

Active Networks and MPLS integration for better access networks

Framework and Applicability

Sanda Dragos

Supervised by **Dr. Martin Collier**

School of Electronic Engineering - DCU

Switching and Systems Laboratory

Contents

1	Introduction	1
2	From IP and ATM to MPLS	4
2.1	The Internet	4
2.1.1	History	4
2.1.2	Internet concepts	5
2.2	ATM Networks	6
2.2.1	History	7
2.2.2	ATM Networks concepts	8
2.3	Argument for integrating IP and ATM	9
2.4	IP over ATM Techniques	11
2.4.1	Classical IP over ATM	12
2.4.2	Address Resolution protocol (ATM-ARP)	14
2.4.3	Next Hop resolution Protocol (NHRP)	14
2.4.4	LAN emulation (LANE)	15
2.4.5	Multiprotocol over ATM (MPOA)	17
2.4.6	Limitations of IP over ATM approaches	17
2.5	IP Switching Techniques	19
2.5.1	Cell Switch Router (CSR)	20
2.5.2	Ipsilon's IP Switching	21

2.5.3	Aggregate Route-Based IP Switching (ARIS)	23
2.5.4	Cisco's Tag Switching	25
2.6	Multiprotocol Label Switching (MPLS)	27
2.6.1	Short overview of MPLS Architecture	29
2.6.2	The role of MPLS in the Internet Architecture	39
3	Internet Traffic Engineering	41
3.1	Traffic Engineering overview	41
3.1.1	Traffic Engineering in Classical Telephone Networks . .	41
3.1.2	Evolution of Traffic Engineering in Packet Networks . .	43
3.1.3	The Overlay Model	46
3.1.4	Constrained-Based Routing	47
3.1.5	Other IETF Projects Related to Traffic Engineering . .	47
3.2	Traffic Engineering concepts	53
3.2.1	Traffic Engineering Performance Objectives	53
3.2.2	Traffic Engineering Process Model	53
3.2.3	Traffic Engineering Taxonomy	54
3.3	Traffic Engineering and MPLS	58
3.3.1	Concepts	58
3.3.2	Current state-of-the-art	62
4	Active Networks	67
4.1	Active Networks Overview	68
4.1.1	Architectures	69
4.1.2	Applications	75
5	Active Networks and MPLS	78
5.1	Overview	78
5.2	Our Approach	79

5.2.1	Motivation	79
5.2.2	Description	80
5.3	Our Implementation	81
5.3.1	The prerequisite	82
5.3.2	Implementation	87
5.4	Possible applications	90
5.4.1	Applications that modify the packet content	90
5.4.2	Applications that change a packet's path	91
6	Conclusions	93
6.1	Future work	94

List of Figures

2.1	Multiple IP LIS's on one ATM network	13
2.2	A LANE Network.	16
2.3	(a)Physical topology. (b)Resulting logical connectivity.	18
2.4	(a)An IP Switching Network. (b)The structure of an IP Switch.	22
2.5	IBM ARIS Switched Paths	25
2.6	The MPLS cloud and its main components	29
2.7	MPLS: Basic principles	31
2.8	MPLS Label. MPLS shim	32
2.9	MPLS Data structures needed for an unlabelled packet	34
2.10	MPLS Data structures needed for an labelled packet	35
4.1	Application-specific processing within the nodes of an Active Network	69
5.1	Packet processing within the nodes of an MPLS Active Network	80
5.2	An active MPLS architecture	81
5.3	(a)The Linux Kernel (b)The User-Mode Linux Kernel	83
5.4	The Netfilter "hooks" defined in IPv4	85
5.5	The Netfilter routine	86
5.6	A minimal MPLS network	87
5.7	A MPLS node modified to programme the packets	88

5.8 Our experiment test	89
-----------------------------------	----

List of Tables

2.1	Signaling architectures of CR-LDP and RSVP-TE	38
2.2	Signaling support for traffic engineering features in CR-LDP and RSVP-TE	40
3.1	Traffic Trunk Operations	60
3.2	Traffic Trunk Attributes	61

Abstract

As the Internet architecture has evolved over time, adapting to the needs of its users and incorporating new technologies, the introduction of MPLS has immediate applications in traffic engineering and virtual private networks. Therefore MPLS becomes a presence in core networks as well as access areas.

Active Networks represent a new approach to network architecture in which routers can perform computations on user data, while packets can carry programs to be executed on routers and possibly change their state.

In this document, we propose the integration of Multiprotocol Label Switching with the Active Network concept as a solution for the future access networks. The resulting network will provide all the features of MPLS, and in addition, will use active packets to support packet processing.

This integration will be a major benefit for a wide range of applications which will extend the capabilities of either technologies.

Chapter 1

Introduction

The main function of a network is to deliver packets from one end-point to another. This was also the main goal of Internet in its first days. Since then, however, the Internet evolved, expanding in a geometrical proportion, surpassing all other technologies preceding it, including radio, television, and personal computers. Therefore, new solutions and technologies advent as a response to the challenge of rapid growth in number of users and increasing requirements for service quality, reliability, and efficiency.

MPLS is such a technology, which brings several benefits to the Internet Protocol networks by:

- increasing packet forwarding performance;
- maximizing resource usage and optimizing traffic flow using traffic engineering mechanisms;
- minimize downtime and path prioritization to provide service differentiation.

Active Networks is another Internet “solution”, which addresses the problem of slow network service evolution by building programmability into the network. Therefore, the idea of messages carrying procedures and data is a natural step beyond traditional circuit and packet switching and can be used to rapidly adapt the network to changing requirement.

Due to its capabilities for traffic engineering and quality of service, MPLS evolves beyond the core network, penetrating the access area. The big issue that arises is its inability to perform packet computations for services often encountered in an Internet Service Provider network (such as firewalls, Web proxies, multicast routers, or mobile routers). Active Networks is a novel solution for implementing such services providing a flexible network infrastructure with increased capabilities.

Our goal in this document is to demonstrate how and why we would integrate the two technologies: Active Network and MPLS.

The rest of the document is organized as it follows:

Chapter 2 offers an overview over the evolution of the effort of integrating the connectionless IP networks with connection oriented ATM ones in order to provide traffic engineering and quality of service, while still having a robust and scalable network. All those efforts converged into one protocol called *Multiprotocol Label Switching (MPLS)*, which is also shortly described in this chapter.

An overview of Internet Traffic Engineering and MPLS Traffic Engineering is presented in **Chapter 3**. The advantages and capabilities of MPLS for Traffic Engineering are also outlined in this chapter.

Active Networks concept, architectures, and applications are presented in **Chapter 4**.

Chapter 5 describes our work in trying to prove that the integration of Active Networks and MPLS is a viable a worthwhile solution for the next generation of access networks. Therefore, this chapter contains the motivation, an framework that proves that such integration is possible and other possible application that may be developed in this area. An overview of other similar proposals is also presented.

We conclude this document with some final remarks and a presentation of future work in **Chapter 6**.

Chapter 2

From IP and ATM to MPLS

“Before building the network of future, we must first understand what exists.” S. Keshav

2.1 The Internet

The Internet is a loose collection of networks organized into a multilevel hierarchy using a wide variety of interconnecting technologies.

2.1.1 History

The Internet started as a research project in the late 1960's by connecting four computers. Since then, it had grown exponentially to interconnect tens of millions of computers all over the world.

The fundamental basis of the Internet, that is, store-and-forward packet switching, was invented in 1961 by Baran in the United States. After few years, the U.S. Department of Defense Advanced Research Project Agency (DARPA) sponsored a communication network called ARPAnet. ARPAnet

was proposed in 1966 and built in late 1969. Although ARPAnet was successful, it was restricted to sites that received funding from DARPA. However, many other universities were interested in obtaining a connection to a packet-switched network, even if they did not received DARPA funding. This led to construction of other ARPA-like networks. By the early 1980s, ten of such networks joined to form the Internet.

One reason for the explosive growth of computer networking in the 1980s was the invention of Ethernet, a cheap Local Area Network (LAN) technology. Developed by researchers at Xerox's Palo Alto Research Center in the late 1970s, Ethernet allowed computers within a small campus to be interconnected with little effort. Small switches called *bridges* allowed several Ethernet's to talk to each other, expanding the scale of campus network. In 1992 commercial organizations were allow free access to the network.

The dominant application on the Internet is the World Wide Web. Invented as a way for high-energy particle physicists to share experimental data and results, the Web has increased even more the usage of Internet.

2.1.2 Internet concepts

The Internet Protocol (IP), the Layer 3 (network layer) protocol used by the Internet, sends information by breaking up messages into packets. The packets are then sent from a source host to a destination host via routers, which determine the best path for each packet. The router uses a routing protocol to decide the best path locally. In this way packets travel from router to router until they reach the router of the destination host. This method of storing packets, determining the next router or hop, and then forwarding the

packet is often referred to as store-and-forward or hop-by-hop routing. This hop-by-hop routing enables a relatively inexpensive, robust and scalable network. Consequently IP is connection-less, meaning all the information needed for routing is contained in the packet. These features have allowed for the rapid growth of the Internet but IP is a best-effort approach meaning that the delay of the packets is not guaranteed, only minimized. Thus IP networks cannot provide any Quality of Service (QoS)¹ to the user making the use of delay sensitive applications such as video-conferencing not possible. The need for QoS combined with the rapid growth demands new solutions. However, new solutions must be compatible with existing IP networks while improving the performance for minimal cost. A likely solution is to run IP on a fast Layer 2 (data link layer) technology. Asynchronous Transfer Mode (ATM) has emerged as the leading candidate because of its high speed. However, certain characteristics of ATM make the job of integrating ATM with IP difficult.

2.2 ATM Networks

Asynchronous Transfer Mode (ATM) networks aim to combine the flexibility of the Internet with per-user quality-of-service guarantees of the telephone network.

ATM networks are based on some important concepts: virtual circuits, fixed-size packets or *cells*, small packet size, statistical multiplexing and integrating

¹[1] defines *QoS* as a set of service requirements to be met by the network while transporting a flow.

services. All those elements make possible the existence of multiple classes of traffic with quality-of-service guarantees provided to individual streams.

2.2.1 History

Though ATM networks have been a topic of great interest since the late 1980s, the basic ideas in this field date back to the 1960s. In 1961, a group of researchers from Osaka University suggested that adding a header to each time slot could make a time-division multiplexor more efficient. At about the same time, Baran proposed a datagram approach to networking, recognizing that meta-data allows flexibility and efficiency.

In 1968 it was discovered a technique called *asynchronous time division multiplexing* (ATDM). In 1968-1969 the concept of virtual circuits and fixed-sized packet's were proposed. The name "virtual circuit" was meant to suggest a similarity with virtual memory, since neither a virtual circuit nor a virtual memory takes up resources until it is used. In 1970 Fraser and Vollaro from Bell Labs started to emulate a ATDM switch in software. In 1972 a ATDM network was fully operational and served as a research test bed for many years.

By the mid-1980's, telephone companies decided to substitute the Plain Old Telephone Service (POTS) with the Integrated Services Digital Network (ISDN). An ISDN line contains two data lines, each at 64 Kbps, and one control line, at 16 Kbps. The two data lines can be merged to obtain an effective bandwidth of 128 Kbps. ISDN has not been two popular because of its low bandwidth. That is why the International Consultative Committee of Telegraph and Telephone (CCITT) proposed B-ISDN. This was similar with

ISDN except that the data channel would be wider. At about this time, Stratacom, AT&T, and others proposed that the technology for B-ISDN ought to be ATDM. By the late 1980s, CCITT agreed that the B-ISDN network would be build, using ATDM, and the international standard was called ATM.

When the news came to the attention of computer companies, they realized that ATM is another form of packet switching. ATM networks had the advantage over existing LANs because they offer *scalable bandwidth*. Traditional shared medium LANs, such as Ethernet and Fiber Distributed Data Interface (FDDI), were restricted to a single speed - 10 Mbps for Ethernet and 100 Mbps for FDDI. To upgrade an Ethernet or FDDI network to a higher speed, every endpoint in the network must be modified. In contrast, ATM switches can multiplex data received at different rates, so an end-point can increase its network bandwidth independent of other end-points.

2.2.2 ATM Networks concepts

ATM uses short, fixed length packets that allow for fast switching and low latency. This is the main reason for the increasing popularity of ATM. ATM is a connection-oriented technology meaning that a point to point connection is set up for every message sent. The connection remains for the duration of the message and is then torn down. Because the connection, or virtual circuit, is setup prior to the message transfer, a guaranteed bandwidth can be negotiated between the sender and the network. Therefore, an ATM network allows for QoS guarantees (useful for real time applications), which combined with the falling price, make ATM an attractive technology. However, ATM is a complex technology and network management remains a difficult problem.

2.3 Argument for integrating IP and ATM

The advantages of ATM networks are many. Using a fixed length label for forwarding and a relative small fixed length of cells, ATM allows very simple forwarding in hardware. The fixed-size cell also simplifies buffering. Since cells are small they can be forwarded with minimum latency. Also, the fact that cell forwarding is so simple allows products that are based on cell forwarding to be fast. The use of cells comes with one disadvantage: the associated overhead. With ATM, each 48-byte cell carries a 5-byte header, implying approximately a 10% overhead. In addition, since cells are fixed length, whenever an IP packet is not an exact fit into n cells, there can be some capacity wasted in the last cell.

Another ATM advantage is the use of VC's (virtual circuits). The advantage of VC's is traffic engineering. Traffic engineering refers to the process of managing the routes followed by the user data traffic in a network in order to provide relative equal and efficient loading of the resources in the network. The IP "hop-by-hop" routing has a tendency to concentrate all traffic onto the least cost path and that makes difficult to do load balancing and traffic engineering in IP. Using VC's we can use explicit routing for efficient the use of the routes taken by the user data traffic. This allows, for example to coordinate the set of paths used by all VC's originated at (or destined to) a particular node in order to load the network in a relatively equal manner.

VC's may also be used to meet the requirement for QoS routing. This implies

to reserve specific resources for specific flows of user data. In some cases, the most direct path may already be fully reserved and unable to support a particular addition flow of data. For those cases, it will be necessary to route around congestion. This can be accomplished by using explicit routing to set up the path.

Having all those advantages the question which arises is *Can we get rid of datagrams entirely? Can we have ATM running from desktop to desktop?* The answer to those questions is *NO. Why?* Because switching scales very well in terms of bandwidth, but not as well in terms of network size. Internet relies on hierarchical routing in order to scale in size, while switching is inherently flat routing. The Internet's hierarchical routing employs *route aggregation*. The basic method of route aggregation makes use of address summarization using prefixes. That means that a partial address (the prefix address) may be used to represent multiple individual addresses. For addresses prefixes to be used in this manner, addresses have to be assigned such that any address starting with a particular prefix is located in a contiguous part of network, so that the assignment of addresses follows the topology of the network.

The major problem with end-to-end VC's is the amount of state information required, and the associated complexity of trying either to maintain state for each of many small VC's or to multiplex many small VC's onto a smaller number of large VC's in the core of the network. The use of IP datagram protocol in the edge of the network allows a large volume of user data to be multiplexed toward the core of the network and therefore eliminates the problem with end-to-end VC's. This implies that we need both: datagrams

and VC's. That's why many following protocols tried to integrate datagrams and circuit switching. We can group those protocols into two main classes:

- IP over ATM protocols
- IP switching protocols

2.4 IP over ATM Techniques

One way of integrating IP and ATM was that IP would treat ATM as a link-level technology (like Ethernet or FDDI), ignoring the routing and the quality-of-service aspects of ATM. This approach was called IP over ATM.

Many different approaches have been proposed for IP over ATM. Fundamental differences between IP and ATM make integration difficult. There were a number of issues to settle because of those differences in the protocols:

- connection-less (IP) versus connection oriented (ATM);
- packets (IP) versus cells (ATM);
- broadcast LANs (IP) versus point-point connections (ATM).

The main IP over ATM protocols are:

1. *Classical IP over ATM*
2. *ATM Address Resolution Protocol (ATM ARP)*
3. *Next Hop Resolution Protocol (NHRP)*
4. *LAN emulation (LANE)*
5. *Multiprotocol over ATM (MPOA)*

2.4.1 Classical IP over ATM

The classical IP over ATM was originally defined in [2] (Classical IP and ARP over ATM, January 1994). This RFC was made obsolete and replaced by [3] (Classical IP and ARP over ATM, April 1998). It is also known as the overlay model because IP is “layered over” ATM almost identical to how IP is “layered” over Ethernet, frame relay, and other link-layer technologies. IP and ATM use separate address spaces and routing protocols, and IP routers and ATM switches are functionally separate from each other. Because IP and ATM use separate address spaces, an address resolution protocol is required to map between them, similar to the address resolution protocol (ARP) used on broadcast LANs. The overlay model uses independent routing for IP and ATM. ATM uses for routing the VC’s, the ATM Forum private network node interface (PNNI) protocol [4], while IP uses Internet standards such as “open shortest path first” (OSPF) [5] and border gateway protocol (BGP) [6].

To layer IP over ATM, it is necessary to make ATM look like a typical link-layer subnetwork. Because ATM networks can support many more stations (hosts and routers) than are found on typical LANs, a single ATM network can have more than one IP subnetwork layered on top of it. Each of these subnetworks is called a logical IP subnetwork (LIS).

