Consider LSR A forwarding traffic to LSR B. We call A the upstream (wrt to dataflow) LSR and B the downstream LSR. A must apply a label to the traffic that B "understands". Label distribution must ensure that the "meaning" of the label will be communicated between A and B. An important question is whether A or B (or some other entity) allocates the label.

In this discussion we are talking about the allocation and distribution of labels between two peer LSRs that are on a single segment of what may be a longer path. A related but in fact entirely separate issue is the question of where control of the whole path resides. In essence there are two models; by analogy to upstream and downstream for a single segment we can talk about ingress and egress for an LSP (or to and from a label swapping "domain"). In one model a path is setup from ingress to egress in the other from egress to ingress.

#### 4.1.1.1 Downstream Label Allocation

"Downstream Label Allocation" refers to a method where the label allocation is done by the downstream LSR, i.e. the LSR that uses the label as an index into its switching tables.

This is, arguably, the most natural label allocation/distribution mode for unicast traffic. As an LSR build its routing tables (we consider here control driven allocation of tags) it is free, within some limits we will discuss, to allocate labels to in any manner that may be convenient to the particular implementation. Since the labels

that it allocates will be those upon which it subsequently makes forwarding decisions we assume implementations will perform the allocation in an optimal manner. Having allocated labels the default behavior is to distribute the labels (and bindings) to all peers.

In some cases (particularly with ATM) there may be a limited number of labels which may be used across an interface, and/or a limited



number of label assignments which may be supported by a single device. Operation in this case may make use of "on demand" label assignment. With this approach, an LSR may for example request a label for a route from a particular peer only when its routing calculations indicate that peer to be the new next hop for the route.

#### 4.1.1.2 Upstream Label Allocation

"Upstream Label Allocation" refers to a method where the label allocation is done by the upstream LSR. In this case the LSR choosing the label (the upstream LSR) and the LSR which needs to interpret packets using the label (the downstream LSR) are not the same node. We note here that in the upstream LSR the label at issue is not used as an index into the switching tables but rather is found as the result of a lookup on those tables.

The motivation for upstream label allocation comes from the recognition that it might be possible to optimize multicast machinery in an LSR if it were possible to use the same label on all output ports for which a particular multicast packet/cell were destined. Upstream assignment makes this possible.

#### 4.1.1.3 Other Label Allocation Methods

Another option would be to make use of label values which are unique within the MPLS domain (implying that a domain-wide allocation would be needed). In this case, any stream to a particular MPLS egress node could make use of the label of that node (implying that label values do not need to be swapped at intermediate nodes).

With this method of label allocation, there is a choice to be made regarding the scope over which a label is unique. One approach is to configure each node in an MPLS domain with a label which is unique in that domain. Another approach is to use a truly global identifier (for example the IEEE 48 bit identifier), where each MPLS-capable node would be stamped at birth with a truly globally unique identifier. The point of this global approach is to simplify configuration in each MPLS domain by eliminating the need to configure label IDs.

### [4.1.2](#) Piggybacking on Other Control Messages

While we have discussed use of an explicit MPLS LDP we note that there are several existing protocols that can be easily modified to distribute both routing/control and label information. This could be done with any of OSPF, BGP, RSVP and/or PIM. A particular architectural elegance of these schemes is that label distribution uses the same mechanisms as are used in distribution of the underlying routing or control information.

When explicit label distribution is used, the routing computation and label distribution are decoupled. This implies a possibility that at some point you may either have a route to a specific destination without an associated label, and/or a label for a specific destination which makes use of a path which you are no longer using. Piggybacking label distribution on the operation of the routing protocol is one way to eliminate this decoupling.

Piggybacking label distribution on the routing protocol introduces an issue regarding how to negotiate acceptable label values and what to do if an invalid label is received. This is discussed in [section 4.1.3](#).

### [4.1.3](#) Acceptable Label Values

There are some constraints on which label values may be used in either allocation mode. Clearly the label values must lie within the allowable range described in the encapsulation standards that the MPLS WG will produce. The label value used must also, however, lie within a range that the peer LSR is capable of supporting. We imagine that certain machines, for example ATM switches operating as LSRs may, due to operational or implementation restrictions, support a label space more limited than that bounded by the valid range found in the encapsulation standard. This implies that an advertisement or negotiation mechanism for useable label range may be a part of the MPLS LDP. When operating over ATM using ATM forwarding hardware, due to the need for compatibility with the existing use of the ATM VPI/VCI space, it is quite likely that an explicit mechanism will be needed for label range negotiation.

In addition we note that LDP may be one of a number of mechanism used to distribute labels between any given pair of LSRs. Clearly where such multiple mechanisms exist care must be taken to coordinate the allocation of label values. A single label value must have a unique meaning to the LSR that distributes it.

There is an issue regarding how to allow negotiation of acceptable label values if label distribution is piggybacked with the routing

INTERNET DRAFT

A Framework for MPLS

July 30, 1997

protocol. In this case it may be necessary either to require equipment to accept any possible label value, or to configure devices to know which range of label values may be selected. It is not clear in this case what to do if an invalid label value is received as there may be no means of sending a NAK.

