MASTER IN COMPUTER SCIENCE

Traffic engineering in IP over ATM networks

Uhlig, Steve

Award date:
2000

Facultés Universitaires Notre-Dame de la Paix, Namur
Institut d'Informatique
Année Académique 1999-2000

TRAFFIC ENGINEERING

IN

IP OVER ATM NETWORKS

Steve UHLIG

Promoteur : Olivier Bonaventure

Mémoire en vue de l'obtention du grade de Maître en Informatique

Abstract

Today's networking has seen the rise of a new way of thinking the way traffic is handled in the Internet. While best-effort has almost been the only service model in use so far, the requirements in terms of QoS guarantees by many applications, without mentioning network cost, have driven the development of network traffic optimization, called "traffic engineering". Many have perceived MPLS and its traffic engineering capabilities as the tool for providing QoS guarantees over IP.

This dissertation presents some of the currently defined mechanisms for providing QoS. We give some insight for what concerns interdomain traffic behavior. The concepts of "traffic engineering" and QoS often appear in today's networking context. Seemingly, many strategies have already been thought in order to enable "guarantees" to be provided for many types of traffic classes constituting today's Internet traffic, at least in theory.

Today's Internet may be considered as best-effort due to the complete lack of real guarantees in terms of bandwidth, packet loss, delay and jitter. Nevertheless, the ATM technology did already introduce QoS into the Internet by specifying and evaluating its essential aspects. Unfortunately, the widespread IP combined with other factors have not allowed the broad deployment of ATM as a complete solution for the Internet. The evolution of the Internet rather seems to take the direction of using the current IP routing architecture. QoS guarantees to specific flows would thus be provided by "emulating" the connection-oriented nature of ATM via MPLS. The standardization of MPLS by the IETF might constitute a decisive step towards traffic engineering.

Many studies have been carried out on Internet traffic. However, very few have focused on interdomain traffic and none had thoroughly examined interdomain traffic variability before. Considering that heading towards QoS guarantees requires the knowledge of traffic behavior, carrying an interdomain traffic analysis was necessary to provide a deep understanding of the broad characteristics of Internet traffic with regard to variability.

We discuss some implications of using MPLS to carry best-effort traffic. We stress the problems that might appear while trying to provide QoS guarantees to "best-effort" traffic. We evaluate interdomain LSP variability as well as bandwidth reservation techniques for interdomain flows.

We show that some change is required for what concerns the way sources generate traffic. Using as much bandwidth as one can without taking into account other traffic sources (or being obliged to do so) may be a serious problem. Traffic burstiness at the interdomain level might prevent ISPs to provide QoS to their customers. Not at least without relying on huge over-provisioning, a current practice today.

Traffic engineering is a broadly used term in networking today. However, its classical definition as "traffic optimization" may be overstated. We are far from being able to optimize anything with regard to interdomain traffic. Therefore, the aim of this dissertation is at bringing some traffic-centered view about realistic traffic engineering capabilities. We try to qualitatively evaluate the existing solutions that intend to tackle the QoS issue.

The pessimistic conclusions we draw are no reason to think that traffic engineering is useless. The deployment of Diffserv and other service models is likely to change a lot with regard to traffic characteristics because resource use will relate with billing. Sources will adapt their behavioral characteristics to the change in the economical model of the new QoS Internet. The day of fixed-price resource consumption should very soon be over. Traffic engineering makes sense in an Internet where traffic characteristics may be predicted. Resource-oriented billing might transform the Internet in such a way.