Figure 2.1 shows how several LIS’s can be layered over an ATM network. In such a network, two IP stations in the same LIS can communicate directly via ATM SVC’s, while IP stations from different LIS’s must intercommunicate via a router. For example, hosts A and B, which are in the same LIS, intercommunicate directly by opening an ATM SVC from one to another when there is a data to send. Host C, however, must indirectly communicate

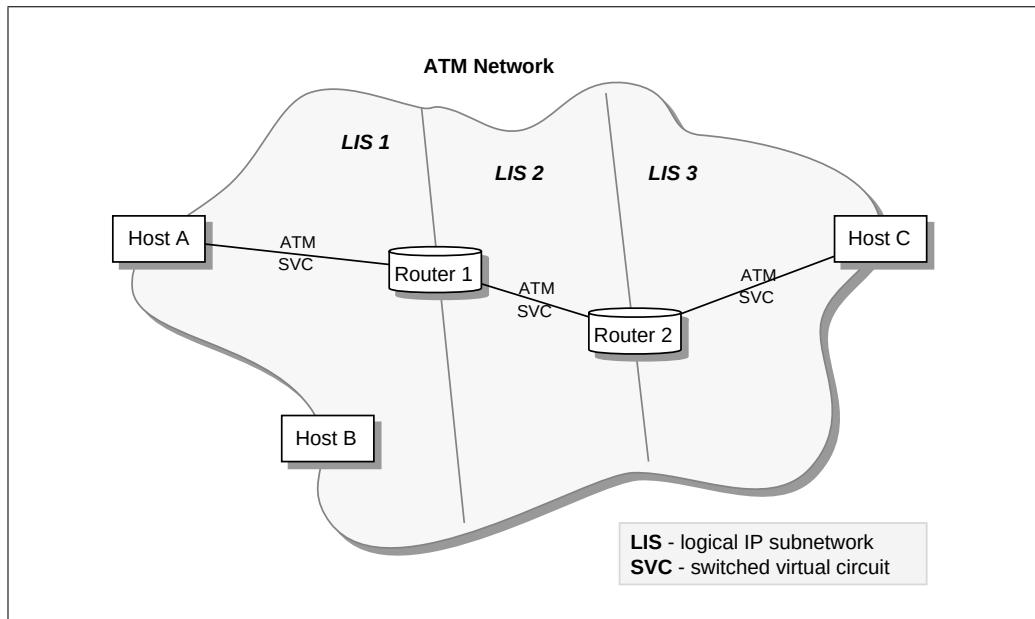


Figure 2.1: Multiple IP LIS's on one ATM network

with hosts A and B by opening an SVC to router 2, which will route the packets to router 1, which will in turn open an SVC to host A and/or B.

This approach has the advantages of being the closest possible of adaptation of “classical” IP to work over ATM and of not requiring any changes to ATM switches and protocols because all the work is confined in IP layer.

The disadvantages of this approach are that it does not solve the delay problem presented in the store-and-forward network; packets sent to a host outside of the source LIS will be routed; and it does not support point-to-multipoint connections like in IP networks.

2.4.2 Address Resolution protocol (ATM-ARP)

Classical IP over ATM defines an ATMARF server and protocol [2, 3] to be used for resolving IP addresses to ATM addresses. This solution is in contrast with the approach used for IP over Ethernet, which makes use of broadcast to transmit ARP requests to all of the other stations on the LAN. Because ATM does not have a broadcast mechanism, a server must be used instead. Each LIS requires an ATMARF server to translate from IP to ATM addresses.

Every host within a LIS has to register its ATM and IP addresses with a configured ATMARF server. Then, it may use the ATMARF server to request the IP address for any host in the same LIS corresponding with a certain ATM address in order to be able to open a SVC for sending one or more packets.

2.4.3 Next Hop resolution Protocol (NHRP)

As we described in 2.4.1, hosts which are in separate LIS's must communicate via a router, even when it means that the packets must leave the ATM network, only to reenter the ATM network. This process is inefficient because the cells must be reassembled as they enter the router, each packet must be processed and switched by the router, and they must be re-segmented into cells as they enter the ATM network. In addition, if the application sending the packet had any particular QoS requirements for the flow of IP packets, the QoS provided by the ATM network would be lost at each router hop.

To eliminate this inefficiency, the IETF developed the NHRP [7], which al-

allows IP hosts in different LIS's, but on the same ATM network, to be able to open direct VC's between each other.

NHRP can be viewed as an enhancement of ATMARP. ATMARP clients (end stations) became NHRP clients, and NHRP servers (NHS's) replace ATMARP servers. Like ATMARP, NHRP is based on a query/response model: NHRP client that wish to forward IP packets destined to other hosts send an NHRP query to their local NHS. The NHS forwards the query along the direction that a packet to the same destination will take, until it reaches either an NHS that would forward the packet off of the ATM network or an NHS that knows the ATM address of the destination host. At this point, an NHRP response is sent back to the originating client indicating the ATM address to use to reach the specific destination. Then, the client that originated the query opens a direct point-to-point SVC to either the destination host or the router that is to forward the IP packet off the ATM network.

A drawback of this method it would be that the response returned is based on the state of the IP routing at the time that the query is sent. If the route that should be followed changes, there is no provision for the originating NHRP client to be notified. In such cases the packets are misrouted.

2.4.4 LAN emulation (LANE)

When it first started to promote the widespread use of ATM, the ATM Forum, realized that it needed to make ATM easy to use for networking applications. Therefore, the forum provided it with the ability to emulate existing LAN's [8].

The LAN emulation (LANE) is implemented in the host's ATM network interface cards (NIC's). No changes are required to the host's code. The NIC's present a LAN interface to the host software and adapt the LAN protocols to the ATM environment. Because LAN protocols require LAN broadcast facilities LANE uses a broadcast server to emulate the LAN broadcasting. LANE also uses configuration and media access control (MAC)-to-ATM address resolution servers to support the LANE service. Eventually, we can say that the operation of LANE is based on LANE configuration servers (LECS's), LANE servers (LES's), and broadcast unknown servers (BUS's).

A LANE Network is described in Fig 2.2:

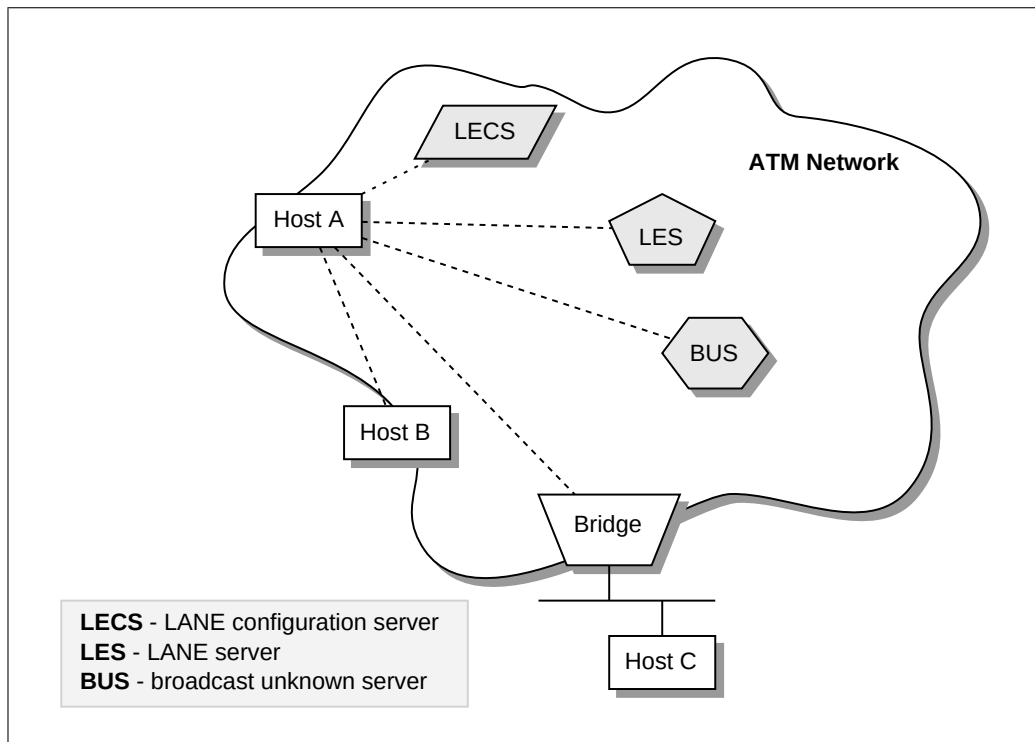


Figure 2.2: A LANE Network.

Each LANE client opens a VC to the LECS. The LECS informs the client of the address of the correct LES that the client should contact. The client opens a SVC to the LES in order to register itself. If later on, the client has traffic to send to another client, then it sends a LANE address resolution protocol (LE-ARP) request to the LES to determine the ATM address of the other client. In some cases, the LES may not know the ATM address corresponding to the MAC address specified in an LE-ARP. For example, the client may be attached via a proxy client. In this case, the client needs to send the packet to the BUS. The BUS then sends the packet to all proxy LANE clients. The return traffic will allow the location of the previously unknown client to be learned.

Although LANE solution is preferable to IP over ATM because all the additional facilities it offers, however, the additional layer of overhead, protocol processing, and interactions with a second set of servers adds to the processing delay in the network and is a more expensive solution for the end user.

2.4.5 Multiprotocol over ATM (MPOA)

MPOA [9] is a combination of LANE and NHRP, combining the virtual networking ability of LANE with the address resolution ability of NHRP. An MPOA client has both LANE and NHRP client capabilities, and MPOA servers are extensions to NHS's.

2.4.6 Limitations of IP over ATM approaches

The main problem with the techniques described previously is scaling. IP routing protocols scale on the order of the number of router-to-router

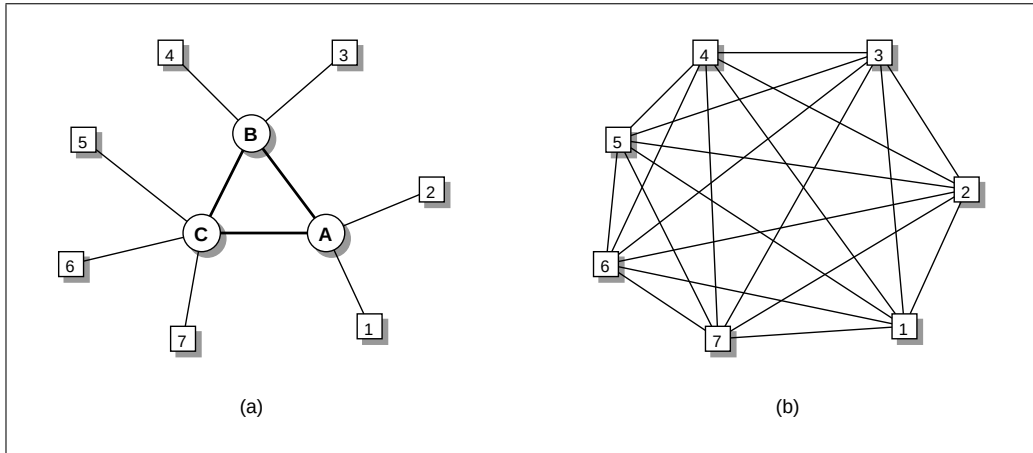


Figure 2.3: (a)Physical topology. (b)Resulting logical connectivity.

connection in the routing topology, be it physical or virtual. In the overlay model, as you can see in Figure 2.3, even if the number of physical links is not so big, the number of logical links is, because the total number of logical links that are advertised between the n ATM-attached routers equals $n * (n - 1) / 2$ links. As networks grow, the number of entries in the routing protocol that is required to describe the logical topology begins very quickly to greatly exceed the number of physical devices and links that exist in the network.

Breaking up the overlay into multiple IP subnetworks solved the scaling problem with the overlay model. While reducing the number of route connections, this creates another problem: inefficient paths for data traffic. This second problem has led to the addition of protocols such as NHRP to the overlay model. These additional protocols increase the complexity of the overlay model and make its performance depend on traffic-sensitive caching techniques.

The overlay method of running IP over ATM also suffers from other problems such as:

1. excessive management complexity and overhead associated with running two separate routing protocols;
2. difficulty of representing the ATM subnetwork in a way that IP routing can deal with yet that accurately reflects the capabilities of the ATM subnetwork;
3. resulting difficulty in extending the approach to QoS routing, traffic engineering, and similar advanced features.

2.5 IP Switching Techniques

We have reviewed the techniques that have been developed to support the overlay model of IP and ATM. The alternative for all those techniques is to let the ATM switches to participate in IP routing. This means that IP and ATM routing is based on IP routing protocols. Such proposals are called *IP switching techniques*. Among the most popular *IP Switching Techniques* are Ipsilon's IP switching, Cisco's Tag switching, IBM's ARIS and Toshiba's CSR. Their common characteristic is a multi-layer label-swapping mechanism, realized by:

- providing a semantic to bind labels to specific streams of packets;
- using a protocol to distribute binding information among routers;
- forwarding packets from incoming interface to outgoing interface based solely on the label information, not the destination IP address.

The forwarding can be done in hardware by the switch fabric of the router, or it can be done in software by indexing the label of the incoming packet into the Tag Information Base (TIB) to find out the corresponding outgoing interface. The result is a router with the speed of a link-layer (layer-2) switch and the flexibility of a network-layer (layer-3) router.

The main IP Switching protocols are:

1. *Toshiba's Cell Switch Router (CSR)*
2. *Ipsilon's IP Switching*
3. *IBM's Aggregate Route-Based IP Switching (ARIS)*
4. *Cisco's Tag Switching*
5. *Multiprotocol Label Switching*

2.5.1 Cell Switch Router (CSR)

The CSR proposal by Toshiba [10] is one of the first attempts to implement IP switching. Essential to the proposal is the notion of a “*cell switch router*”(CSR), which is a device that interconnects logical IP subnetworks (LIS's) and is capable of both IP forwarding and ATM cell switching. Within a LIS, running LANE or classical IP over ATM provides connectivity among hosts and/or routers. The address resolution depends upon ATMARP [3] and InATMARP [11] servers. Connectivity that spans multiple LIS's is provided via CSR's that interconnect LIS's. The CSR tries to identify individual traffic flows and to bind each flow to a virtual circuit. This method is called “*cut-through*” or “*shortcut routing*”.

A signaling protocol is needed to establish new VPI/VCI values for specific flows of IP packets arriving at an input interface. Then, these special values could be bound to corresponding VPI/VCI values to an output interface. In this way a cell arriving with one VPI/VCI value could be switched at the ATM layer to the appropriate output interface and could be assigned the correct VPI/VCI for forwarding to the next hop router or end station.

The tag binding creation is driven by either RSVP messages or data traffic. The distribution and maintenance of tag binding information is done via a separate protocol: the flow attribute notification protocol (FANP) [12].

2.5.2 Ipsilon's IP Switching

IP Switching is a technology proposed by Ipsilon [13] and gained notoriety in the mid 1990s. It is very similar in many respects with Toshiba's *Cell Switch Router*.

In Ipsilon's IP Switching proposal, the main element is the IP Switch. An IP Switch is made by taking the hardware of an ATM switch and by removing completely the software resident in the control processor above AAL-5. Therefore, the signaling, any existing routing protocol, and any LAN emulation server or address resolution servers are removed. A simple low-level control protocol, called general switch management protocol (GSMP) [14], replaces the ATM software. GSMP gives the IP switch controller access to the switching hardware. The IP switch controller is a processor running standard IP router software with extension that allows it to make use of the switching hardware. You can see the structure of the IP switch, as well as an example of an IP Switching network in Figure 2.4:

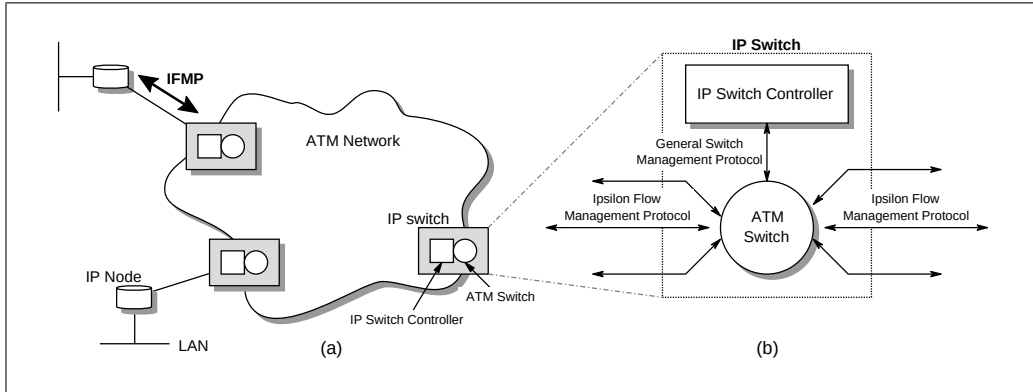


Figure 2.4: (a)An IP Switching Network. (b)The structure of an IP Switch.

Up to this point, existing proposals relied on use of native ATM signaling to establish at least default ATM virtual circuits. Ipsilon Networks abandoned the standard ATM signaling and introduced a new signaling protocol, which associates IP flows with ATM virtual channels. This protocol was called Ipsilon Flow Management Protocol (IFMP) [15] and it was used for establishing VPI/VCI values used by neighboring ATM switches for specific flows.

The Ipsilon approach had the advantage, relative to Toshiba's CSR proposal, of being potentially able to reduce the default-forwarding load by a large percentage of IP packets. Unlike CSR, however, IFMP depend to a large degree on flow detection at each IP routing node in a network composed of IFMP-participating IP routers. This could result in a significant overhead in IP packet processing in the default-forwarding mode.

2.5.3 Aggregate Route-Based IP Switching (ARIS)

ARIS was almost solely sponsored by IBM, though it was also under development as an open IETF standard. It was intended for use with switched network technologies, whether ATM, frame relay, or LAN switches and permits layer 2 switching to be used for IP datagram forwarding. Although it was intended to apply to multiple media, it mainly focuses on ATM by proposing *VC merging*. That means that packets arriving with different VPI/VCI values can be forwarded with the same VPI/VCI value (merged).

ARIS also proposes the *route-based* paradigm for assigning the labels. A route in this sense is rather like a multicast distribution tree, rooted at the egress point, and traversed in reverse. The egress point is specified by an “*egress identifier*”, which may be one of:

- IP destination prefix
- egress router IP address
- OSPF Router ID
- multicast (source, group) pair
- multicast (ingress-of-source, group) pair

Using the IP destination prefix, packets from any ingress point forming a leaf on the route tree, and intended for that egress point’s destination prefix, are switched and merged to the root egress. This is useful in a relatively small (campus or small enterprise) environment running RIP, but does not scale to backbones. The egress router IP address is used primarily for BGP protocol updates. OSPF Router ID is most generally useful, supporting both IPv4

and IPv6, and allowing aggregation of all OSPF-routed datagrams.