A similar issue occurs with multicast traffic over broadcast media, where there may be multiple nodes which receive the same transmission (using a single label value). Here again it may be "non-trivial" how to allow n-party negotiation of acceptable label values.

#### [4.1.4](#) LDP Reliability

The need for reliable label distribution depends upon the relative performance of L2 and L3 forwarding, as well as the relationship between label distribution and the routing protocol operation.

If label distribution is tied to the operation of the routing protocol, then a reasonable protocol design would ensure that labels are distributed successfully as long as the associated route and/or reachability advertisement is distributed successfully. This implies that the reliability of label distribution will be the same as the reliability of route distribution.

If there is a very large difference between L2 and L3 forwarding performance, then the cost of failing to deliver a label is significant. In this case it is important to ensure that labels are distributed reliably. Given that LDP needs to operate in a wide variety of environments with a wide variety of equipment, this implies that it is important for any LDP developed by the MPLS WG to ensure reliable delivery of label information.

Reliable delivery of LDP packets 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).

#### [4.1.5](#) Label Purge Mechanisms

Another issue to be considered is the "lifetime" of label data once it arrives at an LSR, and the method of purging label data. There are several methods that could be used either separately, or (more

likely) in combination.

One approach is for label information to be timed out. With this approach a lifetime is distributed along with the label value. The label value may be refreshed prior to timing out. If the label is not

refreshed prior to timing out it is discarded. In this case each lifetime and timer may apply to a single label, or to a group of labels (e.g., all labels selected by the same node).

Similarly, two peer nodes may make use of an MPLS peer keep-alive mechanism. This implies exchange of MPLS control packets between neighbors on a periodic basis. This in general is likely to use a smaller timeout value than label value timers (analogous to the fact that the OSPF HELLO interval is much shorter than the OSPF LSA lifetime). If the peer session between two MPLS nodes fails (due to expiration of the associated timer prior to reception of the refresh) then associated label information is discarded.

If label information is piggybacked on the routing protocol then the timeout mechanisms would also be taken from the associated routing protocol (note that routing protocols in general have mechanisms to invalidate stale routing information).

An alternative method for invalidating labels is to make use of an explicit label removal message.

#### [4.2 Stream Merging](#)

In order to scale  $O(n)$  (rather than  $O(n^2)$ ), MPLS makes use of the concept of stream merge. This makes use of multipoint to point streams in order to allow multiple streams to be merged into one stream.

##### Types of Stream Merge:

There are several types of stream merge that can be used, depending upon the underlying media.

When MPLS is used over frame based media merging is straightforward. All that is required for stream merge to take place is for a node to allow multiple upstream labels to be forwarded the same way and

mapped into a single downstream label. This is referred to as frame merge.

Operation over ATM media is less straightforward. In ATM, the data packets are encapsulated into an ATM Adaptation Layer, say AAL5, and the AAL5 PDU is segmented into ATM cells with a VPI/VCI value and the cells are transmitted in sequence. It is contingent on ATM switches to keep the cells of a PDU (or with the same VPI/VCI value) contiguous and in sequence. This is because the device that reassembles the cells to re-form the transmitted PDU expects the cells to be contiguous and in sequence, as there isn't sufficient information in the ATM cell header (unlike IP fragmentation) to

reassemble the PDU with any cell order. Hence, if cells from several upstream link are transmitted onto the same downstream VPI/VCI, then cells from one PDU can get interleaved with cells from another PDU on the outgoing VPI/VCI, and result in corruption of the original PDUs by mis-sequencing the cells of each PDU.

The most straightforward (but erroneous) method of merging in an ATM environment would be to take the cells from two incoming VCs and merge them into a single outgoing VCI. If this was done without any buffering of cells then cells from two or more packets could end up being interleaved into a single AAL5 frame. Therefore the problem when operating over ATM is how to avoid interleaving of cells from multiple sources.

There are two ways to solve this interleaving problem, which are referred to as VC merge and VP merge.

VC merge allows multiple VCs to be merged into a single outgoing VC. In order for this to work the node performing the merge needs to keep the cells from one AAL5 frame (e.g., corresponding to an IP packet) separate from the cells of other AAL5 frames. This may be done by performing the SAR function in order to reassemble each IP packet before forwarding that packet. In this case VC merge is essentially equivalent to frame merge. An alternative is to buffer the cells of one AAL5 frame together, without actually reassembling them. When the end of frame indicator is reached that frame can be forwarded. Note however that both forms of VC merge requires that the entire AAL5 frame be received before any cells corresponding to that frame be forwarded. VC merge therefore requires capabilities which are

generally not available in most existing ATM forwarding hardware.

The alternative for use over ATM media is VP merge. Here multiple VPs can be merged into a single VP. Separate VCIs within the merged VP are used to distinguish frames (e.g., IP packets) from different sources. In some cases, one VP may be used for the tree from each ingress node to a single egress node.