Acknowledgements

This part probably is the easiest one because it is about recognizing everyone's contribution for what concerns some of my greatest pleasures: work and research. The first reason for which I would like to thank my advisor, Professor Olivier Bonaventure, concerns the opportunity he gave me to work on such a marvelous subject. It allowed me to get what I like the most in computer science: new and counter-intuitive results. Anything you do brings its number of mistakes, findings and new questions. The second reason consists in the numerous hours we spent together at trying to understand the potential reasons of the many unexpected results we got, sometimes finding reasons, some other times getting more confused than ever before. These moments constitute some of the most constructive ones of my formation as a computer scientist and a researcher. Not only his open-mindedness but also his continuous will at trying to understand the reasons of every result provided the drive to go deeper into the numbers. Research is about questioning and never accepting a "well-established" fact just because someone stated someday it has to go that way. Even if you are convinced it works in a certain way, verify. You will be either deceived because it actually works that way or you will learn something! The final reason for which I would like to thank Olivier relates to the way he copes with problems: try to resolve them by yourself! The easiest way is not the best. Maybe you will loose some time, but you will learn much more than by bypassing it. He always tries to make you resolve them by yourself, if you are not able to do it now, you will learn! This often made me confused, grumble (against the situation) and even sometimes angry. Benefits cannot be perceived on the very moment. Later, by getting some relative view of the situation, you admit it was the ideal way. I think now that he guided me along one of the best paths towards networking research: practice or how he likes to say "les mains dans le cambouis" (© O. Bonaventure).

The other person at FUNDP that helped me a lot is Aimé Kassa Molala by his sense of humor, his permanent good mood and his "last but not least" pure and innocent mentality. He did many things that transformed my working into a much more enjoyable activity.

Traffic traces analysis could not have been carried without the help of Rudy Van Gaver, Damien Collart and Marc Roger, so we probably would not be here if they did not sacrifice their time. Getting interdomain traffic traces is a difficult task, even requiring some luck. They are this luck. They did many

things that helped us, not only by providing traffic traces but also by giving some very helpful comments with regard to our traffic analysis.

There are many other people that did more or less directly contribute to this dissertation whose name will not be mentioned here: students at FUNDP, friends…and others.

However, special thanks to a close friend of mine, Xavier, who has had to endure many very long e-mails and tiresome conversations for several years.

Finally, my greatest thanks are directed towards my parents and my sister for the care they took of me and their wonderful and never-ending patience. My dedication at work and the satisfaction I get from research do not come with benefits only: it takes a lot of time to get answers from numbers and concepts. All of this work has been carried at the expense of the attention I should have provided to them. I sincerely apologize for that. I would also like to thank my parents for the upbringing they did provide to me. They made of me what I am today…

For my parents and my sister…

Contents

List of Figures

Chapter 1

Introduction

1.1 The Internet Protocol

The IP protocol has been designed to interconnect systems of packet-switched computer communication networks. It provides a means for transmitting blocks of data called "packets" from sources to destinations. All hosts in the Internet are identified by their IP address, which is a 32-bit number. IP addresses also serve at transmitting packets towards their destination. The selection of a path for transmission is called routing. IP treats each packet as an independent entity unrelated to any other packet. There is no concept of connection or logical circuit. Hosts are partitioned into two types: hosts and gateways. An IP host is the ultimate consumer of communication services. It executes application programs on behalf of one or more users and employs Internet communication services in support for this function. Gateways (or routers) are the packet-switching computers that interconnect networks.

According to [Bra89], the architecture of the Internet relies on several assumptions:

- The Internet is a network of networks: each host is connected to some network. The connection of a particular host to the Internet is purely conceptual. Two hosts in the Internet communicate with each other using the same set of protocols, no matter they are located on the same network or not.

- Gateways do not keep connection state information: robustness is a basic objective of the design of IP. Gateways are designed to be stateless. This permits to exploit redundant paths to provide robust service in spite of failures that could occur on the path between the source and the destination. The state information required to for end-to-end flow control and reliability is implemented in the hosts. This ensures that connection control information cannot be lost unless one end-point fails.

- Routing complexity is located in the gateways: routing is complex. This is why it should be performed by gateways only. Hosts should not suffer from a change caused by the evolution of the Internet routing architecture.

- The Internet must tolerate wide network variations: the Internet is due to cope with a wide range of network characteristics (e.g., bandwidth, delay, packet loss...).