The main element in an ARIS network is the *Integrated Switched Router*(ISR). An ARIS network (or a network comprised of ARIS capable ISRs) establishes switched paths to egress points, independently of any traffic. These egress points are established using the standard layer 3 routing protocols such as OSPF and/or BGP. It is the responsibility of the egress ISR to initiate the path setup by sending messages (*establish messages*) to upstream neighbors, as you can see in Figure 2.5. These neighbors forward *establish messages* upstream in reverse path multicast style, so eventually all ARIS ISRs have switched paths to every egress ISR.

An important element of ARIS is that switched paths are guaranteed to be loop-free, despite the use of standard IP routing protocols. Each ISR appends its own ISR ID to *establish messages* it forward, so it can then determine whether the *establish message* has passed before. If so, it means that there is a loop and refuses to continue the path.

Another ARIS aspect is that the path information is soft state. That means that it is maintained only as long as ARIS messages are seen within a timeout period. *KeepAlive* messages are used to maintain state in the absence of the real traffic.

ARIS is also the first to introduce the term *label* and the concept of *stack of labels*.

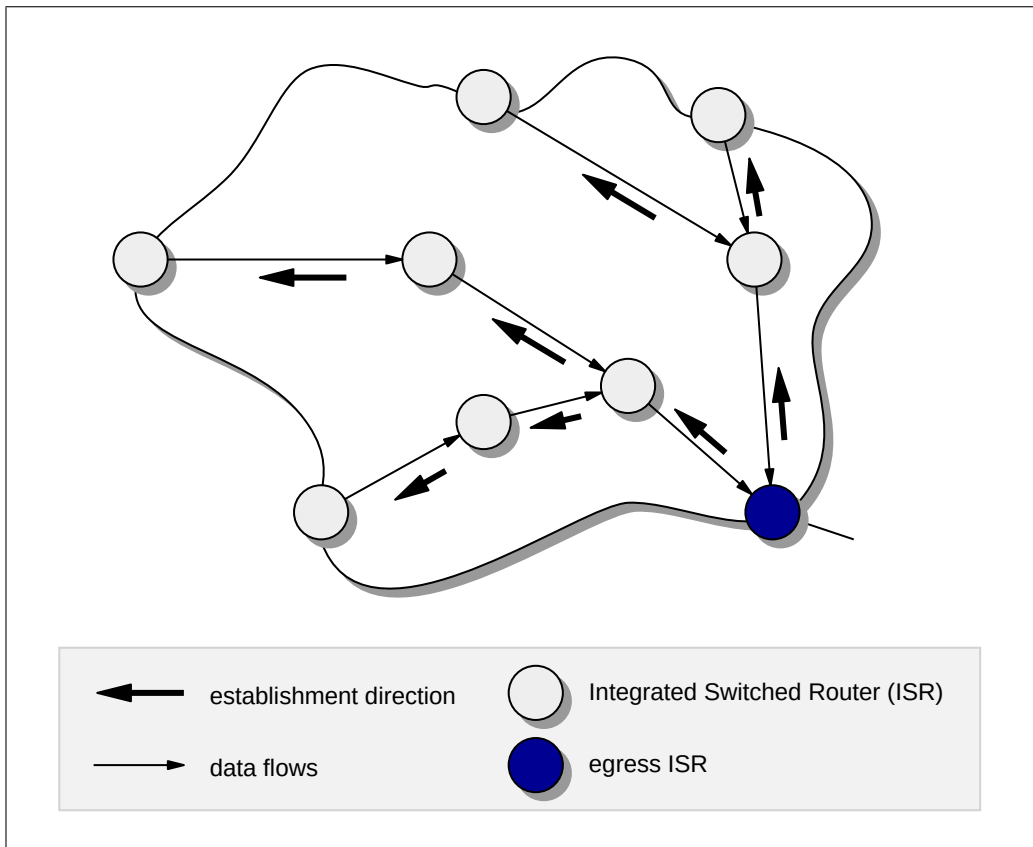


Figure 2.5: IBM ARIS Switched Paths

2.5.4 Cisco's Tag Switching

With its Tag Switching architecture, Cisco Systems is attempting to arrive at a generic approach to one of the traditional bottlenecks of IP routers, the *longest-prefix-match* lookup of a packet's destination address. This architecture is intended to be applicable across all switch points in a heterogeneous network, whether layer 3 routers or layer 2 switches. The architecture is outlined in [16].

As an upper-level protocol packet enters a tag switching capable “cloud”, it is given a *tag*, which is a unique label that is an index into a *Tag In-*

formation Base (TIB) residing in each Tag-Switching capable entry router. The *tag* functions are much like an ATM VPI/VCI header. It is possible for a specially designed tag switching interior router to implement a very fast, hardware-based, layer 2-like switching capability for those packets that carry these tags. A software upgrade to a conventional router's operating system confers some of the benefits of the quicker lookup, although without the enhanced hardware switching. The system is being designed to allow the use of *tags* across the ATM portion of heterogeneous networks; indeed, for ATM switches the tag is likely to be mapped directly to cell VPI/VCI fields. For conventional routers, the *tag* is embedded as an additional protocol element, either between the Network and Transport layers header, or within the Network Layer header. The TIB associates each *tag* entry with a new outgoing *tag*, as well as outgoing interface and link-level information. *Tags* are swapped at each switch point, as in native ATM. Routing information resides in a Forwarding Information Base (FIB), which is constructed using standard routing protocols (e.g. OSPF, BGP). Tag-Switching capable devices exchange FIB information using Tag Distribution Protocol (TDP).

Tag switches forward packets by simple label swapping, rather than by slower Network Layer forwarding as routers do. The tags may be somewhat more complex than ATM VPI/VCI headers, though, and actually there can be a stack of tags. This permits information hiding (and significant simplification) when routing across enclosed domains; by using tag switches as ingress/egress routers, only the border switch-routers need maintain exterior routing information. Switches within the domain need only know about interior routing, since packets being forwarded through the domain will have exterior routing information pushed onto the tag stack at the ingress switch and popped off

at the egress switch.

Tag Switching is very similar with ARIS in the sense that both approaches included proposals for signaling the values to be used by peers in implementing the switching paradigm, both relied on use of topology information from routing protocols to establish the paths to be used in packet switching and both had the concept of *stack of tags*. In addition to this, the tag-switching proposals added depth to the earlier CSR and IP-Switching proposals by providing alternatives in distribution of switching information.

While there had been an earlier attempt to establish a tag-switching forum, with the advent of TAG, ARIS and other proposals, it was clear that the possibility of developing a standard packet switching approach needed to be considered. Hence IETF working group was formed - which would later come to be called Multi-Protocol Label Switching (MPLS).

2.6 Multiprotocol Label Switching (MPLS)

Multiprotocol Label Switching was the name given by the IETF committee to a class of related technologies,. A common characteristic of those technologies is that the MPLS network is a transit network. End systems do not connect directly to MPLS switch-routers. Rather, the MPLS network moves data packets between ingress and egress points, from where conventional routers using legacy LAN techniques deliver them to the end systems. This is in contrast to such native ATM approaches as LANE and Classical IP over ATM, where ATM is used within the end system LAN itself. Originally, the MPLS technologies, result in independent standardization efforts. However,

a single Working Group was formed to deal with all of them in an attempt to ensure interoperability. The major technologies (and their primary industrial supporters) are:

- CSR - Cell Switched Router (Toshiba)
- IP Switching (Ipsilon)
- ARIS - Aggregate Route-based IP Switching (IBM)
- Tag Switching (Cisco Systems)

These approaches have certain similarities. The primary one is the attempt to achieve fast packet forwarding by using short, layer 2-like labels. However, while Tag Switching and ARIS attempt to remain independent of any lower-layer hardware implementation, IP Switching and CSR explicitly blend an ATM (layer 2) switching fabric with an IP (layer 3) routing and forwarding capability.

Since December 1996, the IETF² MPLS Working Group³ has been responsible for standardizing a base technology for using label switching and for the implementation of label-switched paths over various link-level technologies, such as Packet-over-Sonet, Frame Relay, ATM, and LAN technologies (e.g. all forms of Ethernet, Token Ring, etc.). This includes procedures and protocols for the distribution of labels between routers, encapsulations and multicast considerations. The primary goals of the working group have been largely completed. In particular, it has produced a number of Requests for

²*Internet Engineering Task Force* (IETF is an organization which develops standards for data transfer on the internet (see more at <http://www.ietf.org>)

³see <http://www.ietf.org/html.charters/mpls-charter.html>

Comments (RFCs)⁴ that define the base Label Distribution Protocol (LDP) [17, 18], the basic MPLS architecture [19] and encapsulations [20], the definitions for how MPLS runs over ATM [21] and Frame Relay [22] links, and the requirements for traffic engineering over MPLS [23].

As mentioned above, MPLS can run over many routed protocols such as: IP, ATM or Frame Relay. In this document, however we will refer only to IP as the routed.

2.6.1 Short overview of MPLS Architecture

The MPLS IETF workgroup defines in [19] the main elements of an MPLS system as follows:

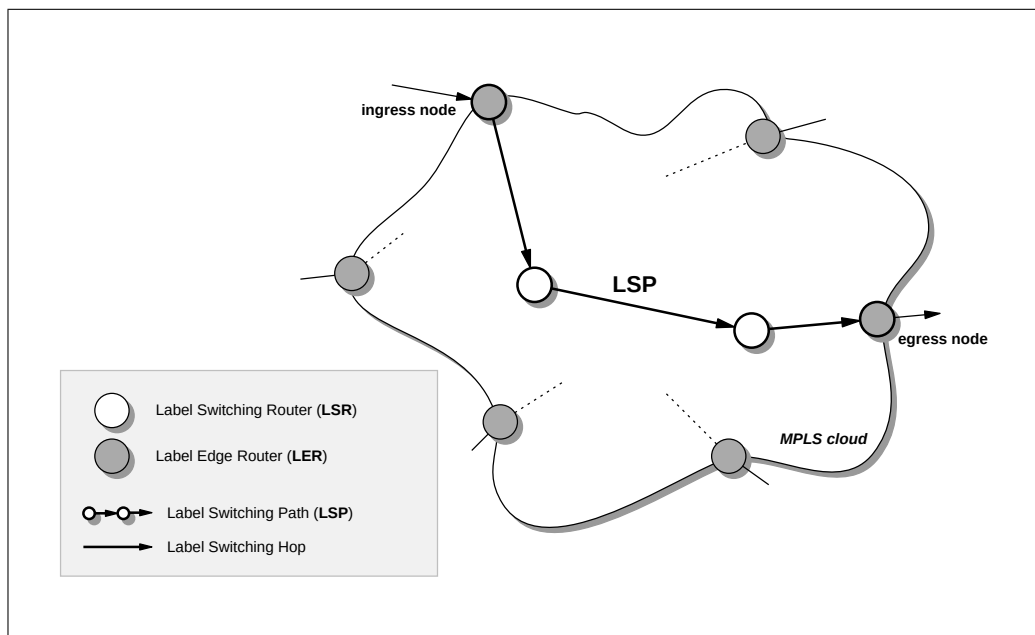


Figure 2.6: The MPLS cloud and its main components

⁴*RFCs* are IETF documents that contain proposals comments and standards related to Internet technology.

MPLS domain is a contiguous set of MPLS capable nodes which operate MPLS routing and forwarding and which are also in one Routing or Administrative Domain;

Label Switching Router (LSR) is an MPLS capable node, which 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, and may optionally be also capable of forwarding native L3 packets;

Label Edge Router (LER) is an MPLS capable 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 that is not running MPLS, that that LSR is an MPLS edge node;

Label Switched Path (LSP) is the path through one or more LSRs at one level of the hierarchy followed by packets in a particular Forwarding Equivalence Class;

label switched hop is the hop between two MPLS nodes, on which forwarding is done using labels;

ingress node is an MPLS capable edge node which handles traffic that enters an MPLS domain;

egress node is an MPLS capable edge node which handles traffic that leaves an MPLS domain.

As you can see in Figure 2.6, MPLS relies on the concepts of Label Switching Router (LSR), Label Switched Path (LSP), and labeled packets.

MPLS forwarding

Figure 2.7 describes the MPLS forwarding principles:

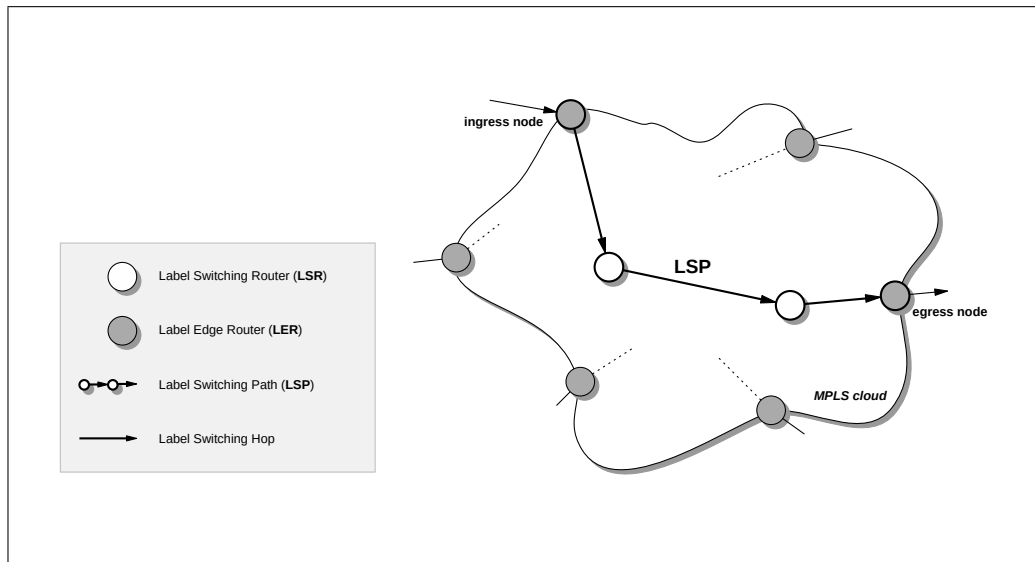


Figure 2.7: MPLS: Basic principles

A packet enters an MPLS cloud through an *Label Edge Router* (LER). The *LER* labels the packet and forwards it to the next *Label Switching Router* (LSR). Based on the label the *LSR* swaps the label and forwards the packet to the next *LSR*. The packet leaves the MPLS cloud passing through a *LER* which pops the label and forwards the packet to another legacy router or to the destination, in the traditional way.

We will describe further on the MPLS forwarding concepts in more detail.

The *Forwarding Equivalence Class*. As the packet enters the MPLS cloud, the *Label Edge Router* (LER) analyzes the packets network layer header and based on the information gathered, it assigns the packet to a

Forwarding Equivalence Clas. Packets which belong to the same *Forwarding Equivalence Class* are labeled with the same label and treated the same by the network.

The MPLS shim. When a LER labels a packet it places a “*shim*”, as you can see in Figure 2.8, between the header of layer 2 and the header of layer 3. This *shim* has 32 bits that are distributed amongst 4 fields as follows:

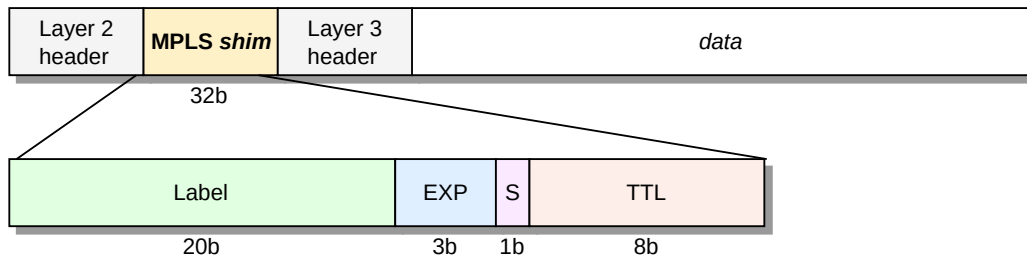


Figure 2.8: MPLS Label. MPLS shim

- ◇ **20b**- *Label*; the values from 0 to 15 are reserved.
- ◇ **3b** - *EXP*(*EXPerimental use*)
- ◇ **1b** - *S*(*bottom of Stack*) If the label is in the bottom of the stack, the value of this field will be 1 and 0 otherwise.
- ◇ **8b** - *TTL*(*Time To Live*)

However, because MPLS supports not only a single label, but a stack of labels, the MPLS *shim* is also referred in [20] as a *label stack entry*, and the label stack is defined as a sequence of label stack entries. More details about the MPLS label stack encoding can be found in [20].

Packet transitions. A packet suffers three types of modification when it passes an MPLS cloud:

1. In the *ingress node*, when the packet should be labeled, an MPLS shim is created and to the packet.
2. Traversing the *Label Switching Path*:
 - one or more labels can be popped out of the MPLS label stack
 - one or more labels can be pushed in the MPLS label stack
 - the current label can be swapped with another one
3. In the *egress node*, when the MPLS shim must be removed from packet and the forwarding decision to the next hop is made analyzing the packet's network header.

The MPLS Data Structures are tables which contain the information needed by a router in order to be able to receive a unlabeled packet, to label it and to forward it; or to receive a labeled packet and based on that label to be able to do the necessary modifications and to forward it. There are 3 types of such data structures:

1. The **Next Hop Label Forwarding Entry** (NHLFE) - assists to the adding of an outgoing label;
2. The **Incoming Label Map** (ILM) - helps to interpret the incoming labels;
3. The **“Forward Equivalence Class” to NHLFE** (FTN) - helps to decide what label to add to an unlabelled packet.

As you can see in Figure 2.9, when an unlabelled packet arrives to an *Label Edge Router* (LER), its network layer header is analyzed and the packet is assigned to a *Forwarding Equivalence Class*(FEC). The *FEC-to-NHLFE* (FTN) maps each FEC to a set of NHLFEs. If the set of NHLFEs mapped by FTN contains more than one element, only one NHLFE element must be chosen before the packet is forwarded. The NHLFE entry contains the packet's next hop and the operation to perform on the packet's label stack (*push* in this case). Then, the labeled packet is forwarded to the next hop (LSR).