#### Interoperation of Merge Options:

If some nodes support stream merge, and some nodes do not, then it is necessary to ensure that the two types of nodes can interoperate within a single network. This affects the number of labels that a node needs to send to a neighbor. An upstream LSR which supports Stream Merge needs to be sent only one label per forwarding equivalence class (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.

If a particular upstream neighbor does not support stream merge, it is not known a priori how many labels it will need. The upstream neighbor may need to explicitly ask for labels for each FEC. The upstream neighbor may make multiple such requests (for one or more labels per request). 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 more labels 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 node will need to obtain the required two labels.

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 outgoing VPI/VCI for a particular FEC (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 it will require a single outgoing VPI/VCI per FEC for itself (assuming that it can be an ingress node), plus enough outgoing VPI/VCIs to map to incoming 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.

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). 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).

Note that there are multiple options for coordinating VCIs within a

VP. Description of the range of options is FFS.

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). VP merge nodes would therefore request one 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 they can be ingress nodes). However, non-merge nodes which can only do VC forwarding (and not VP forwarding) will need to know which VCIs are used within each VP in order to install the correct VCs in its forwarding table. A detailed description of how this could work is FFS.

### Coordination of the VCI space with VP Merge:

VP merge requires that the VCIs be coordinated to ensure uniqueness. There are a number of ways in which this may be accomplished:

1. Each node may be pre-configured with a unique VCI value (or values).
2. Some one node (most likely the root of the multipoint to point tree) may coordinate the VCI values used within the VP. A protocol mechanism will be needed to allow this to occur. How hard this is to do depends somewhat upon whether the root is otherwise involved in coordinating the multipoint to point tree. For example, allowing one node (such as the root) to coordinate the tree may be useful for purposes of coordinating load sharing (see [section 4.10](#)). Thus whether or not the issue of coordinating the VCI space is significant or trivial may depend upon other design choices which at first glance may have appeared to be independent protocol design choices.
3. Other unique information such as portions of a class B or class C address may be used to provide a unique VCI value.
4. Another alternative is to implement a simple hardware extension in the ATM switches to keep the VCI values unique by dynamically altering them to avoid collision.

VP merge makes less efficient use of the VPI/VCI space (relative to VC merge). When VP merge is used, the LSPs may not be able to transit public ATM networks that don't support SVP.

### Buffering Issues Related To Stream Merge:

There is an issue regarding the amount of buffering required for frame merge, VC merge, and VP merge. Frame merge and VC merge requires that intermediate points buffer incoming packets until the entire packet arrives. This is essentially the same as is required in traditional IP routers.



VP merge allows cells to be transmitted by intermediate nodes as soon as they arrive, reducing the buffering and latency at intermediate nodes. However, the use of VP merge implies that cells from multiple packets will arrive at the egress node interleaved on separate VCIs. This in turn implies that the egress node may have somewhat increased buffering requirements. To a large extent egress nodes for some destinations will be intermediate nodes for other destinations, implying that increase in buffers required for some purpose (egress traffic) will be offset by a reduction in buffers required for other purposes (transit traffic). Also, routers today typically deal with high-fanout channelized interfaces and with multi-VC ATM interfaces, implying that the requirement of buffering simultaneously arriving cells from multiple packets and sources is something that routers typically do today. This is not meant to imply that the required buffer size and performance is inexpensive, but rather is meant to observe that it is a solvable issue.

### [4.3](#) Loop Handling

Generally, methods for dealing with loops can be split into three categories: Loop Survival makes use of methods which minimize the impact of loops, for example by limiting the amount of network resources which can be consumed by a loop; Loop Detection allows loops to be set up, but later detects these loops and eliminates them; Loop Prevention provides methods for avoiding setting up L2 forwarding in a way which results in a L2 loop.

Note that we are concerned here only with loops that occur in L2 forwarding. Transient loops at L3 will continue to be part of the normal IP operation, and will be handled the way that IP has been handling loops for years (see [section 3.5](#)).

#### Loop Survival:

Loop Survival refers to methods that are used to allow the network to operate well even though short term transient loops may be formed by the routing protocol. The basic approach to loop survival is to limit the amount of network resources which are consumed by looping packets, and to minimize the effect on other (non-looping) traffic. Note that loop survival is the method used by conventional IP

forwarding, and is therefore based on long and relatively successful

experience in the Internet.

The most basic method for loop survival is based on the use to a TTL (Time To Live) field. The TTL field is decremented at each hop. If the TTL field reaches zero, then the packet is discarded. This method works well over those media which has a TTL field. This explicitly includes L3 IP forwarding. Also, assuming that the core MPLS specifications will include definition of a "shim" MPLS header for use over those media which do not have their own labels, in order to carry labels for use in forwarding of user data, it is likely that the shim header will also include a TTL field.

However, there is considerable interest in using MPLS over L2 protocols which provide their own labels, with the L2 label used for MPLS forwarding. Specific L2 protocols which offer a label for this purpose include ATM and Frame Relay. However, neither ATM nor Frame Relay have a TTL field. This implies that this method cannot be used when basic ATM or Frame Relay forwarding is being used.