1.2 Intradomain vs. Interdomain Routing

The Internet implements adaptive routing. Adaptive routing means that routing decisions change as conditions on the network change. The main conditions that influence routing decisions are failure and congestion. Node (or link) failure implies that the paths traversing the node or link cannot be used any more. Congestion means that the incoming traffic exceeds the outgoing forwarding capacity at a particular area: packets should then be routed around the area to prevent their loss. The problem with adaptive routing relates to the information about the state of the network that needs to be exchanged among the nodes. There is a tradeoff between the accuracy of the routing information and the overhead that arises from this informational exchange. More routing information (in size and

frequency) implies a better routing decision. However, information exchange implies a load on the network, thus performance degradation.

Routing in the Internet today relies on the partitioning into autonomous systems. An Autonomous System (AS) is a group of routers that exchange information via a common routing protocol. An AS is due to be managed by a single administrative authority. An AS should always be connected to the Internet, except during failures. The Internet routing architecture relies on the concept of AS to divide the routing function into intradomain routing (within an AS) and interdomain routing (between ASs). An interior routing protocol (or IGP for Interior Gateway Protocol) passes routing information between routers within an AS. The protocol used within the AS is not required to be implemented outside of the AS. On the other hand, an exterior routing protocol (or EGP for Exterior Gateway Protocol) serves at exchanging routing information between routers belonging to different ASs. An EGP is expected to require less information exchange than an IGP. The reason is that when a packet traverses several ASs on its path between the source and the destination, a router in the source AS only needs to determine the target AS and a corresponding route. Every time the packet enters an intermediate AS, the interior routers can cooperate to forward it inside the AS. The EGP should not be concerned about intra-AS routing, only about inter-AS routing.

1.3 Traffic Engineering

While the Internet routing architecture relies on IGPs and EGPs, none of them offers sufficient traffic control capabilities. Reporting and incorporating in the routing information resource availability or utilization is poorly achieved by today's routing algorithms. The routing algorithms currently used tend to converge traffic onto the same network links. This contributes to congestion and unbalanced network resource utilization. The lack of control over the existing routing system makes it difficult to implement effective policies to address the network performance problem.

The need for traffic engineering finds its cause in the evolution of the Internet. The Internet has evolved into a critical communications infrastructure, supporting important economic, educational and social activities. At the same time, the delivery of Internet services has become a very competitive market. Hence, optimizing the performance of large IP networks has become an important issue. Furthermore, due to the complexity of the network performance requirements, the traffic engineering problem is very challenging. If in addition several traffic classes are considered, each requiring a particular treatment by routers, resource sharing issues could transform traffic engineering into a very tricky game.

1.4 MPLS

MPLS is a connection-oriented forwarding scheme that includes extensions to conventional IP routing protocols. It extends the Internet routing model and enhances packet forwarding and control (see [MPLSAR]). At the ingress router of an MPLS-capable network, IP packets are classified and routed based on a variety of factors. These factors include a combination of the information carried in the IP header and the local routing information maintained by the LSR (label switched router). MPLS is a very powerful technology for Internet traffic engineering because it supports explicit routes, which allows the implementation of constrained-based routing in IP networks (see [MPLSTE]). Traffic engineering deals with performance evaluation and performance optimization of IP networks. MPLS also allows a more hierarchical routing function than what is done today with interdomain routing by using a stack of headers. This stack may be viewed as a stack of IP addresses with the difference that labels do not identify IP hosts but only MPLS paths, known as LSPs.

1.5 Dynamics of IP Traffic

While the traffic engineering objective relates to the optimization of network traffic, it assumes a preliminary knowledge of IP traffic dynamics. Routing protocols might be considered as the engineering part of traffic engineering. However, traffic engineering must also deal with the characteristics of the traffic that floods through the network. Evaluating traffic dynamics is important before ever thinking about traffic optimization. The knowledge of intradomain traffic variability is important to ensure a good balance of the traffic within the network. Unfortunately, intradomain traffic distribution within the network depends on the behavior of interdomain traffic. Trying to optimize intradomain traffic without taking into account interdomain traffic characteristics does not make sense. Because interdomain routing encounters more difficulties than intradomain routing for what concerns routing information exchange, interdomain traffic dynamics is much more critical. In addition, the price of interdomain bandwidth is today far higher than for the intradomain case.