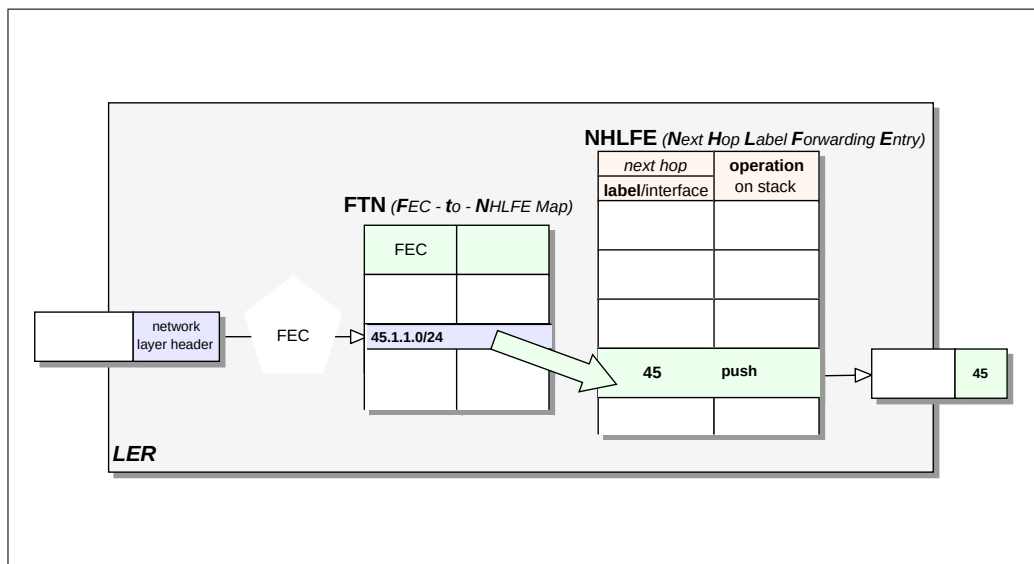


Figure 2.9: MPLS Data structures needed for an unlabelled packet

Figure 2.10 depicts the situation when a labeled packet arrives at an *Label Switching Router* (LSR). Using ILM, the incoming label is mapped to a set of NHLFEs. If more NHLFE entries correspond to the same label, only one of them should be chosen. The NHLFE entry contains the packet's next hop and the operation to perform on the packet's label stack. This operation may

be one of the following:

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

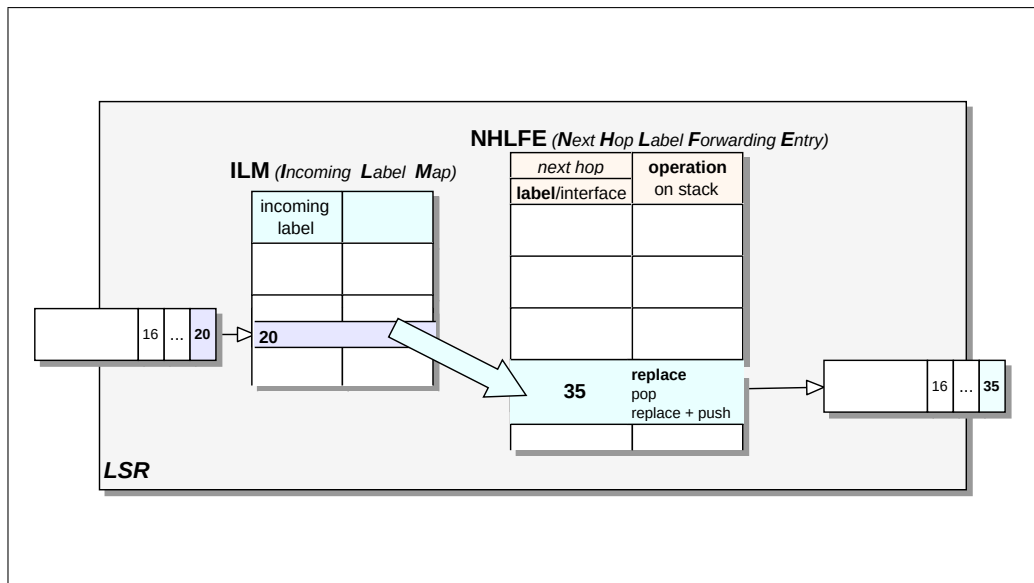


Figure 2.10: MPLS Data structures needed for an labelled packet

The MPLS label stack of the packet is modified by the operation mentioned in the NHLFE entry and the packet is forwarded to the next hop (LSR).

More details about MPLS and MPLS forwarding concepts could be found in [19, 24, 25].

MPLS signaling

A fundamental concept in MPLS is that two *Label Switching Routers* (LSRs) must agree on the meaning of the labels used to forward traffic between and through them. This common understanding is achieved by using a set of procedures, called a *label distribution protocol*, by which one LSR informs another of label bindings it has made. The request for comment [17] defines a set of such procedures called LDP (for Label Distribution Protocol) by which LSRs distribute labels to support MPLS forwarding along normally routed paths.

The standard defines some principles for this protocol:

1. The *Downstream* which can be:
 - “Downstream-on-Demand”, meaning that an explicit request has been made for a label binding
 - “Unsolicited Downstream”, which means that the bindings are distributed without any request
2. The *Label Retention Mode* can be:
 - “Liberal Label Retention Mode”, which keeps track of invalid bindings. It has the advantage of a quicker adaptation to routing changes.
 - “Conservative Label Retention Mode”, which discards the invalid bindings.
3. The *Scope and Uniqueness of Labels*:
 - “Per-Interface Label Space”, when labels are unique for one interface.

- “Per-Platform Label Space”, when labels are unique for the same LSR.

4. The *LSP control* can be:

- “Independent”, when each LSR makes its own decision to bind a label and to distribute it to its label distribution peers
- “Ordered”, when only the LSR which is the egress node or which has already received a binding label, binds a label.

The implementation of the protocol are many and different. We mention here only three of them: **LDP**, **CR-LDP** and **RSVP-TE**.

The *Label Distribution Protocol* (LDP) is based on TCP and is thus a *hard-state*⁵ protocol. LDP was primarily designed (based on the original Tag Distribution Protocol (TDP) and Aggregate Route-based IP Switching (ARIS) signaling proposal) as a mechanism to be used in setting up LSPs for hop-by-hop routing. For this reason, the simplest use of LDP is to establish single links of LSPs at a time.

The *Constraint-based Routing Label Distribution Protocol* (CR-LDP) provides extension to the base LDP for support of explicitly routing LSP setup requests and (potentially) reservin resources along the resulting along the resulting LSP. The capability of specifying an explicit route to be used in LSP setup permits the network operator or network management system to establish LSPs that are constraint by considerations other than

⁵a *hard-state* protocol is one in which the state information remains valid until explicit changed. Proper operation of hard-state protocol requires absolute reliability in delivery of message events because it must not be possible for events to be missed. Most protocols that are considered hard state are based on Transfer Control Protocol (TCP).

the necessity of following a routed path strictly. The ability to associate resources with such an LSP allows traffic to be channeled across the network routing infrastructure to provide *traffic engineering* or *virtual private networking* service.

The *Resource Reservation Protocol for Traffic Engineering* (RSVP-TE) provides extensions of the base RSVP for support of explicitly routed LS setup requests. These extensions are specifically defined to support LSP tunnels. RSVP itself defines mechanisms for support of allocation of network resources to the paths defined by protocol activity. RSVP-TE is a *soft-state*⁶ protocol.

Category	CR-LDP	RSVP-TE
Transport Mechanism	Transport on TCP (reliable)	Raw IP packets (unreliable)
State Management	Hard state	Soft state; needs per-flow refresh management
Messages required for LPS setup and maintenance	Request and mapping	Path, Resv, and ResvConf
Base Architecture	Based on LDP development for MPLS	Based on RSVP, but may require major changes to the basic protocol to improve its scalability.

Table 2.1: Signaling architectures of CR-LDP and RSVP-TE

⁶a *soft-state* protocol is one in which the failure to receive an update or refresh of state information causes the information to become out of date and be discarded. It can operate fairly well in an environment in which delivery of update or refresh message events is not reliable. The Resource Reservation Protocol (RSVP) is often referred to as a soft-state protocol.

A comparison between the signaling architectures of CR-LDP and RSVP-TE and between the signaling support for traffic engineering in CR-LDP and RSVP-TE was made in [26]. Table 2.2 contains the differences between the signaling architectures of the two protocols, while table ?? shows their different capabilities in supporting traffic engineering:

2.6.2 The role of MPLS in the Internet Architecture

Currently, the two most important uses of MPLS are traffic engineering and virtual private networks (VPNs). Although both can and are performed currently using existing standard protocols, MPLS makes these tasks simpler because of the advantage of separation of routing and forwarding. Therefore, some routing limitations are reduced or eliminated.

We will refer in the remaining of this document only to traffic engineering (Chapter 3) and to the advantages and capabilities of MPLS for traffic engineering (Section 3.3).

Category	CR-LDP	RSVP-TE
Signaling of Qos and traffic parameters	DifServ and ATM traffic classes	Extendable; currently based on IntServ traffic classes
Types of CR-LSPs	Strict, loose, and loose pinned	Strict and loose; no pinning
Models of label distribution and LSP	Easy to support all modes since CR-LDP is based on LDP	Only downstream on demand; need to run both RSVP and LDP for other modes
Path preemption	Supported	Supported
Failure notification	Reliable procedure	Unreliable procedure
Failure recovery	Global and local repair	Global and local repair; local repair done using fast-reroute which requires precomputing alternate paths at every node
Loop detection/prevention	LPD employs Path Vector TVL to prevent Label Request messages from looping. Hop Count TVL is used to find looping LSPs.	May be done using the Record Route object
Path optimization and rerouting	LSP ID can be used to prevent double booking of bandwidth for an LSP when doing "make-before-break"	Shared explicit filter prevents double booking of bandwidth for an LSP when doing "make-before-break"

40
Table 2.2: Signaling support for traffic engineering features in CR-LDP and RSVP-TE

Chapter 3

Internet Traffic Engineering

Internet traffic engineering is defined in [27] as that aspect of Internet network engineering dealing with the issue of performance evaluation and performance optimization of operational IP networks. Traffic Engineering encompasses the application of technology and scientific principles to the measurement, characterization, modeling, and control of Internet traffic [23, 28].

3.1 Traffic Engineering overview

This section is a briefly review of different traffic engineering approaches proposed and implemented in telecommunications and computer networks.

3.1.1 Traffic Engineering in Classical Telephone Networks

The early telephone network relied on static hierarchical routing. Therefore, routing patterns remained fixed independent of the state of the network or time of day. The hierarchy was intended to accommodate overflow traffic, improve network reliability via alternate routes, and prevent call looping by em-

ploying strict hierarchical rules. The network was typically over-provisioned since a given fixed route had to be dimensioned so that it could carry user traffic during a busy hour of any busy day. Hierarchical routing in the telephony network was found to be too rigid upon the advent of digital switches and stored program control, which were able to manage more complicated traffic engineering rules.

Dynamic routing was introduced to alleviate the routing inflexibility in the static hierarchical routing so that the network would operate more efficiently. This resulted in significant gains such as: reducing the overall loss probability by 10 to 20 percent and improving network resilience by recalculating routes on a per-call basis and periodically updating routes.

There are three main types of dynamic routing in the telephone network:

time-dependent routing - regular variations in traffic loads (such as time of day or day of week) are exploited in pre-planned routing tables;

state-dependent routing (SDR) - routing tables are updated online according to the current state of the network (e.g., traffic demand, utilization, etc.);

event dependent routing (EDR) - routing changes are incepted by events (such as call setups encountering congested or blocked links) whereupon new paths are searched out using learning models

Dynamic non-hierarchical routing (DNHR) is an example of dynamic routing that was introduced in the AT&T network in the 1980's to respond to time-dependent information such as regular load variations as a function of time.

3.1.2 Evolution of Traffic Engineering in Packet Networks

In terms of traffic management, the Internet has been a best effort service environment until recently. In particular, very limited traffic management capabilities existed in IP networks to provide differentiated queue management and scheduling services to packets belonging to different classes.

In terms of routing control, the Internet has employed distributed protocols for intra-domain routing. These protocols are highly scalable and resilient. However, they are based on simple algorithms for path selection which have very limited functionality to allow flexible control of the path selection process.

Adaptive Routing in the ARPANET

The early ARPANET recognized the importance of adaptive routing where routing decisions were based on the current state of the network. Therefore, each node maintained a table of network delays, representing the estimated delay that a packet would experience along a given path toward its destination. The minimum delay table was periodically transmitted by a node to its neighbors. The shortest path, in terms of hop count, was also propagated to give the connectivity information.

One drawback to this approach is that congestion will be shifted from one location of a network to another location, resulting in oscillation and network instability.

Dynamic Routing in the Internet

The Internet evolved from the ARPANET and adopted dynamic routing algorithms with distributed control to determine the paths that packets should take en-route to their destinations. The routing algorithms are adaptations of shortest path algorithms where costs are based on link metrics. The link metric can be based on static (local criteria) or dynamic (e.g. delay, packet loss) quantities.

It was apparent early that static link metric assignment was inadequate because it can easily lead to unfavorable scenarios in which some links become congested while others remain lightly loaded. One of the many reasons for the inadequacy of static link metrics is that link metric assignment was often done without considering the traffic matrix in the network. Also, the routing protocols did not take traffic attributes and capacity constraints into account when making routing decisions. This results in traffic concentration being localized in subsets of the network infrastructure and potentially causing congestion.

The inadequacy of the legacy Internet interior gateway routing system is one of the factors motivating the interest in path oriented technology with explicit routing and constraint-based routing capability such as MPLS.

Type-of-Service (ToS) routing

Type-of-Service (ToS) routing involves different routes going to the same destination with selection dependent upon the ToS field of an IP packet [29]. Each link is associated with multiple link costs and each link cost is used to compute routes for a particular ToS. A separate shortest path

tree is computed for each ToS. The shortest path algorithm must be run for each ToS resulting in very expensive computation. Classical ToS-based routing is outdated as the IP header field has been replaced by a Diffserv field. Effective traffic engineering is difficult to perform in classical ToS-based routing because each class still relies exclusively on shortest path routing which results in localization of traffic concentration within the network.

Equal Cost Multi-Path (ECMP)

ECMP is another technique that attempts to address the deficiency in the Shortest Path First (SPF) interior gateway routing systems [5]. In the classical SPF algorithm, if two or more shortest paths exist to a given destination, the algorithm will choose one of them. The algorithm is modified slightly in ECMP so that if two or more equal cost shortest paths exist between two nodes, the traffic between the nodes is distributed among the multiple equal-cost paths. Traffic distribution across the equal-cost paths is usually performed in one of two ways:

packet-based in a round-robin fashion;

flow-based using hashing on source and destination IP addresses and possibly other fields of the IP header

In ECMP, link costs are static and bandwidth constraints are not considered, so ECMP attempts to distribute the traffic as equally as possible among the equal-cost paths independent of the congestion status of each path. As a result, given two equal-cost paths, it is possible that one of the paths will be more congested than the other. Another drawback of ECMP is that load sharing cannot be achieved on multiple paths which have non-identical costs.

Nimrod

is a routing system developed to provide heterogeneous service specific routing in the Internet, while taking multiple constraints into account [30]. Essentially, Nimrod is a link state routing protocol which supports path oriented packet forwarding. It uses the concept of maps to represent network connectivity and services at multiple levels of abstraction. Mechanisms are provided to allow restriction of the distribution of routing information.

Even though Nimrod did not enjoy deployment in the public Internet, a number of key concepts incorporated into the Nimrod architecture, such as explicit routing which allows selection of paths at originating nodes, are beginning to find applications in some recent constraint-based routing initiatives.

3.1.3 The Overlay Model

In the overlay model, a virtual-circuit network, such as ATM or frame relay, provides virtual-circuit connectivity between routers that are located at the edges of a virtual-circuit cloud. Thus, two routers that are connected through a virtual circuit see a direct adjacency between themselves independent of the physical route taken by the virtual circuit through the ATM or frame relay network. By decoupling the logical topology that routers see from the physical topology that the ATM or frame relay network manages, the overlay model enables a network administrator or an automaton to employ traffic engineering concepts to perform path optimization by re-configuring or rearranging the virtual circuits so that a virtual circuit on a congested or sub-optimal physical link can be re-routed to a less congested or more

optimal one.

The overlay model using IP over ATM requires the management of two separate networks with different technologies (IP and ATM) resulting in increased operational complexity and cost (see Section 2.4.1).

3.1.4 Constrained-Based Routing

Constraint-based routing refers to a class of routing systems that compute routes through a network subject to the satisfaction of a set of constraints and requirements.

The concept of constraint-based routing within the context of MPLS traffic engineering requirements in IP networks was first defined in [23].

Unlike QoS routing (see [1]) which generally addresses the issue of routing individual traffic flows to satisfy prescribed flow based QoS requirements subject to network resource availability, constraint-based routing is applicable to traffic aggregates as well as flows and may be subject to a wide variety of constraints which may include policy restrictions.

3.1.5 Other IETF Projects Related to Traffic Engineering

The Internet Engineering Task Force (IETF) is a large open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet. The actual technical work of the IETF is done in its

working groups, which are organized by topic into several areas (e.g., routing, transport, security, etc.). Much of the work is handled via mailing lists.

The main IETF projects related to Traffic Engineering that are going to be mentioned in this document are:

- *Integrated Services*
- *RSVP*
- *Differentiated Services*
- *MPLS*

Integrated Services

The IETF Integrated Services working group developed the integrated services (Intserv) model. This model requires resources, such as bandwidth and buffers, to be reserved a priori for a given traffic flow to ensure that the quality of service requested by the traffic flow is satisfied. The integrated services model includes additional components beyond those used in the best-effort model such as:

packet classifiers - used to identify flows that are to receive a certain level of service;

packet schedulers - handles the scheduling of service to different packet flows to ensure that QoS commitments are met;

admission control - used to determine whether a router has the necessary resources to accept a new flow.

Two services have been defined under the Integrated Services model: *guaranteed service* [31] and *controlled-load service* [32].