Another basic method for loop survival is the use of dynamic routing protocols which converge rapidly to non-looping paths. In some instances it is possible that congestion caused by looping data could effect the convergence of the routing protocol (see [section 3.5](#)). MPLS should be designed to prevent this problem from occurring. Given that MPLS uses the same routing protocols as are used for IP, this method does not need to be discussed further in this framework document.

Another possible tool for loop survival is the use of fair queuing. This allows unrelated flows of user data to be placed in different queues. This helps to ensure that a node which is overloaded with looping user data can nonetheless forward unrelated non-looping data, thereby minimizing the effect that looping data has on other data. We cannot assume that fair queuing will always be available. In practice, many fair queuing implementations merge multiple streams into one queue (implying that the number of queues used is less than the number of user data flows which are present in the network). This implies that any data which happens to be in the same queue with looping data may be adversely effected.

Loop Detection:

Loop Detection refers to methods whereby a loop may be set up at L2, but the loop is subsequently detected. When the loop is detected, it may be broken at L2 by dropping the label relationship, implying that packets for a set of destinations must be forwarded at L3.

---

A possible method for loop detection is based on transmitting a "loop detection" control packet (LDCP) along the path towards a specified destination whenever the route to the destination changes. This LDCP is forwarded in the direction that the label specifies, with the labels swapped to the correct next hop value. However, normal L2 forwarding cannot be used because each hop needs to examine the packet to check for loops. The LDCP is forwarded towards that destination until one of the following happens: (i) The LDCP reaches the last MPLS node along the path (i.e. the next hop is either a router which is not participating in MPLS, or is the final destination host); (ii) The TTL of the LDCP expires (assuming that the control packet uses a TTL, which is optional but not absolutely necessary), or (iii) The LDCP returns to the node which originally transmitted it. If the latter occurs, then the packet has looped and the node which originally transmitted the LDCP stops using the associated label, and instead uses L3 forwarding for the associated destination addresses. One problem with this method is that once a loop is detected it is not known when the loop clears. One option would be to set a timer, and to transmit a new LDCP when the timer expires.

An alternate method counts the hops to each egress node, based on the routes currently available. Each node advertises its distance (in hop counts) to each destination. An egress node advertises the destinations that it can reach directly with an associated hop count of zero. For each destination, a node computes the hop count to that destination based on adding one to the hop count advertised by its actual next hop used for that destination. When the hop count for a particular destination changes, the hop counts needs to be readvertised.

In addition, the first of the loop prevention schemes discussed below may be modified to provide loop detection (the details are straightforward, but have not been written down in time to include in this rough draft).

#### Loop Prevention:

Loop prevention makes use of methods to ensure that loops are never set up at L2. This implies that the labels are not used until some method is used to ensure that following the label towards the destination, with associated label swaps at each switch, will not result in a loop. Until the L2 path (making use of assigned labels) is available, packets are forwarded at L3.

Loop prevention requires explicit signaling of some sort to be used

when setting up an L2 stream.

One method of loop prevention requires that labels be propagated starting at the egress switch. The egress switch signals to neighboring switches the label to use for a particular destination. That switch then signals an associated label to its neighbors, etc. The control packets which propagate the labels also include the path to the egress (as a list of routerIDs). Any looping control packet can therefore be detected and the path not set up to or past the looping point. <Operation when routing changes needs to be described here in more detail>.

Another option is to use explicit routing to set up label bindings from the egress switch to each ingress switch. This precludes the possibility of looping, since the entire path is computed by one node. This also allows non-looping paths to be set up provided that the egress switch has a view of the topology which is reasonably close to reality (if there are operational links which the egress switch doesn't know about, it will simply pick a path which doesn't use those links; if there are links which have failed but which the the egress switch thinks are operational, then there is some chance that the setup attempt will fail but in this case the attempt can be retried on a separate path). Note therefore that non-looping paths can be set up with this method in many cases where distributed routing plus hop by hop forwarding would not actually result in non-looping paths. This method is similar to the method used by standard ATM routing to ensure that SVCs are non-looping [PNNI].

Explicit routing is only applicable if the routing protocol gives the egress switch sufficient information to set up the explicit route, implying that the protocol must be either a link state protocol (such as OSPF) or a path vector protocol (such as BGP). Source routing therefore is not appropriate as a general approach for use in any network regardless of the routing protocol. This method also requires some overhead for the call setup before label-based forwarding can be used. If the network topology changes in a manner which breaks the existing path, then a new path will need to be explicitly routed from the egress switch. Due to this overhead this method is probably only appropriate if other significant advantages are also going to be obtained from having a single node (the egress switch) coordinate the paths to be used. Examples of other reasons to have one node

coordinate the paths to a single egress switch include: (i) Coordinating the VCI space where VP merge is used (see [section 4.2](#)); and (ii) Coordinating the routing of streams from multiple ingress switches to one egress switch so as to balance the load on multiple alternate paths through the network.

In principle the explicit routing could also be done in the alternate direction (from ingress to egress). However, this would make it more difficult to merge streams if stream merge is to be used. This would

also make it more difficult to coordinate (i) changes to the paths used, (ii) the VCI space assignments, and (iii) load sharing. This therefore makes explicit routing more difficult, and also reduces the other advantages that could be obtained from the approach.