1.6 Goals and Scope of this Dissertation

The purpose of this dissertation is to provide an interdomain perspective on the subject of traffic engineering. Traffic engineering is a very wide subject that would require far more than a simple dissertation. We thus limit the subject to traffic-oriented and interdomain traffic engineering. By traffic-oriented, we mean that the focus is placed on traffic characteristics. Traffic control has already been covered in details so that we felt important to develop a good intuition about interdomain traffic dynamics. No known study had ever been carried for what concerns interdomain traffic variability yet. We thus analyzed interdomain traffic traces in order to determine whether problems might appear if MPLS were used to carry best-effort traffic at the interdomain level. We relied on measurements, not simulations because all factors having an impact on the performance of the real system are present in the formers. The main issue at carrying traffic traces analysis relates to the expensiveness of such studies. A summary of the operational aspects of our study is presented in Appendix 5. The dissertation is structured as follows.

Chapter 2 introduces the subject of label switching. We compare the approaches of IP and ATM with regard to routing. We discuss the performance of both forwarding paradigms. We also present some theoretical aspects of label switching, providing a more general view of routing. Readers interested in the different approaches implementing label switching can find them in Appendix 4. Appendix 4 also constitutes a good introduction to Chapter 3 since the designers of the MPLS standard gathered many ideas from those label switching implementations. Chapter 3 presents an overview of the new MPLS standard. MPLS is a technology designed to use label switching, thus aimed at unifying layer 3 (e.g., IP) and layer 2 (e.g., ATM) forwarding paradigms. The most interesting features of MPLS related with its traffic engineering capabilities are discussed. We then plunge ourselves into traffic engineering with Chapter 4 where the main existing solutions for traffic engineering are evaluated. The previously mentioned chapters may be viewed as a prerequisite to understand the factors that drove the need for interdomain traffic engineering.

Chapter 5 constitutes the core of this dissertation. It presents our interdomain traffic traces analysis. This chapter studies the variability of interdomain traffic for two very different ISPs. It exhibits the many problems that might appear if MPLS were used in the Internet without changing the way traffic behaves. We discuss the different issues related with interdomain traffic engineering. Because interpreting interdomain traffic traces requires a deep understanding about the way interdomain routing works in the Internet today, we refer readers that would be new to the subject of interdomain routing to Appendix 6.

Finally, we draw our conclusions in Chapter 6 and we discuss some further work.

Chapter 2

Label Switching

2.1 Introduction

This chapter first presents the essential concepts allowing a broad understanding of today's IP routing. We explain the functioning of IP routing as well as its limitations. The reasons why classical IP routing should be changed are also briefly introduced. We then explain several solutions that aimed at resolving classical IP routing limitations through an historical perspective. The remainder of the chapter discusses label switching concepts.

2.2 Routing in the Internet Today

The Internet today is mostly based on the TCP/IP protocol stack. IP routing relies on what is called "best-effort". The IP routing protocol offers only best-effort delivery of packets between hosts attached to the Internet. Best-effort implies the absence of guarantee concerning the delivery of packets. IP tries to make its best to carry the packets from the source to the destination but loss, non-delivery, corruption and reordering may occasionally occur. TCP tries to make up for this lack of guarantee by controlling the packet flow at both end-systems.

Each host in the Internet is identified by its IP address. An IP address is 32 bits long and comprises two parts: a network identifier part and a host identifier part. The network identifier part, as its name implies, identifies the set of hosts managed by the same organization (a network). The host identifier identifies the end-system within the organization's address space. Since network sizes are due to be different depending on the needs of the organization, there are several address classes with different lengths of the network identifier. This enables organizations to choose between several network sizes when connecting to the Internet.

IP routing is the mechanism by which a router forwards an IP packet. The routing of a packet involves receiving it on an input port, examining some part of it and sending it on the appropriate output port based on the latter examination. The field (or set of fields) of the packet which permitted the router to make the decision of the output port where to forward the packet is called the header. A header is a generally short and fixed-length identifier used to forward packets. Depending on the technological environment, the semantics of the label is explicit or implicit. When the header is the IP destination address, we say that semantics of label is explicit because we see the IP host which IP address has same value as the header. On the other hand, when the header is a short local identifier, we say that semantics of header is implicit because the header means nothing for us but has only local significance for the router.