The **guaranteed service** can be used for applications requiring bounded packet delivery time. For this type of application, data that is delivered to the application after a pre-defined amount of time has elapsed is usually considered worthless. Therefore, *guaranteed service* was intended to provide a firm quantitative bound on the end-to-end packet delay for a flow. This is accomplished by controlling the queuing delay on network elements along the data flow path. The guaranteed service model does not, however, provide bounds on jitter (inter-arrival times between consecutive packets).

The **controlled-load service** can be used for adaptive applications that can tolerate some delay but are sensitive to traffic overload conditions. This type of application typically functions satisfactorily when the network is lightly loaded but its performance degrades significantly when the network is heavily loaded. *Controlled-load service*, therefore, has been designed to provide approximately the same service as best-effort service in a lightly loaded network regardless of actual network conditions. *Controlled-load service* is described qualitatively in that no target values of delay or loss are specified.

The main issue with the Integrated Services model has been scalability, especially in large public IP networks which may potentially have millions of active micro-flows in transit concurrently.

A notable feature of the Integrated Services model is that it requires explicit signaling of QoS requirements from end systems to routers. The Resource

Reservation Protocol (RSVP) was initially thought to perform this signaling function and is a critical component of the Integrated Services model. The RSVP protocol is described next.

Resource ReSerVation Protocol (RSVP)

RSVP is a soft state signaling protocol [33]. It supports receiver initiated establishment of resource reservations for both multicast and unicast flows. RSVP was originally developed as a signaling protocol within the integrated services framework for applications to communicate QoS requirements to the network and for the network to reserve relevant resources to satisfy the QoS requirements.

RSVP functioning principles are that the sender or source node sends a PATH message to the receiver with the same source and destination addresses as the traffic which the sender will generate. Every intermediate router along the path forwards the PATH Message to the next hop determined by the routing protocol. After receiving a PATH Message, the receiver responds with a RESV message which includes a flow descriptor used to request resource reservations. The RESV message travels to the sender or source node in the opposite direction along the path that the PATH message traversed. Every intermediate router along the path can reject or accept the reservation request of the RESV message. If the request is rejected, the rejecting router will send an error message to the receiver and the signaling process will terminate. If the request is accepted, link bandwidth and buffer space are allocated for the flow and the related flow state information is installed in the router.

One of the issues with the original RSVP specification was Scalability. This is because reservations were required for micro-flows, so that the amount of state maintained by network elements tends to increase linearly with the number of micro-flows (see [34]).

Recently, RSVP has been modified and extended in several ways to mitigate the scaling problems. As a result, it is becoming a versatile signaling protocol for the Internet. For example, RSVP has been extended to reserve resources for aggregation of flows, to set up MPLS explicit label switched paths, and to perform other signaling functions within the Internet. There are also a number of proposals to reduce the amount of refresh messages required to maintain established RSVP sessions [34].

Differentiated Services

The goal of the Differentiated Services (Diffserv) effort within the IETF is to devise scalable mechanisms for categorization of traffic into behavior aggregates, which ultimately allows each behavior aggregate to be treated differently, especially when there is a shortage of resources such as link bandwidth and buffer space [35]. One of the primary motivations for the Diffserv effort was to devise alternative mechanisms for service differentiation in the Internet that mitigate the scalability issues encountered with the Intserv model.

The IETF Diffserv working group has defined a Differentiated Services field in the IP header (DS field). The DS field consists of six bits of the part of the IP header formerly known as TOS octet. The DS field is used to indicate the forwarding treatment that a packet should receive at a node [29]. The Diffserv working group has also standardized a number of Per-Hop Behavior

(PHB) groups. Using the PHBs, several classes of services can be defined using different classification, policing, shaping, and scheduling rules.

For an end-user of network services to receive Differentiated Services from its Internet Service Provider (ISP), it may be necessary for the user to have a Service Level Agreement (SLA) with the ISP. An SLA may explicitly or implicitly specify a Traffic Conditioning Agreement (TCA) which defines classifier rules as well as metering, marking, discarding, and shaping rules.

Packets are classified, and possibly policed and shaped at the ingress to a Diffserv network. When a packet traverses the boundary between different Diffserv domains, the DS field of the packet may be re-marked according to existing agreements between the domains.

Differentiated Services allows only a finite number of service classes to be indicated by the DS field. The main advantage of the Diffserv approach relative to the Intserv model is scalability. Resources are allocated on a per-class basis and the amount of state information is proportional to the number of classes rather than to the number of application flows.

Multiprotocol Label Switching

MPLS is a very powerful technology for Internet traffic engineering¹ because it supports explicit LSPs which allow constraint-based routing to be implemented efficiently in IP networks [28]. The requirements for traffic engineering over MPLS are described in [23]. Extensions to RSVP to support instantiation of explicit LSP are discussed in [36]. Extensions to LDP, known

¹For more details on Traffic Engineering and MPLS see 3.3.

as CR-LDP, to support explicit LSPs are presented in [37].

3.2 Traffic Engineering concepts

3.2.1 Traffic Engineering Performance Objectives

The key performance objectives associated with traffic engineering are [23]:

traffic oriented - include the aspects that enhance the QoS of traffic streams

(minimization of packet loss, minimization of delay, maximization of throughput, and enforcement of service level agreements);

resource oriented - include the aspects referring to the optimization of

resource utilization *(minimizing congestion and load balancing)*.

3.2.2 Traffic Engineering Process Model

The process model describes the aspects of Internet traffic engineering in an operational context. It is described in [23] as a sequence of actions that a traffic engineering system must perform to optimize the performance of an operational network.

The traffic engineering process model is iterative [28] and has four phases that repeat continually.

1. **Define the control policies.**
2. **Feedback mechanism.** Involves the acquisition of measurement data from the operational network.
3. **Analyze the network state and characterize traffic workload.**
Performance analysis may be:

- *proactive* - identifies potential problems that do not exist, but could manifest in the future;
- *reactive* - identifies existing problems, determines their cause through diagnosis, and evaluates alternative approaches to remedy the problem, if necessary.

A traffic matrix may be constructed as part of the analysis process.

4. **Performance optimization.** Involves a decision process which selects and implements a set of actions from a set of alternatives. Optimization actions may include the use of appropriate techniques to either control the offered traffic or to control the distribution of traffic across the network. Optimization actions may also involve adding additional links or increasing link capacity, deploying additional hardware such as routers and switches, systematically adjusting parameters associated with routing such as IGP metrics and BGP attributes, and adjusting traffic management parameters.

3.2.3 Traffic Engineering Taxonomy

In this section we will present various classifications of traffic engineering systems:

Time-dependent vs State-dependent vs Event-dependent

- *Time-dependent* algorithms - use historical information based on periodic variations in traffic, (such as time of day) to pre-program routing plans and other TE control mechanisms. Time-dependent algorithms are suitable for predictable traffic variations. An example of a time-dependent algorithm is a global centralized optimizer where the input

to the system is a traffic matrix and multi-class QoS requirements as described [38].

- *State-dependent* algorithms - adapt the routing plans for packets based on the current state of the network (e.g. utilization, packet delay, packet loss, etc.). Constraint-based routing is an example of state-dependent TE operating in a relatively long time scale. An example operating in a relatively short time scale is a load-balancing algorithm described in [39].
- *Event-dependent* algorithms - use learning models to find good paths for TE in a network.

Offline vs Online

- *Offline* performed computation of routing plans can be done for scenarios where routing plans need not be executed in real-time (e.g. forecast information, extensive searches on multi-dimensional solution spaces).
- *Online* performed computation is required when the routing plans must adapt to changing network conditions as in state-dependent algorithms. Unlike offline computation (which can be computationally demanding), online computation is geared toward relative simple and fast calculations to select routes, fine-tune the allocations of resources, and perform load balancing.

Centralized vs Distributed

- *Centralized* control has a central authority which determines routing plans and perhaps other TE control parameters on behalf of each router. The central authority collects the network-state information

from all routers periodically and returns the routing information to the routers. The routing update cycle is a critical parameter directly impacting the performance of the network being controlled. Centralized control may need high processing power and high bandwidth control channels.

- *Distributed* control determines route selection by each router autonomously based on the routers view of the state of the network. The network state information may be obtained by the router using a probing method or distributed by other routers on a periodic basis using link state advertisements.

Local vs Global

- *Local* network-state information belongs to the state of a portion of the domain. Examples include the bandwidth and packet loss rate of a particular path. Local state information may be sufficient for certain instances of distributed-controlled TEs.
- *Global* network-state information belongs to the state of the entire domain undergoing traffic engineering. Examples include a global traffic matrix and loading information on each link throughout the domain of interest. Global state information is typically required with centralized control. Distributed TE systems may also need global information in some cases.

Prescriptive vs Descriptive

- *Prescriptive* traffic engineering systems evaluate alternatives and recommends a course of action. It can be:

- *corrective* - prescribes a course of action to address an existing or predicted anomaly.
- *perfective* - prescribes a course of action to evolve and improve network performance even when no anomalies are evident.
- *Descriptive* traffic engineering systems characterize the state of the network and assesses the impact of various policies without recommending any particular course of action.

Open Loop vs Closed Loop

- *Open Loop* traffic engineering control is where control action does not use feedback information from the current network state. The control action may use its own local information for accounting purposes, however.
- *Closed Loop* traffic engineering control is where control action utilizes feedback information from the network state. The feedback information may be in the form of historical information or current measurement.

Tactical vs Strategic

- *Tactical* traffic engineering aims to address specific performance problems (such as hot-spots) that occur in the network from a tactical perspective, without consideration of overall strategic imperatives. Without proper planning and insights, tactical TE tends to be ad hoc in nature.
- *Strategic* traffic engineering approaches the TE problem from a more organized and systematic perspective, taking into consideration the immediate and longer term consequences of specific policies and actions.

3.3 Traffic Engineering and MPLS

3.3.1 Concepts

The attractiveness of MPLS for Traffic Engineering can be attributed to the following factors [23]:

1. *explicit label switched paths* which are not constrained by the destination based forwarding paradigm can be easily created through manual administrative action or through automated action by the underlying protocols;
2. *LSPs* can potentially be *efficiently maintained*;
3. *traffic trunks* can be instantiated and *mapped onto LSPs*;
4. a set of *attributes* can be *associated with traffic trunks* which modulate their behavioral characteristics;
5. a set of *attributes* can be *associated with resources* which constrain the placement of LSPs and traffic trunks across them;
6. MPLS allows for both *traffic aggregation and disaggregation* whereas classical destination only based IP forwarding permits only aggregation;
7. it is relatively easy to integrate a “*constraint-based routing*” framework with MPLS;
8. a good implementation of MPLS can offer significantly *lower overhead* than competing alternatives for Traffic Engineering.

Having this long list of MPLS capabilities, the most important ones required in order to fully support Traffic Engineering over MPLS in large networks

are: the *traffic trunk* and its *attributes* (4), the *resource attributes* (5) and the “*constraint-based routing*” framework (7).

The Traffic Trunk

According to Li and Rekhter [40], a traffic trunk is an aggregation of traffic flows of the same class which are placed inside a Label Switched Path. A traffic trunk can have characteristics associated with it (addresses, port numbers) and can be routed because it is an aspect of the LSP and, therefore, the path through which the traffic trunk flows can be changed.

Traffic Trunk properties The basic properties of traffic trunks are:

- A traffic trunk is an *aggregate* of traffic flows belonging to the same class.
- In a single class service model, such as the current Internet, a traffic trunk could encapsulate all of the traffic between an ingress LSR and an egress LSR, or subsets thereof.
- Traffic trunks are routable objects (similar to ATM VCs).
- A traffic trunk is distinct from the LSP through which it traverses. In operational contexts, a traffic trunk can be moved from one path onto another.
- A traffic trunk is unidirectional.

Operations on Traffic Trunks The basic operations on traffic trunks significant to Traffic Engineering purposes are listed in Table 3.1.

Action	Description
Establish	Create an instance of a traffic trunk.
Activate	Begin passing traffic via this traffic trunk instance
Deactivate	Cease passing traffic via this traffic trunk instance
Modify	Change attributes of this traffic trunk instance
Reroute	Redetermine the path used by this traffic trunk instance
Destroy	Remove a traffic trunk instance and free associated resources

Table 3.1: Traffic Trunk Operations

Attributes of Traffic Trunks An attribute of a traffic trunk is a parameter assigned to it which influences its behavioral characteristics [23].

Attributes can be explicitly assigned to traffic trunks through administration action or they can be implicitly assigned by the underlying protocols when packets are classified and mapped into equivalence classes at the ingress to an MPLS domain. Regardless of how the attributes were originally assigned, for Traffic Engineering purposes, it should be possible to administratively modify such attributes.

The basic attributes of traffic trunks particularly significant for Traffic Engineering are listed in Table 3.2. Traffic trunks attributes are described in much more detail in [23].

Resource Attributes

Resource attributes are part of the topology state parameters, which are used to constrain the routing of traffic trunks through specific resources.

Attribute	Description
Traffic parameter	indicates the resource requirements of the traffic trunk: (<i>average and peak rates, burst tolerance, etc.</i>)
Generic Path selection and maintenance	define the rules for: - <i>selecting</i> the route taken by a traffic trunk - <i>maintenance</i> of paths that are already established
Priority	defines the relative importance of traffic trunks
Preemption	defines the relative preemption between traffic trunks
Resilience	determines the behavior of a traffic trunk under fault conditions
Policing	determines the actions that should be taken by the underlying protocols when a traffic trunk becomes non-compliant

Table 3.2: Traffic Trunk Attributes

Maximum Allocation Multiplier (MAM) is an administratively configurable attribute which determines the proportion of the resource (e.g. bandwidth) that is available for allocation to traffic trunks.

Resource Class Attributes can be viewed as "colors" assigned to resources such that the set of resources (e.g. links) with the same "color" conceptually belong to the same class. Resource class attributes can be used to implement a variety of policies or for identification purposes.

Constraint-Based Routing

In contemporary terminology, constraint-based routing is often referred to as "QoS Routing" see [1].

A constraint-based routing framework uses the *traffic trunk attributes*, *resource attributes* and other topology state information in order to automatically compute on each node explicit route for each traffic trunk originating

from that node.

3.3.2 Current state-of-the-art

There are many document related to traffic engineering in MPLS. We will present during this subsection part of those articles and their proposals for achieving traffic engineering in MPLS.

General concepts of Traffic Engineering in MPLS

The most important document referring to MPLS Traffic Engineering is [28]. It defines the main concepts² of MPLS Traffic Engineering which are then standardized in [23].

Another article [26] discusses the direction taken by IETF and some other standardization efforts for traffic engineering using MPLS. The primary focus of this paper is on the signaling protocols (CR-LDP and RSVP-TE) developed for this purpose. A comparison between the two signaling protocols is also presented in the paper.

MPLS advantages for traffic engineering are depicted in [41]. The specific MPLS architectural features discussed are: the separation of control and forwarding, the label stack, multiple control planes, and integrated IP constrained-based routing. The article also discusses how these features address network scalability, simplify network service integration, offer integrated recovery, and simplify network management.

Authors in [42] describe phases in designing an MPLS system for traffic engi-

²these concepts are also described in Section 3.3

neering in an Internet Service Provider's (ISP)³ network. They also discuss how to provide Quality of Service (QoS) in a network with MPLS. Therefore, this article offer readers the practical issues of traffic engineering and a working solution for traffic engineering with MPLS in the Internet.

Problems of traffic engineering in MPLS based IP networks are mathematically formulated in [43]. The problems discussed are: constraint based routing, connection admission control, rerouting and capacity panning. The authors say that obtaining the optimal solution of traffic engineering problems has undesirable computational complexity, and they show that the problems are NP-complete⁴. Therefore the problems can not be solved with the current tools.

Tools and algorithms for MPLS Traffic Engineering

Routing And Traffic Engineering Server (RATES) is a software system developed for MPLS Traffic engineering. Along with the description of RATES, [44] also describes some new routing ideas incorporated in RATES for MPLS explicit path selection. The RATES implementation consists of a policy and flow database, a browser based interface for policy definition and entering resource provisioning requests, and a COPS (Common Open Policy Service) server-client implementation for communicating paths and resource information to edge routers. RATES also uses the OSPF topology database for dynamically obtaining link-state information. RATES can set-up bandwidth guaranteed label switched paths (LSPs) between specified ingress-egress pairs. The path selection for LSPs is based on a new minimum-

³The ISP mentioned in this paper is *GlobalCenter*.

⁴optimal solutions can not be found by any known polynomial time algorithm and hence the computational complexity increases greatly with the size of the problem

interference routing algorithm aimed at making the best use of network infrastructure in an on-line environment where LSP requests arrive one-by-one with no a priori information about future requests. Though developed for an MPLS application, the RATES implementation has many similarities in components to an intra-domain differentiated services bandwidth broker.

Minimum Interference Routing Algorithm (MIRA) described in [45] is a new algorithm for dynamic routing of bandwidth guaranteed tunnels where tunnel routing requests arrive one-by-one and there is no a priori knowledge regarding future requests.

MPLS Adaptive Traffic Engineering (MATE) is described in [39]. The main goal of MATE is to avoid network congestion by adaptively balancing the load among multiple paths based on measurement and analysis of path congestion. MATE adopts a minimalist approach in that intermediate nodes are not required to perform traffic engineering or measurements besides normal packet forwarding. Moreover, MATE does not impose any particular scheduling, buffer management, or a priori traffic characterization on the nodes. The paper also presents an analytical model, derives a class of MATE algorithms, and proves their convergence. Several practical design techniques to implement MATE are described also in the paper.

PTPlan is a tool for MPLS traffic engineering described in [46]. This tool allows for multi-hour dimensioning of networks supporting simultaneously peer-to-peer and client-server services. It is assumed that all LSPs to be routed and their attributes are known. The dimensioning problem is a combined capacity design and routing problem where the LSP sets are calculated in order to minimize the network operational costs. PTPlan includes

a graphical interface for an easy introduction and edition of the network parameters.

Traffic and Reliability Engineering Server (TRES) is a server deployed by authors from [47]. TRES calculates and sets wide bandwidth paths while satisfying many policy constraints. It also provides path rerouting in the event of a link or a node failure.