If label distribution is piggybacked on the routing protocol (see [section 4.1.2](#)), then loop prevention is only possible if the routing protocol itself does loop prevention.

#### What To Do If A Loop Is Detected:

With all of these schemes, if a loop is known to exist then the L2 label-swapped path is not set up. This leads to the obvious question of what does an MPLS node do when it doesn't have a label for a particular destination, and a packet for that destination arrives to be forwarded? If possible, the packet is forwarded using normal L3 (IP) forwarding. There are two issues that this raises: (i) What about nodes which are not capable of L3 forwarding; (ii) Given the relative speeds of L2 and L3 forwarding, does this work?

Nodes which are not capable of L3 forwarding obviously can't forward a packet unless it arrives with a label, and the associated next hop label has been assigned. Such nodes, when they receive a packet for which the next hop label has not been assigned, must discard the packet. It is probably safe to assume that if a node cannot forward an L3 packet, then it is probably also incapable of forwarding an ICMP error report that it originates. This implies that the packet will need to be discarded in this case.

In many cases L2 forwarding will be significantly faster than L3 forwarding (allowing faster forwarding is a significant motivation behind the work on MPLS). This implies that if a node is forwarding a

large volume of traffic at L2, and a change in the routing protocol causes the associated labels to be lost (necessitating L3 forwarding), in some cases the node will not be capable of forwarding the same volume of traffic at L3. This will of course require that packets be discarded. However, in some cases only a relatively small volume of traffic will need to be forwarded at L3. Thus forwarding at L3 when L2 is not available is not necessarily always a problem. There may be some nodes which are capable of forwarding equally fast at L2 and L3 (for example, such nodes may contain IP forwarding hardware which is not available in all nodes). Finally, when packets are lost this will cause TCP to backoff, which will in turn reduce the load on the network and allow the network to stabilize even at reduced forwarding rates until such time as the label bindings can be reestablished.

Note that in most cases loops will be caused either by configuration

errors, or due to short term transient problems caused by the failure of a link. If only one link goes down, and if routing creates a normal "tree-shaped" set of paths to any one destination, then the failure of one link somewhere in the network will effect only one link's worth of data passing through any one node in the network. This implies that if a node is capable of forwarding one link's worth of data at L3, then in many or most cases it will have sufficient L3 bandwidth to handle looping data.

#### [4.4](#) Interoperation with NHRP

<note: Future versions of this document will contain a picture to clarify the discussion in this section. In addition there are alternate interaction scenarios which probably should be discussed briefly.>

When label switching is used over ATM, and there exists an LSR which is also operating as a Next Hop Client (NHC), the possibility of direct interaction arises. That is, could one switch cells between the two technologies without reassembly. To enable this several important issues must be addressed.

The encapsulation must be acceptable to both MPLS and NHRP. If only a single label is used, then the null encapsulation could be used. Other solutions could be developed to handle label stacks.

NHRP must understand and respect the granularity of a stream.

Currently NHRP resolves an IP address to an ATM address. The response may include a mask indicating a range of addresses. However, any VC to the ATM address is considered to be a viable means of packet delivery. Suppose that an NHC NHRPs for IP address A and gets back ATM address 1 and sets up a VC to address 1. Later the same NHC NHRPs for a totally unrelated IP address B and gets back the same ATM address 1. In this case normal NHRP behavior allows the NHC to use the VC (that was set up for destination A) for traffic to B.

Note: In this section we will refer to a VC set up as a result of an NHRP query/response as a shortcut VC.

If one expects to be able to label switch the packets being received from a shortcut VC, then the label switch needs to be informed as to exactly what traffic will arrive on that VC and that mapping cannot change without notice. Currently there exists no mechanism in the defined signaling of an shortcut VC. Several means are possible. A binding, equivalent to the binding in LDP, could be sent in the setup message. Alternatively, the binding of prefix to label could remain in an LDP session (or whatever means of label distribution as

appropriate) and the setup could carry a binding of the label to the VC. This would leave the binding mechanism for shortcut VCs independent of the label distribution mechanism.

A further architectural challenge exists in that label switching is inherently unidirectional whereas ATM is bi-directional. The above binding semantics are fairly straight-forward. However, effectively using the reverse direction of a VC presents further challenges.

Label switching must also respect the granularity of the shortcut VC. Without VC merge, this means a single label switched flow must map to a VC. In the case of VC merge, multiple label switched streams could be merged onto a single shortcut VC. But given the asymmetry involved, there is perhaps little practical use

Another issue is one of practicality and usefulness. What is sent over the VC must be at a fine enough granularity to be label switched through receiving domain. One potential place where the two

technologies might come into play is in moving data from one campus via the wide-area to another campus. In such a scenario, the two technologies would border precisely at the point where summarization is likely to occur. Each campus would have a detailed understanding of itself, but not of the other campus. The wide-area is likely to have summarized knowledge only. But at such a point level 3 processing becomes the likely solution.