IP routing is based on explicit-semantics header, which means that addresses are global for the Internet (or unique at a given time). This property of address semantics implies that forwarding correlates with global topology knowledge. Unfortunately, the tremendous growth of the Internet led to the well-known problem of scalability. Scalability is the property of a network to be able to sustain any growth as important as it be. The problems arising from the growth of the Internet are the limited address space provided by IPv4, the tremendous growth of routing tables, the bandwidth requirements (real-time and multimedia applications are heavy consumers) and the pressure placed on the functionality of

routing protocols (resource reservation, multicast, QoS...). Today, more than 50 % of the addresses have been attributed. Figure 2.1 shows the growth in the number of Internet hosts from 1992 in a logarithmic scale. The growth experiences an exponential behavior to attain a number of about 18 millions hosts in 2000.

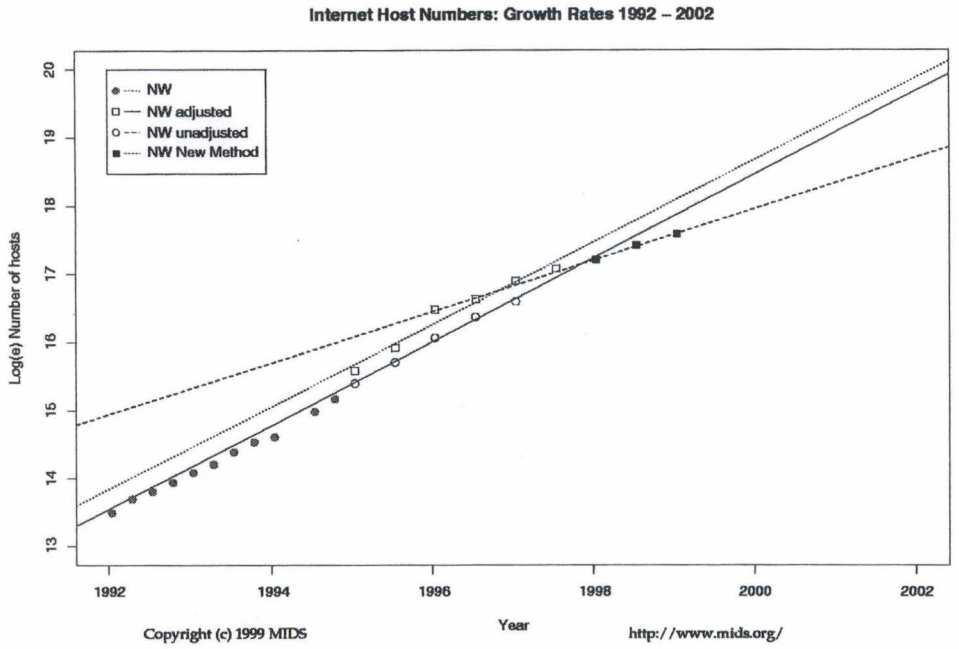

Figure 2.1: Growth in the number of Internet hosts

Routing in the Internet rely on routers that forward IP packets from router to router on a path from the source to the destination. Routers need thus base their routing decision on knowledge of the topology and conditions of the Internet. This topology knowledge is located in the routing table of every router. A routing table binds a particular value of the IP address of the destination (or a subset of it) to a particular output port on which the packet should be forwarded in order to reach the destination. The size of the Internet does not allow every router to maintain a complete map of its topology. A partial map must suffice if the routing function has scalability as an objective. Because the conditions of the Internet often change, adaptive routing is used. Of course, if nothing ever changed, routing would be a lot simpler: a routing table could be like a static associative table in which each entry never changes. Any destination address would always be associated with the same output port at a given router. Unfortunately, congestion and routers failure happen. When a router fails, it can no longer be used as part of a routing path. When a particular point of the Internet experiences severe congestion, i.e. the incoming traffic at a particular router exceeds its forwarding capacity, the packet which path was due to traverse the congested area should preferably be routed around it. Not only the packet has a high loss probability by using the congested path but also sending packets to an already congested area is not going to help.