DiffServ, Intserv and MPLS

There are many articles which address the integration of *Integrated Services* and *Differentiated Services* with MPLS:

- Authors from [48] propose DRUM (Diffserv and RSVP/intserv Use of MPLS), an architecture that delivers end-to-end service guarantees for both Diffserv and Intserv networks, where part of the underlying technology used for IP transport is MPLS using Diffserv-like mechanisms for QoS provision.
- [49] describes how MPLS can be used to prioritize traffic and provision network bandwidth in order to offer integrated voice/data services, differentiated services and service level agreement.

Even more articles are referring to the integration of Differentiated Services with MPLS.

- Authors in [50] describe various approaches which can be used to support differentiated services in MPLS environments.
- [51] proposes an approach to Traffic Engineering that uses Differentiated Services (diffserv) and MPLS to provide quantitative QoS guarantees over an IP network. An algorithm that determines QoS-constrained

routes is proposed and a framework that uses such an algorithm is outlined. This framework removes the responsibility of QoS guarantees from the core nodes, thereby reducing their complexity.

- Authors from [52] propose a new MPLS traffic engineering scheme based on a constraint-based routing algorithm, which enhances the original E-LSP with per-class traffic engineering and load balancing.
- [53] proposes a model and architectural framework for supporting DiffServ-based end-to-end QoS in the Internet, assuming underlying MPLS-based explicit routed paths. The proposed integrated management and control architecture will allow providers to offer both quantitative and qualitative based services while optimizing the use of underlying network resources.

Other MPLS traffic engineering related issues such as constraint-based routing, dynamic traffic partitioning or congestion were presented in some documents, but they are out of the scope of this document.

Our insight to MPLS traffic engineering with respect to bandwidth management is described in [54].

Chapter 4

Active Networks

Traditionally, the function of a network has been to deliver packets from one endpoint to another and any processing within the network was limited basically to routing, congestion control and quality of service (QoS) schemes. This kind of network can be regarded as “passive”. The problems with “passive” networks are mentioned in [55, 56]:

- the difficulty of integrating new technologies and standards into the shared network infrastructure;
- the reduced performance due to redundant operations at several protocol layers;
- the difficulty to accommodate new services in the existing architectural model.

New applications which required computation within the network that have emerged, such as firewalls, Web proxies, multicast routers and mobile proxies, have adopted a variety of ad hoc services for performing user-driven computations at nodes within the network.

A innovative idea was proposed as a solution to this problems: to give the user the possibility to program the network. This technology is called *Active Networks* [56, 57, 55].

The “active” networks are active in two ways [56]:

1. routers and switches within the network can perform computations on user data flowing through them;
2. and users can “program” the network, by supplying their own programs to perform these computations.

4.1 Active Networks Overview

The concept of active networking emerged from discussions within the broad DARPA research community in 1994 and 1995 on the future directions of networking systems.

In an *active network*, the routers or switches of the network perform customized computations on the messages flowing through them or even modify their content. Figure 4.1 illustrates how the routers of an IP network could be augmented to perform such customized processing on the datagrams flowing through them. These active routers could also interoperate with legacy routers, which transparently forward datagrams in the traditional manner.

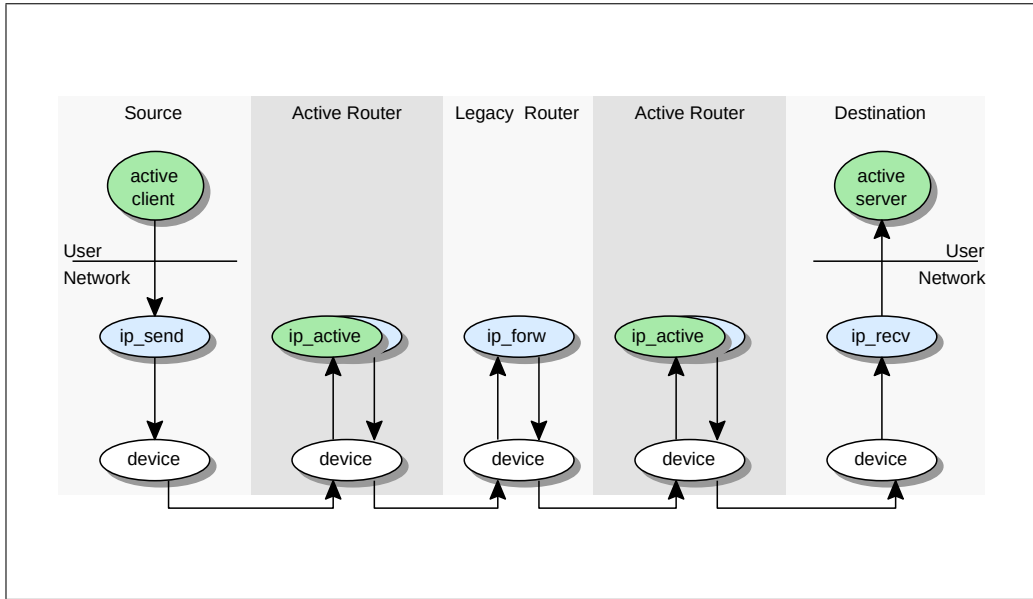


Figure 4.1: Application-specific processing within the nodes of an Active Network

4.1.1 Architectures

There are three architecture approaches for implementing active networks. They differ by the placement of the code in the network.

Active nodes approach

The authors of [56, 57] refer to this as the *discrete model* or *Programmable switches* approach because the programs and data are carried separately (i.e., are discrete), while [55] refers to it as the *active node approach*.

In this approach, the packets carry some identifiers or references to pre-defined functions that reside in the active nodes. The packets are active in the sense that they decide which functions are going to be executed on their data, but the actual code resides in the active node.

Examples of “active nodes” architectures are:

An architecture for Active Networking proposed at Georgia Institute of Technology. In this architecture users control the invocation of pre-defined network-based functions through control information in packet headers [58]. Users can select from the available set of functions to be computed on their data and supply parameters as input to those computations. The available functions are chosen and implemented by the network service provider, and support-specific services. Thus, users are able to influence the computation of a selected function but cannot define arbitrary functions to be computed.

DAN Architecture. The packets in *Distributed Code Caching for Active Networks (DAN)* architecture [59] contain a finite sequence of function identifiers, and parameters for the functions. The functions are daisy-chained in the sense that one function calls the next according to the order of the identifiers in the packet. Depending on the type of node that the packet is processed upon and the packet’s content, only a subset of the functions may be called. If the node is unable to locate a function, it temporarily suspends the processing of the packet and calls a “code server” for the implementation of the function. The code server is a well known node in the network which provides a library of functions for different types of operating systems from various developers. Once the module is downloaded, it is permanently stored locally on the node in order to prevent more downloads of the same module. The option of downloading modules differentiates this technology from the previous one.

ANTS , the *Active Network Transport System* [60], is a Java-based active network toolkit where arbitrary new protocols are automatically deployed at both intermediate nodes and end systems by using mobile code¹ techniques. The network is viewed as a distributed programming system. The architecture introduces three components schemes: capsules, active nodes, and code distribution. Capsules are replacements for a packet. Their function is to include reference to the routine to be used to process the capsule at each active node.

Active packets approach

The Active packets approach, referred to as *the integrated approach* or *capsules* in [56, 57] is characterized by the fact that the code is carried by the packet. The nodes are also active because they allow computations up to the application layer to take place, but no active code resides in them. This is the reason why [55] refers to this approach as *active packets*.

Being carried by the packet the code within an active node cannot be very large, but only composed of “primitive” instructions, that perform basic computations on the packet’s content.

Examples of “active packets” architectures are:

Smart Packets project was proposed at BBN Technologies [61]. In an attempt to provide a rich and flexible programmable environment with-

¹A mobile software agent is a program that acts on behalf of a user or another program, and its capable of moving within the network under its own control. The agent chooses when and to where it will migrate, and may interrupt its own execution and continue elsewhere on the network.

out overloading the computer power of the managed node and without making an environment so complex that it is difficult to be secure, two decisions were made:

1. programs must be self contained and must fit entirely into one packet (cannot be more than 1 Kbyte long);
2. the operating environment must provide safety and security because packets containing executable code are extremely dangerous.

Active IP Option [62] describes an extension to the IP options mechanism that supports the embedding of program fragments in datagrams and the evaluation of these fragments as they traverse the network. Legacy passive packets are replaced by active capsules, which are miniature programs that are executed as they travel. These capsules can invoke predefined primitives that interact with the local node environment, and leave information behind in a node that they have visited.

M0 Architecture. The messenger in the M0 system is similar to the capsule or the smart packet. Messengers are programs exchanged between M0 nodes. There are four elements inside the M0 node:

1. concurrent messenger threads;
2. a shared memory area;
3. a simple synchronization mechanism;
4. channels toward neighboring nodes.

Messenger code is written in the M0 language. M0 is a high level language that inherits from PostScript the main concepts of operand, dictionary, execution stack, and the main data manipulation and flow control operators. The M0 interpreter is written in C.

Active packets and nodes approach

The third approach is a combination of the previous two, where active packets carry the actual code and other more complex code resides in active nodes. Using this approach, users can choose either the active packets approach or the active nodes approach according to the nature of their application.

Examples of “active nodes and packets” architectures are:

SwitchWare Architecture [63] uses a layered architecture to provide a range of different flexibility, safety and security, performance, and usability tradeoffs. The three layers defined in SwitchWare are:

1. *Active Packets* - realizes the active packets approach;
2. *Switchlets* - realizes the active nodes approach;
3. *Active Router Infrastructure*.

In SwitchWare, active packets carry programs consisting of both code and data, and replace both the header and the payload of conventional packets. The programming language used is PLAN (*Programming Language for Active Networks*) [64].

NetScript Architecture. The NetScript Project [65] wants to program networks efficiently. Therefore, it provides:

- an architecture for programming networks;
- an architecture of a dynamically programmable network device/node;
- a language called *NetScript* for building network software on a programmable network.

NetScript uses delegated agents to program and control the functions of intermediate network devices/nodes.

A short comparison between the three approaches

The active nodes approach has good performance because security issues are much less than in the previous approach. However, the flexibility of the relevant architectures is limited. In an effort to increase flexibility, DAN and ANTS architectures have adopted a scheme where code is downloaded on demand and is cached for future use. As a result, these two technologies can easily deploy any new arbitrary protocol. Nevertheless, downloading code on demand causes some delay that reduces the overall performance.

The active packets approach suffers from performance related problems because safety and security requirements are huge. In an effort to reduce the security burden and thus increase performance, some researchers have decided to restrict the functionality of the programs carried by the active packets, resulting in architectures with decreased active packets, resulting in architectures with decreased packet approach that can provide arbitrary functionality, thanks to its novel caching technique.

The combination of both approaches seems to be very appealing. SwitchWare architecture realizes this idea by the use of a layered architecture, and manages to provide a range of different flexibility, safety and security, performance, and usability tradeoffs. Finally, NetScript architecture proposes a novel viewpoint where the network is treated as a single programmable abstraction.

4.1.2 Applications

The most important applications of active networks stems directly for their ability to program the network: new protocols and innovative cost-effective technologies can be easily employed at intermediate nodes.

Efforts to improve performance using active networking span a wide variety of areas. Amongst the most important areas are:

Network Management

The conventional approach to network management is to poll managed devices from a management station, requesting the values of variables and checking for anomalies. This approach concentrates the intelligence in the management stations, resulting in processing and communication bottlenecks. Further, the pollandcheck approach severely limits the ability to track problems in a timely and efficient manner. Several projects are considering the use of active networking to improve network management. The Smart Packets project [61] at BBN is developing architecture, languages, and protocols for making managed nodes programmable. The Netscript project [65] at Columbia is developing techniques to automatically create systematic management instrumentation and respective MIBs, using the structure of active elements.

Congestion Control

Network conditions such as the presence of congestion or a lossy link can significantly degrade the quality of application streams. Schemes that require the sender to adapt to network conditions have wellknown limitations, including the time for the sender to detect the condition, react, and transmit

the adapted data to the receiver. During the adaptation interval, the receiver will experience uncontrolled losses (when conditions worsen) or less than optimal performance (when conditions improve). By moving information about how to adapt to network conditions into network nodes, the appropriate type of adaptation can occur when and where it is needed.

Efforts in this area have included transparent inline protocol “boosters” to adapt to network conditions (e.g., by adding forward error correction over errorprone links) and intelligent discard strategies for preserving the quality of MPEG video in the face of network congestion [58]. An interesting combination of multicast and MPEG video distribution is examined later in this section.

Multicasting

The “traditional” IP multicast service hides from its users the details of the routing topology and the number and location of receivers. For unreliable multicast, this approach makes sense and allows scaling to larger applications; however, there are inherent problems in using this model when it is desired to deliver data to all receivers reliably. The difficulty arises in recovering from losses, which typically affect all receivers downstream in the multicast tree from the point of a loss. A common approach is to spread the responsibility for multicast retransmissions among the receivers, to avoid overburdening the sender. For good performance, this requires that receivers be aware of nearby receivers, that are above the loss point, to provide retransmission.

By including state and processing in the network, retransmissions can be directed to nearby receivers or can come from caches at network nodes.

Both approaches reduce the delay and transmission resources required for retransmission.

Caching

A substantial fraction of all network traffic today comes from applications in which clients retrieve objects from servers (e.g. the WorldWide Web). The caching of objects in locations “close” to clients is an important technique for reducing both network traffic and response time for such applications. Caching schemes require decisions about where to locate objects and how to forward requests between caches. Current widearea caches are manually configured into a static hierarchy, thus incurring an administrative burden and limiting the ability to react to dynamic conditions.

Efforts to use active networking for caching include using network mechanisms to route cache requests to preconfigured cache locations, and combining small caches with information about the contents of nearby caches, at each network node. The Adaptive Web Caching project is developing an architecture for an adaptive, scalable web caching system, with mechanisms that could be realized using active networking or applicationlayer protocols.

Chapter 5

Active Networks and MPLS

5.1 Overview

There are some attempts to integrate Active Networks and MPLS.

Active Routing permits individual customers, network managers, or network owners to control the paths that their data takes through the network. They propose to use active networks in order to extend the label switching to:

1. provide more flexible access to LSP's;
2. perform a wider range of operations on stacks of labels;
3. implement new rules for constructing stacks of labels;
4. enable a user to establish his own label switching paths;
5. use more flexible restoration techniques;
6. remove the need to increase the address space in IPv4.

Our approach [66] proposed an architecture to integrate Active Networks and MPLS.

5.2 Our Approach

5.2.1 Motivation

The desire to provide the QoS capabilities of MPLS end-to-end motivates its deployment not only in the core network but in the access network as well. Thus, many now advocate the use of MPLS networks in the access area.

However, a disadvantage of MPLS is that the forwarding process is opaque in the sense that the network is insensitive to the packets it carries and they are transferred between MPLS nodes without modification. In the access area there are situations where such modifications are required.

In today's networks, there are two ways of implementing such computations within the network:

In the traditional ("passive") way, the processing within the network is limited primarily to routing, congestion control and QoS schemes and the points that perform such computations are fixed elements in the network. Such a network poses difficulties in integrating new technologies and standards into the shared network infrastructure, in reducing redundant operations at several protocol layers and in accommodating new services in the existing architectural model.

Another solution is using *Active Networks*.

5.2.2 Description

Given the arguments in the previous section we propose the integration of MPLS with Active Networks as a possible solution for the future access networks.

The routers of an IP network can be augmented to have active capabilities. These active routers can coexist and interoperate with legacy IP routers, which transparently forward packets in a traditional manner.

In the same way, as you can see in Figure 5.1, we propose a network in which some LSRs are augmented with active capabilities. Legacy LSRs and active ones can coexist and interoperate.

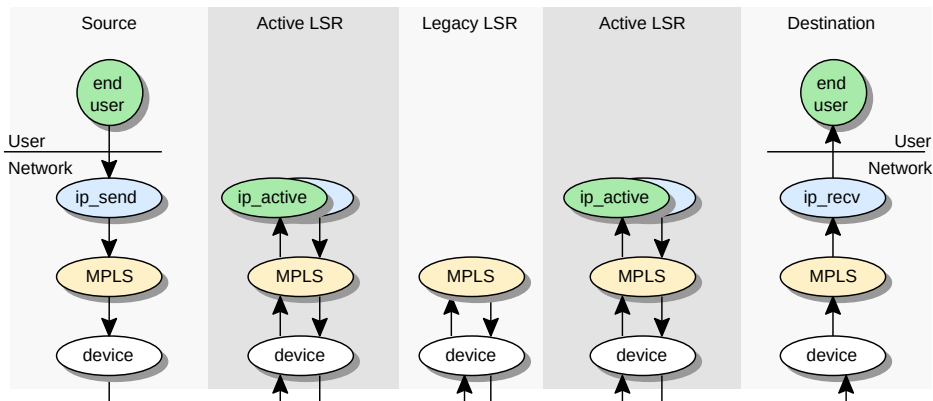


Figure 5.1: Packet processing within the nodes of an MPLS Active Network

A more detailed architecture of an active LSR is shown in Figure 5.2. A packet is determined to be an “active” packet or not based on the label. If the packet is “active”, it is sent to the IP layer and from there to the relevant

active code in order to be processed. After being processed, the packet is sent back to MPLS as if it were generated locally. Then, the packet is assigned to a Forwarding Equivalence Class and labelled corresponding with its class.