#### [4.5](#) Operation in a Hierarchy

This section is FFS.

#### [4.6](#) Stacked Labels in a Flat Routing Environment

This section is FFS.

#### [4.7](#) Multicast

This section is FFS.

#### [4.8](#) Multipath

Many IP routing protocols support the notion of equal-cost multipath routes, in which a router maintains multiple next hops for one destination prefix when two or more equal-cost paths to the prefix exist. There are a few possible approaches for handling multipath with MPLS.

In this discussion we will use the term "multipath node" to mean a node which is keeping track of multiple switched paths from itself

for a single destination.

The first approach maintains a separate switched path from each ingress node via one or more multipath nodes to a merge point. This requires MPLS to distinguish the separate switched paths, so that learning of a new switched path is not misinterpreted as a replacement of the same switched path. This also requires an ingress MPLS node be capable of distributing the traffic among the multiple switched paths. This approach preserves switching performance, but at a cost of proliferating the number of switched paths. For example, each switched path consumes a distinct label.



The second approach establishes only one switched path from any one ingress node to a destination. However, when the paths from two different ingress nodes happen to arrive at the same node, that node may use different paths for each (implying that the node becomes a multipath node). Thus the switched path chosen by the multipath node may assign a different downstream path to each incoming stream. This conserves switched paths and maintains switching performance, but cannot balance loads across downstream links as well as the other approaches, even if switched paths are selectively assigned. With this approach is that the L2 path may be different from the normal L3 path, as traffic that otherwise would have taken multiple distinct paths is forced onto a single path.

The third approach allows a single stream arriving at a multipath node to be split into multiple streams, by using L3 forwarding at the multipath node. For example, the multipath node might choose to use a hash function on the source and destination IP addresses, in order to avoid misordering packets between any one IP source and destination. This approach conserves switched paths at the cost of switching performance.

#### [4.9](#) Host Interactions

There are a range of options for host interaction with MPLS:

The most straightforward approach is no host involvement. Thus host operation may be completely independent of MPLS, rather hosts operate according to other IP standards. If there is no host involvement then this implies that the first hop requires an L3 lookup.

If the host is ATM attached and doing NHRP, then this would allow the host to set up a Virtual Circuit to a router. However this brings up a range of issues as was discussed in [section 4.4](#) ("interoperation with NHRP").

On the ingress side, it is reasonable to consider having the first

hop LSR provide labels to the hosts, and thus have hosts attach labels for packets that they transmit. This could allow the first hop LSR to avoid an L3 lookup. It is reasonable here to have the host request labels only when needed, rather than require the host to

remember all labels assigned for use in the network.

On the egress side, it is questionable whether hosts should be involved. For scaling reasons, it would be undesirable to use a different label for reaching each host.

#### [4.10](#) Explicit Routing

There are two options for Route Selection: (1) Hop by hop routing, and (2) Explicit 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 an algorithm performed 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).

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 L3 packet. This implies that explicit routing with MPLS is relatively efficient (when compared with the efficiency of explicit routing for pure datagrams).

Explicit routing may be useful for a number of purposes such as allowing policy routing and/or facilitating traffic engineering.

##### [4.10.1](#) Establishment of Point to Point Explicitly Routed LSPs

In order to establish a point to point explicitly routed LSP, the LDP packets used to set up the LSP must contain the explicit route. This implies that the LSP is set up in order either from the ingress to the egress, or from the egress to the ingress.

One node needs to pick the explicit route: This may be done in at least two possible ways: (i) by configuration (eg, the explicit route may be chosen by an operator, or by a centralized server of some kind); (ii) By use of a routing protocol which allows the ingress and/or egress node to know the entire route to be followed. This would imply the use of a link state routing protocol (in which all nodes know the full topology) or of a path vector routing protocol (in which the ingress node is told the path as part of the normal

operation of the routing protocol).

Note: The normal operation of path vector routing protocols (such as BGP) does not provide the full set of routers along the path. This implies that either a partial source route only would be provided (implying that LSP setup would use a combination of hop by hop and explicit routing), or it would be necessary to augment the protocol in order to provide the complete explicit route. Detailed operation in this case is FFS.

In the point to point case, it is relatively straightforward to specify the route to use: This is indicated by providing the addresses of each LSR on the LSP.

#### 4.10.2 Explicit and Hop by Hop routing: Avoiding Loops

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 the explicit route is likely to follow a path which is inconsistent with the path followed by hop by hop routing. If some of the nodes along the path follow an explicit route but some of the nodes make use of hop by hop routing (and ignore the explicit route), then inconsistent routing may result and in some cases loops (or severely inefficient paths) may form. This implies that for any one LSP, there are two possible options: (i) The entire LSP may be hop by hop routed; or (ii) The entire LSP may be explicit routed.

For this reason, it is important that if an explicit route is specified for setting up an LSP, then that route must be followed in setting up the LSP.