Figure 2.2 shows the simplified structure of a routing table. We see each entry (line) associating an IP prefix with a next hop router. An IP prefix is a $<IP\ address, masklength>$ pair where *masklength* specifies the number of significant bits (from left to right) of the *IP address* (here in decimal representation). When a packet arrives at an incoming interface, the router tries to find a "best-matching" entry in its routing table. The best-matching entry is the one having the longest prefix in common with the packet IP destination address. Finding the best-matching entry requires to perform a table lookup and to compare every routing table entry with the destination address. At best, the required number of operations to determine the output port is $log_2(number\ of\ entries)$ if we suppose that table organization is purely hierarchical (ordered by 32 bit prefix lexicographical order). Given that

a complete routing table contains about 70.000 entries[1] for a big ISP's router, it makes around 15 comparisons for each packet to forward if some binary-tree-based index is used to find the best-matching entry. When millions of packets need to be forwarded every second, one can realize the burden placed on routers in the case of adaptive routing. Because adaptive routing requires taking into account the conditions of the Internet, routing tables need to be updated on a regular basis.

Destination IP	Next Hop router	Interface
3.0.0.0/8	192.168.0.1	atm0
4.0.0.0/8	192.168.0.1	atm0
4.24.148.0/24	192.168.0.1	atm0
6.0.0.0/8	192.168.0.1	atm0
9.2.0.0/16	192.168.0.1	atm0
9.20.0.0/17	192.168.0.1	atm0

Figure 2.2: Simplified routing table example

Adaptive routing suffers from several drawbacks. First, routing decisions are more complex. Since the router has the choice between several routes, some criterion has to be used to decide which entry will be chosen to make the forwarding decision. The algorithm used today is the "shortest path" algorithm: among all best-matching prefixes, the one having the smallest number of intermediate hops is chosen. We call the function used to select the routing entry the "routing metric." Note that we always have to choose the shortest path entry among all best-matching entries because we also want the selected entry to be the most specific one. This is correct if we assume a "longer-matching" prefix to be closer from the destination. Actually, the best-matching prefix should also be the one that is the most likely to be the shortest path one. Second, the quality of the routing decision in terms of the routing metric depends on its informational complexity. The more information exchanged about the network conditions, the better the routing decisions but also the more important the burden placed on routers. One cannot expect to have a routing table giving the instantaneous conditions of the whole Internet. A trade-off must be found between the accuracy of the information and the amount of overhead generated by routing tables update. Finally, synchronization of routing information may produce pathologies like route oscillations and even cause congestion or loops. Reacting too quickly could make the router re-route traffic to other paths by considering local congestion that did already resorb. On the other hand, reacting too late may produce severe congestion or packet loss due to lack of adaptability. Adaptive routing can therefore enhance routing performance but at the price of complexity.

One solution to be proposed was Classless InterDomain Routing (CIDR), which helped resorb address space waste and permitted more scaleable routing due to a more hierarchical routing function. The motivations for CIDR were the exhaustion of class B network addresses, the growth of routing tables in the Internet and eventual exhaustion of the 32-bit IP address space. Note that the main objective of CIDR was to limit the impacts of Internet growth, not to solve the problem, which had to be done in another way (via a long-term solution). The addressing and routing plan associated with CIDR consists in distributing the allocation of Internet address space and providing a mechanism for the aggregation of routing information (see [FLY+93]). This plan permits IP network prefixes (and masks) to be of any length compared to previously 8, 16, 24 bits. The benefits of the new addressing plan were the improvement in the assignment of class C's to mid-sized organizations (200-4000 host range) and an immediate decrease in the number of routing table entries[2], followed by a significant reduction of the growth rate of routing table size. Figure 2.3 shows the growth in routing table size from 1989 to early 2000 (see [Telstra]).

[1] By the end of 1999.

[2] After deployment of new interdomain routing protocol.