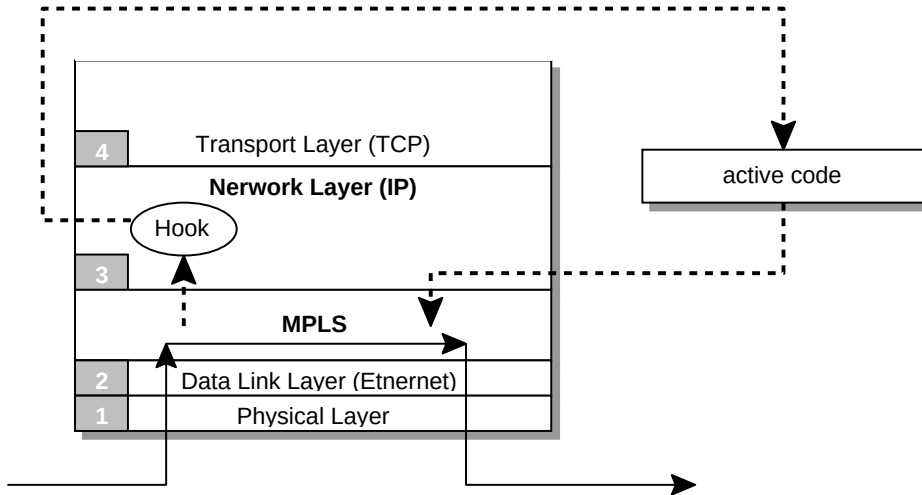


Figure 5.2: An active MPLS architecture

Using this architecture only the active packets (as identified by the label) that traverse an active LSR will be processed by the active code. Conventional packets will be processed in accordance with the standard MPLS protocol.

5.3 Our Implementation

We used for our implementation the *Linux Operating System*. We simulate a network using *User-Mode Linux*, and we configured the LSR's using *Netfilter*. We will describe these concepts in the following sections.

5.3.1 The prerequisite

Linux Operating System

Linux is a free Unix-type operating system originally created by Linus Torvalds with the assistance of developers around the world. Developed under the GNU General Public License¹, the source code for Linux is freely available to everyone.

Due to the very nature of Linux's functionality and availability, it has become quite popular worldwide and a vast number of software programmers have taken Linux's source code and adapted it to meet their individual needs. At this time, there are dozens of ongoing projects for porting Linux to various hardware configurations and purposes.

There is also an mpls-linux project, which is a open source effort to create a set of MPLS signalling protocols and an MPLS forwarding plane for the Linux operating system. We used this MPLS implementation for our purposes. The description and code sources of mpls-linux implementation can be found at: <http://sourceforge.net/projects/mps-linux/>.

User-Mode Linux

User-Mode Linux [67] is a safe, secure way of running Linux versions and Linux processes.

User-Mode Linux gives you a virtual machine that may have more hardware and software virtual resources than your actual, physical computer.

¹see: <http://www.gnu.org/copyleft/gpl.html>

Disk storage for the virtual machine is entirely contained inside a single file on your physical machine. You can assign your virtual machine only the hardware access you want it to have. With properly limited access, nothing you do on the virtual machine can change or damage your real computer, or its software.

Usually, the Linux Kernel talks straight to your hardware (video card, keyboard, hard drives, etc), and any programs which run ask the kernel to operate the hardware as you can see in Figure 5.3(a).

The User Mode Linux Kernel is different. Instead of talking to the hard-

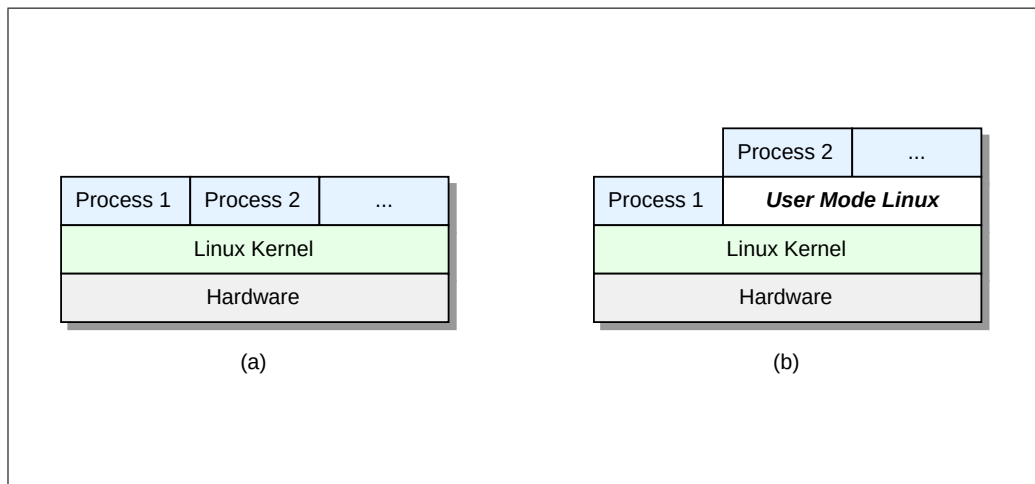


Figure 5.3: (a)The Linux Kernel (b)The User-Mode Linux Kernel

ware, it talks to the “real” Linux kernel (called the “host kernel”), like any other program. Programs can then run inside User-Mode Linux as if they were running under a normal kernel, as you can see in Figure 5.3(b).

To install² it you only need a Linux kernel, an UML patch and a filesystem.

We used UML in order to simulate different “virtual” Linux boxes on a single “real” one. Setting each separate “virtual” Linux box and interconnecting them and connecting them with the “real” Linux box we obtained a small network on which we could perform our simulations.

Netfilter

Netfilter [68, 69, 70] is a part of the networking software within the Linux kernel. This framework offers the possibility of “packet mangling” (i.e. modification of the header or payload contents). For each networking protocol netfilter implements “hooks” that are well defined points in a packet’s traversal of that protocol stack. In IPv4 there are 5 such “hooks” defined as illustrated in Figure 5.4:

NF_IP_PRE_ROUTING : placed just after entering IPv4 and before any route module;

NF_IP_LOCAL_IN : placed after the point where the routing module decides that the packet is local and before exiting IPv4 for an upper (transport) layer;

NF_IP_FORWARD : placed after the point where the routing module decides that the packet should be forwarded;

NF_IP_POST_ROUTING : placed after the point where the routing module decides where to forward the packet;

²See more detail about installing at <http://user-mode-linux.sourceforge.net/>

NF_IP_LOCAL_OUT : placed just after IPv4 entry point from an upper (transport) layer and before any route module

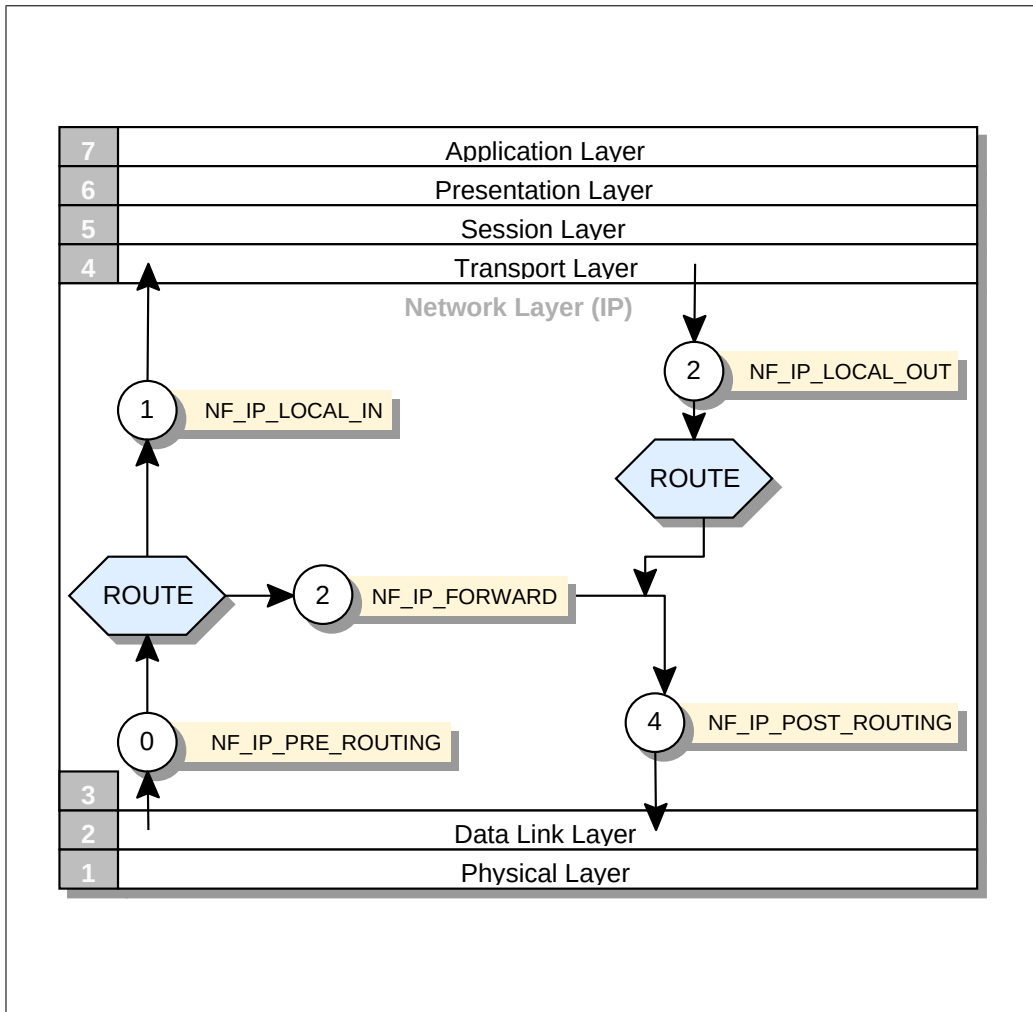


Figure 5.4: The Netfilter “hooks” defined in IPv4

As showed in Figure 5.5, parts of the kernel (modules) can register to listen to the different hooks for each protocol. If someone has registered for a particular protocol and hook, the packets passing that point in the protocol stack can be directed to the netfilter framework, where they can

be dropped (NF_DROP), accepted (NF_ACCEPT) or queued to userspace (NF_QUEUE).

Queued packets are sent to userspace, where a userspace process can examine the packet, can alter it, and reinject it at the same hook from which it left the kernel.

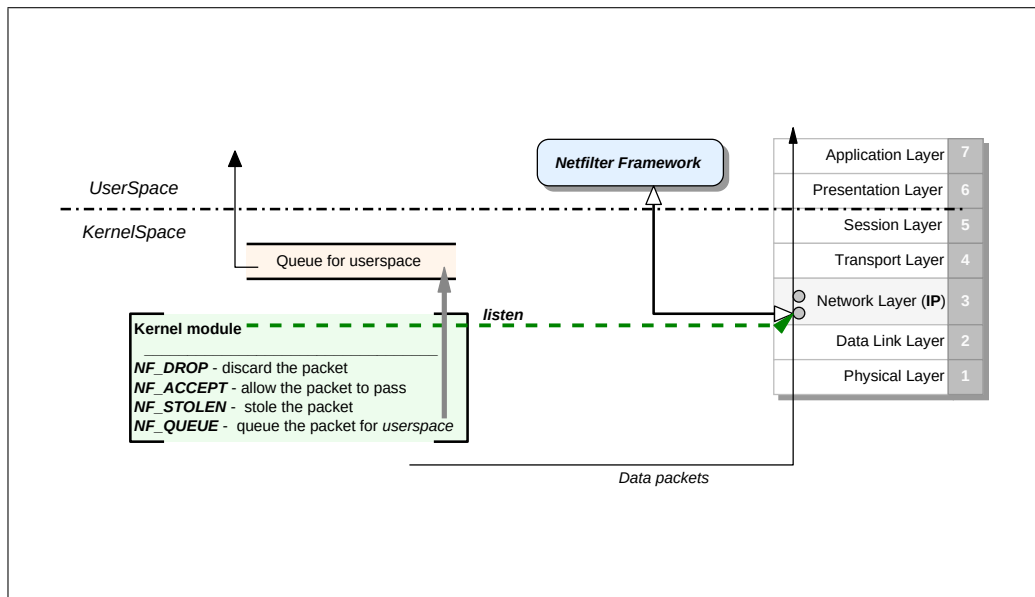


Figure 5.5: The Netfilter routine

The packets could also be processed directly in the kernel. However, the workload of the kernel would be increased, which would adversely affect the performance of the operating system. Also, packets arriving after the active packet would be delayed while it was being processed. By processing the active packet in userspace, conventional packets that succeed it encounter no such delay, and the active packet is re-injected asynchronous (if appropriate) into the packet stream after processing.

5.3.2 Implementation

To prove the concept of integrating MPLS and active networks, we set up a minimal MPLS network using the `mpls-linux` implementation as shown in Figure 5.6. **LSR A** to **LSR C** through **LSR B**.

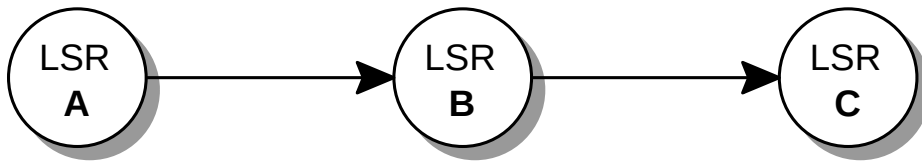


Figure 5.6: A minimal MPLS network

We set labels so that active packets that travel from **LSR A** to **LSR C**, arriving at **LSR B** are identified by a certain label and consequently they will be sent up to the IP layer. Then, using the netfilter framework, we implemented an trivial example of modifying (“mangling”) the packet.

Using the netfilter framework, we registered to listen the first hook defined by IPv4 (`NF_IP_PRE_ROUTING`) which is placed at the entry of the packet in the protocol stack, just after the sanity checks (i.e., not truncate, IP checksum OK, not promiscuous receive). We captured packets passing that hook and queued them for userspace. We wrote an userspace application which modifies the packet’s source address. We tested the system as you can see in Figure 5.8, by sending ICMP (Internet Control Message Protocol) messages from **LSR A** to **LSR C**.

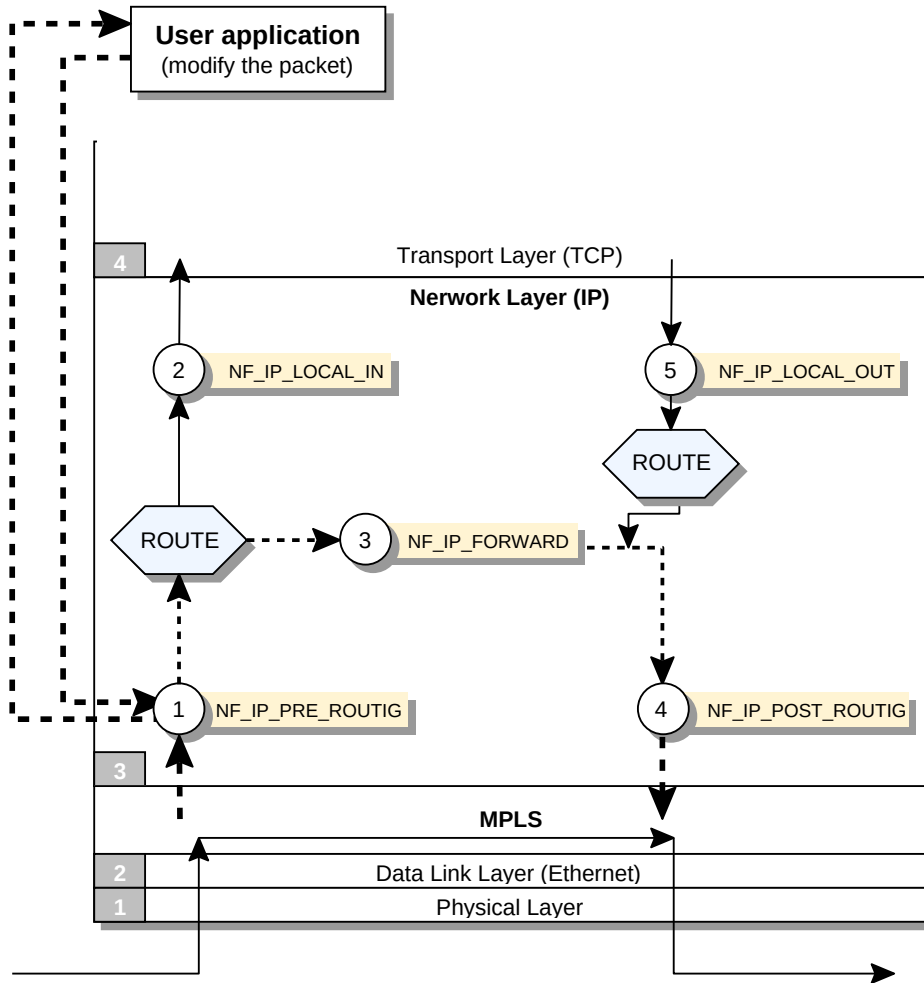


Figure 5.7: A MPLS node modified to programme the packets

A response ICMP message should be received by **LSR A** as a reply for each ICMP message sent, but because packets from **LSR A** destined to **LSR C** went through **LSR B**, their source address was changed with the address of **LSR D**. Therefore, **LSR C** received the ICMP messages but saw that the packets came from another destination (as the destination was altered) and sent the reply to that destination, which was **LSR D**.

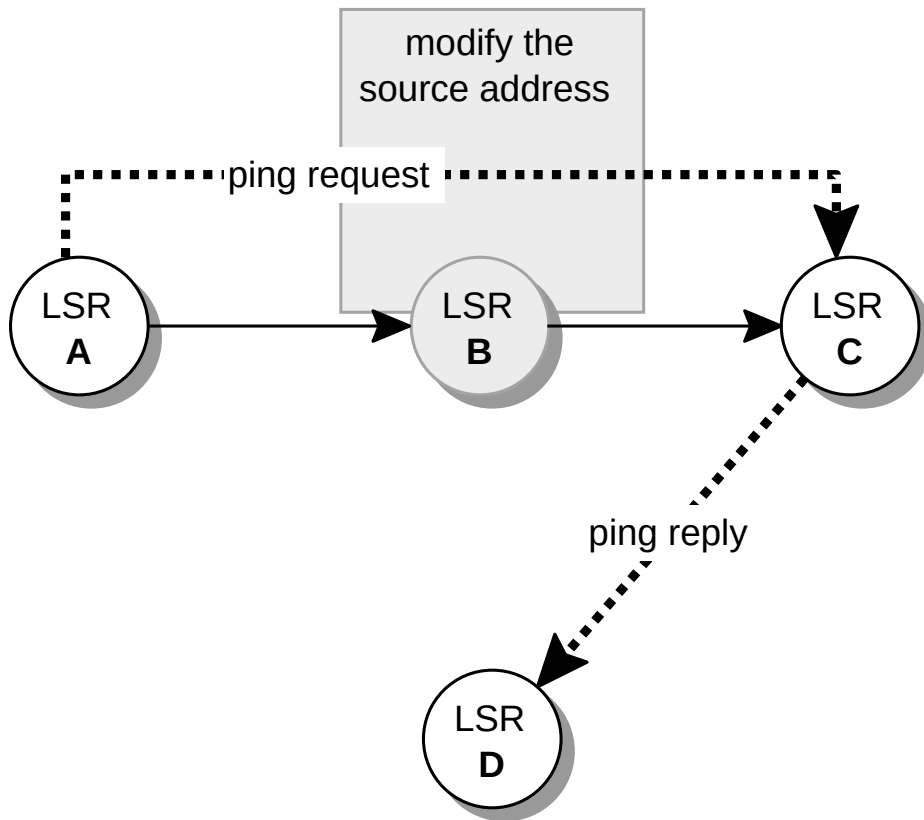


Figure 5.8: Our experiment test

Although this implementation it is only a “*proof of concept*”, it wants to simulate an active MPLS network in which we considered **LSR B** to be an “active” LSR, therefore it should sent the packets received with a specific label to IP protocol stack. From IP stack we captured the packet and we sent it to an application that modifies the packet (e.g the source address). The modified packet was then labelled and sent to the next hop.