There is a related issue when a link or node in the middle of an explicitly routed LSP breaks: In this case, the last operating node on the upstream part of the LSP will continue receiving packets, but will not be able to forward them along the explicitly routed LSP (since its next hop is no longer functioning). In this case, it is not in general safe for this node to forward the packets using L3 forwarding with hop by hop routing. Instead, the packets must be discarded, and the upstream partition of the explicitly routed LSP must be torn down.

Where part of an Explicitly Routed LSP breaks, the node which originated the LSP needs to be told about this. For robustness reasons the MPLS protocol design should not assume that the routing protocol will tell the node which originated the LSP. For example, it is possible that a link may go down and come back up quickly enough that the routing protocol never declares the link down. Rather, an

explicit MPLS mechanism is needed.

#### [4.10.3](#) Merge and Explicit Routing

Explicit Routing is slightly more complex with a multipoint to point LSP (i.e., in the case that stream merge is used).

In this case, it is not possible to specify the route for the LSP as a simple list of LSRs (since the LSP does not consist of a simple sequence of LSRs). Rather the explicit route must specify a tree. There are several ways that this may be accomplished. Details are FFS.

#### [4.10.4](#) Using Explicit Routing for Traffic Engineering

In the Internet today it is relatively common for ISPs to make use of a Frame Relay or ATM core, which interconnects a number of IP routers. The primary reason for use of a switching (L2) core is to make use of low cost equipment which provides very high speed forwarding. However, there is another very important reason for the use of a L2 core: In order to allow for Traffic Engineering.

Traffic Engineering (also known as bandwidth management) refers to the process of managing the routes followed by user data traffic in a network in order to provide relatively equal and efficient loading of the resources in the network (i.e., to ensure that the bandwidth on links and nodes are within the capabilities of the links and nodes).

Some rudimentary level of traffic engineering can be accomplished with pure datagram routing and forwarding by adjusting the metrics assigned to links. For example, suppose that there is a given link in a network which tends to be overloaded on a long term basis. One option would be to manually configure an increased metric value for this link, in the hopes of moving some traffic onto alternate routes. This provides a rather crude method of traffic engineering and provides only limited results.

Another method of traffic engineering is to manually configure multiple PVCs across a L2 core, and to adjust the route followed by each PVC in an attempt to equalize the load on different parts of the network. Where necessary, multiple PVCs may be configured between the same two nodes, in order to allow traffic to be split between

different paths. In some topologies it is much easier to achieve efficient non-overlapping or minimally-overlapping paths via this method (with manually configured paths) than it would be with pure datagram forwarding. A similar ability can be achieved with MPLS via the use of manual configuration of the paths taken by LSPs.

A related issue is the decision on where merge is to occur. Note that once two streams merge into one stream (forwarded by a single label)

then they cannot diverge again at that level of the MPLS hierarchy (i.e., they cannot be bifurcated without looking at a higher level label or the IP header). Thus there may be times when it is desirable to explicitly NOT merge two streams even though they are to the same egress node and FEC. Non-merge may be appropriate either because the streams will want to diverge later in the path (for example, to avoid overloading a particular downstream link), or because the streams may want to use different physical links in the case where multiple slower physical links are being aggregated into a single logical link for the purpose of IP routing.

As a network grows to a very large size (on the order of hundreds of LSRs), it becomes increasingly difficult to handle the assignment of all routes via manual configuration. However, explicit routing allows several alternatives:

1. **Partial Configuration:** One option is to use automatic/dynamic routing for most of the paths through the network, but then manually configure some routes. For example, suppose that full dynamic routing would result in a particular link being overloaded. One of the LSPs which uses that link could be selected and manually routed to use a different path.
2. **Central Computation:** One option would be to provide long term network usage information to a single central management facility. That facility could then run a global optimization to compute a set of paths to use. Network management commands can be used to configure LSRs with the correct routes to use.
3. **Egress Computation:** An egress node can run a computation which optimizes the path followed for traffic to itself. This cannot of course optimize the entire traffic load through the network, but can include optimization of traffic from multiple ingress's to one

egress. The reason for optimizing traffic to a single egress, rather than from a single ingress, relates to the issue of when to merge: An ingress can never merge the traffic from itself to different egresses, but an egress can if desired chose to merge the traffic from multiple ingress's to itself.

#### [4.10.5](#) Using Explicit Routing for Policy Routing

This section is FFS.

#### [4.11](#) Traceroute

This section is FFS.

Callon et al.

Expires January 30, 1998

[Page 47]

---

INTERNET DRAFT

A Framework for MPLS

July 30, 1997

#### [4.12](#) 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 (on startup) only the egress node passes label bindings to its neighbors corresponding to any FECs which leave the MPLS network at that egress node. When initializing, 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.

This may appear to be an advantage of egress LSP control (since with egress control the initial label assignments "bubble up" from the egress to upstream nodes, and consistency is therefore easy to ensure). However, 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.

Egress control and local control can interwork in a very straightforward manner: 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 techniques therefore becomes a tradeoff between avoiding a short period of initial thrashing on startup (in the sense of avoiding the need to withdraw inconsistent labels which may have been assigned using local control) versus the imposition of a short delay on initial startup (while waiting for the initial label assignments to bubble up from downstream). The protocol mechanisms which need to be defined are the same in either case, and the steady state operation is the same in either case.