In a legacy LSR a transit packet will go up to the MPLS layer where, based

on the label, the packet is forwarded to the next hop.

In our proposed active LSR, packets arriving with a certain label which specifies that they are “active” are sent up to network layer (IP) and from there to an active code.

This experiment demonstrates the principle used to provide active packet support in MPLS networks. The benefits of providing such support depend on the capabilities of the active code and its interaction with the MPLS software. We are currently developing an interface between the linux-mpls implementation and our active code modules, which will enable active packets to be used to fine-tune the operation of MPLS so as to emulate the features of multi-layer switching technologies. This features can be used in solving problems such as MPLS based Web switching load balancing [71].

5.4 Possible applications

There is a wide variety of applications which can emerge from integrating Active Networks and MPLS. These applications can be grouped in:

- applications that modify the packet content;
- applications that change a packet’s path;

5.4.1 Applications that modify the packet content

There are not many possible applications to modify the content of a packet to think of, and those modifications can be only modifications in the IP or TCP layer header.

One class of such application can be when a packet should pass several different MPLS clouds separated by IP networks. By changing the destination address at the exit of one MPLS cloud, the packet could be directed (even in the IP network) to choose to pass one specific MPLS cloud. At the entry of that particular MPLS cloud, the IP header modification will be restored.

5.4.2 Applications that change a packet's path

Such applications can also be grouped in:

- Fusion applications, where fewer packet forwarded than received;
- Fission applications, where more packet forwarded than received;
- Caching by delegating of application functionality to internal network nodes.

Fusion applications

This kind of applications are also called *filters* because packets arriving at that specific router are either dropped or merged. An example for merging the content of the packets is the case of receiving information from more sensors, information that should be merged eventually. This kind of merging can be done also inside the network.

Fission applications

This is the case of multicasting, where a single set of packets will be sent to a certain router from where copies of that packet will be sent to each destination.

Delegating application functionality

This is the case where computations are delegated in the network instead of the end nodes. One example could be the online auctions described in [72], where the rejection of lower bids is done before the auction server. Another examples could be the *mixing of sensor data* and the *multicast*, as the computation is done in advance or delayed as much as possible.

Chapter 6

Conclusions

Because, today, the Internet has become a convenient and cost effective medium for collaboration, education, electronic commerce, and entertainment, and tends to become a medium for convergence of voice, video, and data communication, it needs new technologies to respond to such challenges.

As MPLS and Active Networks tend to coexist in the access networks in order to support the new demands such as traffic engineering and end-to-end quality of service by using MPLS and the need to provide a much more flexible infrastructure, with increased capabilities by using Active Networks.

A integration of those two protocols was proposed (see *Chapter 5*). This integration overcomes a significant limitation of MPLS in access networks, namely its inability to perform switching above layer two. We have implemented a prototype network using Linux, which proves the validity of the concept. We have developed a framework which proves that such an integration it is possible.

6.1 Future work

As a future work we are considering to implement an application that takes all the advantages from the integration of Active Networks and MPLS and by that extends their capabilities.

Although choosing such an application is not an easy task, we think that using active networks in order to unify intra-domain MPLS capabilities for traffic engineering or end-to-end quality of service might be a possible candidate.

Bibliography

- [1] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick. A framework for qos-based routing in the internet. Technical Report RFC2386, IETF, August 1998.
- [2] M. Laubach. Classical ip and arp over atm. Technical Report RFC1577, IETF, January 1994.
- [3] M. Laubach and J. Halpern. Classical ip and arp over atm. Technical Report RFC2225, IETF, April 1998.
- [4] ATM Forum. Private network-network interface specification , version 1.0. Technical Report af-pnni-0055.000, ATM Forum, March 1996.
- [5] J. Moy. Ospf version 2. Technical Report RFC2328, IETF, April 1998.
- [6] Y. Rekhter and T. Li. A border gateway protocol 4 (bgp-4). Technical Report RFC1771, IETF, March 1995.
- [7] J. Luciani, D. Katz, D. Piscitello, B. Cole, and N. Doraswamy. Nbma next hop resolution protocol (nhrp). Technical Report RFC2332, IETF, April 1998.
- [8] ATM Forum. Lan emulation over atm version 1.0. Technical Report af-lane-0021.000, ATM Forum, January 1995.

- [9] ATM Forum. Multi-protocol over atm version 1.0. Technical Report af-mpoa-0087.000, ATM Forum, July 1997.
- [10] Y. Katsube, K. Nagami, and H. Esaki. Toshiba's router architecture extensions for atm : Overview. Technical Report RFC2098, IETF, February 1997.
- [11] T. Bradley, C. Brown, and A. Malis. Inverse address resolution protocol. Technical Report RFC2390, IETF, September 1998.
- [12] K. Nagami, Y. Katsube, Y. Shobatake, A. Mogi, S. Matsuzawa, T. Jinmei, and H. Esaki. Toshiba's flow attribute notification protocol (fanp) specification. Technical Report RFC2129, IETF, April 1997.
- [13] Peter Newman, Greg Minshall, and Thomas L.Lyon. Ip switching: Atm under ip. *IEEE/ACM Transactions on Networking*, 6(2):117–129, April 1998.
- [14] P. Newman, W. L. Edwards, R. Hinden, E. Hoffman, F. Ching Liaw, T. Lyon, and G. Minshall. Ipsilon's general switch management protocol specification version 1.1. Technical Report RFC1987, IETF, August 1996.
- [15] P. Newman, W. L. Edwards, R. Hinden, E. Hoffman, F. Ching Liaw, T. Lyon, and G. Minshall. Ipsilon flow management protocol specification for ipv4 version 1.0. Technical Report RFC1953, IETF, May 1996.
- [16] Y. Rekhter, B. Davie, D. Katz, E. Rosen, and G. Swallow. Cisco systems' tag switching architecture overview. Technical Report RFC2105, IETF, February 1997.

- [17] L. Andersson, P. Doolan, N. Feldman, A. Fredette, and B. Thomas. Ldp specification. Technical Report RFC3036, IETF, January 2001.
- [18] B. Thomas, Cisco Systems, E. Gray, and Zaffire. Ldp applicability. Technical Report RFC3037, IETF, January 2001.
- [19] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching architecture. Technical Report RFC3031, IETF, January 2001.
- [20] E. Rosen, D. Tappan, G. Fedorkow, Y. Rekhter D. Farinacci, T. Li, and A. Conta. Mpls label stack encoding. Technical Report RFC3032, IETF, January 2001.
- [21] B. Davie, J. Lawrence, K. McCloghrie, E. Rosen, G. Swallow, Y. Rekhter, and P. Doolan. Mpls using ldp and atm vc switching. Technical Report RFC3035, IETF, January 2001.
- [22] A. Conta, P. Doolan, and A. Malis. Use of label switching on frame relay networks specification. Technical Report RFC3034, IETF, January 2001.
- [23] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus. Requirements for traffic engineering over mpls. Technical Report RFC2702, IETF, September 1999.
- [24] Uyless Black. *MPLS and Label Switching Networks*. Prentice Hall, 2001.
- [25] Eric W. Gray. *MPLS. Implementing the technology*. Addison-Wesley, 2001.
- [26] Anoop Ghanwani, Bilel Jamoussi, Don Fedyk, Peter Ashwood-Smith, Li Li, and Nancy Feldman. Traffic engineering standards in ip networks using mpls. *IEEE Communication Magazine*, 37(12):49–53, December 1999.

- [27] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao. Overview and principles of internet traffic engineering. Technical Report RFC3272, IETF, May 2002.
- [28] Daniel O. Awduche. Mpls and traffic engineering in ip networks. *IEEE Communication Magazine*, 37(12):42–47, December 1999.
- [29] K. Nichols, S. Blake, F. Baker, and D. Black. Definition of the differentiated services field (ds field) in the ipv4 and ipv6 headers. Technical Report RFC2474, IETF, December 1998.
- [30] I. Castineyra, N. Chiappa, and M. Steenstrup. The nimrod routing architecture. Technical Report RFC1992, IETF, August 1996.
- [31] S. Shenker, C. Partridge, and R. Guerin. Specification of guaranteed quality of service. Technical Report RFC2212, IETF, September 1997.
- [32] J. Wroclawski. Specification of the controlled-load network element service. Technical Report RFC2211, IETF, September 1997.
- [33] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource reservation protocol (rsvp) – version 1 functional specification. Technical Report RFC2205, IETF, September 1997.
- [34] L. Berger, D. Gan, G. Swallow, P. Pan, F. Tommasi, and S. Molendini. Rsvp refresh overhead reduction extensions. Technical Report RFC2961, IETF, April 2001.
- [35] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. Technical Report RFC2475, IETF, December 1998.

- [36] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow. Rsvp-te: Extensions to rsvp for lsp tunnels. Technical Report RFC3209, IETF, December 2001.
- [37] B. Jamoussi, L. Andersson, R. Callon, R. Dantu, L. Wu, P. Doolan, T. Worster, N. Feldman, A. Fredette, M. Girish, E. Gray, J. Heinanen, T. Kilty, and A. Malis. Constraint-based lsp setup using ldp. Technical Report RFC3212, IETF, January 2002.
- [38] D. Mitra and K.G. Ramakrishnan. A case study of multiservice, multipriority traffic engineering design for data networks. *GLOBE-COM'99. Global Telecommunications Conference*, 1b:1077–1083, 1999.
- [39] A. Elwalid, C. Jin, S. Low, and I. Widjaja. Mate: Mpls adaptive traffic engineering. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communication Societies.*, volume 3, pages 1300–1309, 2001.
- [40] T. Li and Y. Rekhter. A provider architecture for differentiated services and traffic engineering (paste). Technical Report RFC2430, IETF, October 1998.
- [41] George Swallow. Mpls advantages for traffic engineering. *IEEE Communications Magazine*, 37(12):54–57, December 1999.
- [42] Xipeng Xiao, Alan Hannan, Brook Bailey, and Lionel M. Ni. Traffic engineering with mpls in the internet. *IEEE Network Magazine*, 14(2):28–33, March 2000.
- [43] Muckai K. Girish, Bei Zhou, and Jian-Qiang Hu. Formulation of the traffic engineering problems in mpls based ip networks. In *Fifth IEEE*

- Symposium on Computers and Communications. ISCC 2000*, pages 214–219, 2000.
- [44] P. Aukia, M. Kodialam, P. V. Koppol, T. V. Lakshman, H. Sarin, and B. Suter. Rates: A server for mpls traffic engineering. *IEEE Network Magazine*, 14(2):34–41, March 2000.
- [45] Murali Kodialam and T. V. Lakshman. Minimum interference routing with applications to mpls traffic engineering. In *Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. IFOCOM 2000.*, volume 2, pages 884–893, 2000.
- [46] Lus Cardoso, Jorge Patro, Carlos Lopes, Amaro de Sousa, and Rui Valadas. Ptplan: A tool for mpls traffic engineering. In *Ninth International Symposium on Modeling, Analysis and Simulation of Computers and Telecommunication Systems.*, pages 339–345, 2001.
- [47] K. Matsui, M. Kaneda, H. Takenaka, H. Ishii, and M. Imase. Tres: traffic and reliability engineering server for mpls. In *IEEE Pacific Rim Conference on Communications, Computers and signal Processing. PACRIM 2001.*, volume 2, pages 583–586, August 2001.
- [48] Nicolas Rouhana and Eric Horlait. Differentiated services and integrated services use of mpls. In *Fifth IEEE Symposium on Computers and Communications. ISCC 2000*, pages 194–199, 2000.
- [49] A. Kankkunen. Mpls and next generation access networks. In *1st European Conference on Universal Multiservice Networks. ECUMN 2000.*, pages 5–16, 2000.

- [50] Ilias Andrikopoulos and George Pavlou. Supporting differentiated services in mpls networks. In *Quality of Service, 1999. IWQoS '99. 1999 Seventh International Workshop on*, pages 207–215, 1999.
- [51] Richard Rabbat, Ken Laberteaux, Nirav Modi, and John Kenney. Traffic engineering algorithms using mpls for service differentiation. In *IEEE International Conference on Communications. ICC 2000.*, volume 2, pages 791–795, 2000.
- [52] Melody Moh, Belle Wei, and Jane Huijing Zhu. Supporting differentiated services with per-class traffic engineering in mpls. In *Tenth International Conference on Computer Communications and Networks*, pages 354–360, 2001.
- [53] P. Trimintzios, I. Andrikopoulos, G. Pavlou, D. Griffin, P. Georgatsos, D. Goderis, L. Georgiadis, C. Jacquenet, and R. Egan. A management and control architecture for providing ip differentiated services in mpls-based networks. *IEEE Communication Magazine*, 39(5):80–88, May 2001.
- [54] Sanda Dragos, Radu Dragos, and Martin Collier. Bandwidth management in mpls networks, 2001. <http://telecoms.eeng.dcu.ie/symposium/papers/B1.pdf>.
- [55] K. Psounis. Active networks: Applications, security, safety, and architectures. *IEEE Communications Surveys*, pages 1–16, First Quarter 1999 1999.
- [56] David L. Tennenhouse and David J. Wetherall. Towards an active network architecture. *Computer Communication Review*, 26(2), April 1996.

- [57] David L. Tennenhouse, Jonathan M. Smith, W. David Sincoskie, David J. Wetherall, and Gary J. Minden. A survey of active network research. *IEEE Communications Magazine*, 35(1):80–86, 1997.
- [58] S. Bhattacharjee, K. L. Calvert, and E. W. Zegura. An architecture for active networking. In *IEEE INFOCOM '97*, April 1997.
- [59] Dan Decasper and Bernhard Plattner. Dan: Distributed code caching for active networks. In *IEEE INFOCOM '98*, 29 March - 2 April 1998.
- [60] D. Wetherall, J. Guttag, and D. Tennenhouse. Ants: A toolkit for building and dynamically deploying network protocols, 1998.
- [61] B. Schwartz, W. Zhou, A. Jackson, W. Strayer, D. Rockwell, and C. Partridge. Smart packets for active networks. In <http://www.net-tech.bbn.com/smtpkts/smart.ps.gz>, 1998.
- [62] David J. Wetherall and David L. Tennenhouse. The ACTIVE IP option. In *Proceedings of the Seventh ACM SIGOPS European Workshop*, Connemara, Ireland, 1996.
- [63] D. Alexander, W. Arbaugh, M. Hicks, P. Kakkar, A. Keromytis, J. Moore, C. Gunder, S. Nettles, and J. Smith. The switchware active network architecture. *IEEE Network*, 12(3), May/June 1998.
- [64] Michael Hicks, Pankaj Kakkar, Jonathan T. Moore, Carl A. Gunter, and Scott Nettles. PLAN: A programming language for active networks. *ACM SIGPLAN Notices*, 34(1):86–93, 1999.
- [65] Y. Yemini and S. daSilva. Towards programmable networks. In *IFIP/IEEE International Workshop on Distributed Systems: Operations and Management*, October 1996.

- [66] Sanda Dragos and Martin Collier. A proposed architecture for integrating active networks and mpls. to be published on SOFTCOM 2002, 2002.
- [67] Jeff Dike. User mode linux. running linux on linux. *Linux Magazine*, April 2001. http://www.linux-mag.com/2001-04/user_mode_01.html.
- [68] Rusty Russell. Linux netfilter hacking howto. <http://netfilter.samba.org/unreliable-guides/netfilter-hacking-HOWTO/index.html>.
- [69] Rusty Russell. Netfilter: Packet mangling in 2.4. In *6th International Linux Kongress*, September 1999. Augsburg.
- [70] Rusty Russell. Writing a module for netfilter. *Linux Magazine*, June 2000. http://www.linux-mag.com/2000-06/gear_01.html.
- [71] Radu Dragos, Sanda Dragos, and Martin Collier. Design and implementation of an mpls based load balancing architecture for web switching. In *The 15th ITC Specialist Seminar: Internet Traffic Engineering and Traffic Management*, 2002.
- [72] David Wetherall, Ulana Legedza, and John Guttag. Introducing new internet services: Why and how. *IEEE Network Magazine*, July 1998.
- [73] Alessandro Rubini. *Linux Device Drivers*. O'Reilly, 1998.
- [74] Stephen T. Satchell and H.B.J. Clifford. *Linux IP Stacks Commentary*. Coriolis, 2000.
- [75] Scott Maxwell. *Linux Core Kernel Commentary*. Coriolis, 1999.

- [76] Tony Li. Mpls and the evolving internet architecture. *IEEE Communications Magazine*, 37(12):38–41, dec 1999.
- [77] Wei Sun, P. Bhaniramka, and R. Jain. Quality of service using traffic engineering over mpls: an analysis. In *25th Annual IEEE Conference on Local Computer Networks*, pages 238–241, 2000.
- [78] Grenville Armitage. Mpls: the magic behind the myths [multiprotocol label switching]. *IEEE Communications Magazine*, 38(1):124–131, jan 2000.