#### 4.13 Security

Security in a network using MPLS should be relatively similar to security in a normal IP network.

Routing in an MPLS network uses precisely the same IP routing protocols as are currently used with IP. This implies that route filtering is unchanged from current operation. Similarly, the

security of the routing protocols is not effected by the use of MPLS.

Packet filtering also may be done as in normal IP. This will require either (i) that label swapping be terminated prior to any firewalls performing packet filtering (in which case a separate instance of label swapping may optionally be started after the firewall); or (ii) that firewalls "look past the labels", in order to inspect the entire IP packet contents. In this latter case note that the label may imply semantics greater than that contained in the packet header: In particular, a particular label value may imply that the packet is to take a particular path after the firewall. In environments in which this is considered to be a security issue it may be desirable to terminate the label prior to the firewall.

Note that in principle labels could be used to speed up the operation of firewalls: In particular, the label could be used as an index into a table which indicates the characteristics that the packet needs to have in order to pass through the firewall. Depending upon implementation considerations matching the contents of the packet to the contents of the table may be quicker than parsing the packet in the absence of the label.

## References

- [1] "A Proposed Architecture for MPLS", E. Rosen, A. Viswanathan, R. Callon, work in progress, [draft-rosen-architecture-00.txt](#), July 1997.
- [2] "ARIS: Aggregate Route-Based IP Switching", A. Viswanathan, N.

Callon et al.

Expires January 30, 1998

[Page 49]

---

INTERNET DRAFT

A Framework for MPLS

July 30, 1997

- 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>
- [6] "Tag distribution Protocol", Doolan, Davie, Katz, Rekhter, Rosen, work in progress, internet draft <[draft-doolan-tdp-spec-01.txt](#)>
- [7] "Use of Tag Switching with ATM", Davie, Doolan, Lawrence, McGloaghrie, Rekhter, Rosen, Swallow, work in progress, Internet Draft <[draft-davie](#)- tag-switching-atm-01.txt>
- [8] "Label Switching: Label Stack Encodings", Rosen, Rekhter, Tappan, Farinacci, Fedorkow, Li, work in progress, internet draft <[draft-rosen](#)- tag-stack-02.txt>
- [9] "Partitioning Tag Space among Multicast Routers on a Common Subnet", Farinacci, work in progress, internet draft <[draft-farinacci-multicast](#)- tag-part-00.txt>
- [10] "Multicast Tag Binding and Distribution using PIM", Farinacci, Rekhter, work in progress, internet draft <[draft-farinacci-multicast-tagsw](#)- 00.txt>
- [11] "Toshiba's Router Architecture Extensions fir ATM: Overview", Katsube, Nagami, Esaki, [RFC2098](#).TXT.
- [12] "Soft State Switching: A Proposal to Extend RSVP for Switching RSVP Flows", A. Viswanathan, V. Srinivasan, work in progress, Internet Draft <[draft-viswanathan-aris-rsvp-00.txt](#)>, March 1997.
- [13] "Integrated Services in the Internet Architecture: an Overview", R. Braden et al, [RFC 1633](#), June 1994.
- [14] "Resource ReSerVation Protocol (RSVP), Version 1 Functional Specification", work in progress, [draft-ietf-rsvp-spec-16.txt](#), June 1997

- [15] "OSPF version 2", J. Moy, [RFC 1583](#), March 1994.
- [16] "A Border Gateway Protocol 4 (BGP-4)", Y. Rekhter and T. Li, [RFC 1771](#), March 1995.

- [17] "Ipsilon Flow Management Protocol Specification for IPv4 Version 1.0", P. Newman et al., [RFC 1953](#), May 1996.
- [18] "ATM Forum Private Network-Network Interface Specification, Version 1.0", ATM Forum af-pnni-0055.000, March 1996.
- [19] "NBMA Next Hop Resolution Protocol (NHRP)", J. Luciani et al., work in progress, [draft-ietf-rolc-nhrp-11.txt](#), March 1997.

#### Author's Addresses

Ross Callon  
Ascend Communications, Inc.  
1 Robbins Road  
Westford, MA 01886  
508-952-7412  
rcallon@casc.com

Paul Doolan  
Cisco Systems, Inc  
250 Apollo Drive  
Chelmsford, MA 01824  
508-634-1204  
pdoolan@cisco.com

Nancy Feldman  
IBM Corp.  
17 Skyline Drive  
Hawthorne NY 10532  
914-784-3254  
nkf@vnet.ibm.com

Andre Fredette  
Bay Networks Inc  
3 Federal Street  
Billerica, MA 01821  
508-916-8524  
fredette@baynetworks.com

George Swallow  
Cisco Systems, Inc  
250 Apollo Drive  
Chelmsford, MA 01824

508-244-8143  
swallow@cisco.com

Arun Viswanathan  
IBM Corp.  
17 Skyline Drive  
Hawthorne NY 10532  
914-784-3273  
arunv@vnet.ibm.com