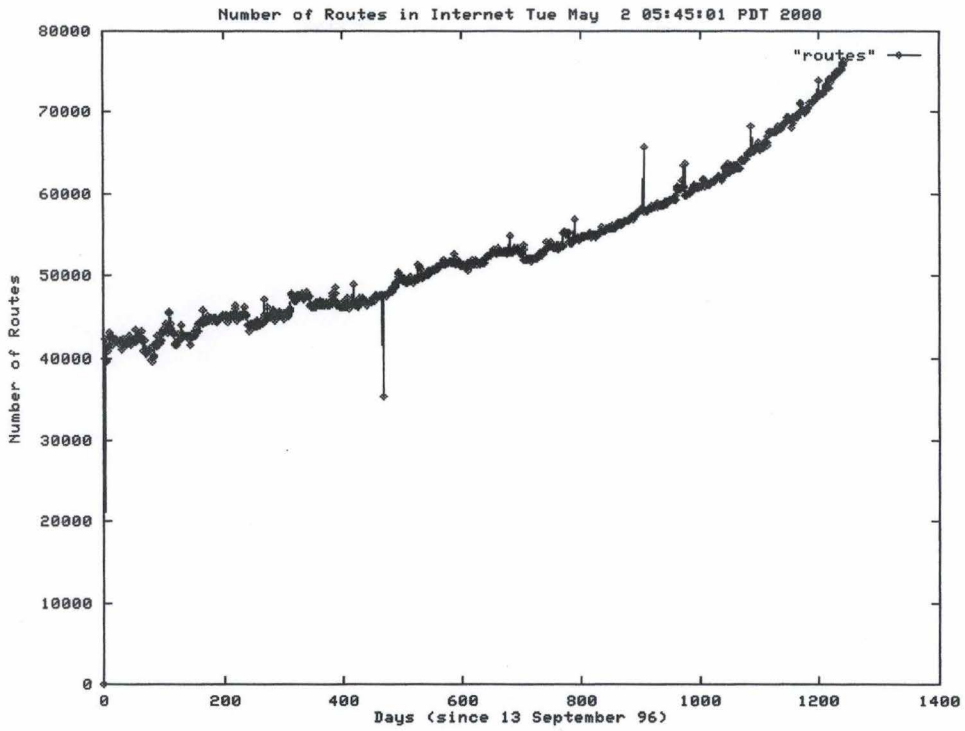

Figure 2.3: Evolution of routing table size

IPv6 (see [IPV6]) was also to be the basis of the next generation IP networks. Actually, it addresses the very specific problem of IP address space and does not change anything at the routing functionality. New functionality like resource reservation, security, multicast and so on will be resolved by other protocols that do not need to depend on IPv6 (like RSVP, RTP, RTCP, etc.).

2.3 Routing vs. Switching

One of the essential components of an internetwork is the router. With the emergence of the new LAN products like Fast Ethernet and Gigabit Ethernet, LAN bandwidth does not constitute a bottleneck any more: the device connecting to the Internet inherits all the pressure. A crucial question arising from the evolving demand for more and more bandwidth concerns the layer at which the forwarding decision process should be made: layer 3 (router) or layer 2 (switch)?

Switching uses a fixed length subset of the packet header and requires only one table lookup to determine the output port on which to forward the packet. While routing implies making a non-trivial choice, switching uses an exact match algorithm in the "switching" table. This is why switching is a lot faster. The price for speed is limitation in routing functionality. Forcing the switching tables to be simpler means that the routing decision cannot rely on complex information. Switching also requires the use of the connection-oriented paradigm: a simpler lookup operation requires stability in the switching table. A connection need be established between the communicating parties (routers or hosts) for which the switching table entry has been created before being able to forward traffic.

At an efficiency point of view, one would like to get a routing device with router's functionality but switch's price/performance ratio. A router acts at layer 3 of the OSI model while a switch at layer 2. That means that a switch is simpler and faster since it only forwards packets or cells based on label value. For its part, a router has much more network-level functionality (route optimization, differentiated behavior according to source/destination address, etc.). This issue is not that simple because it is about making a technological choice: routers that are as fast as switches and cheap at the same time are not real (yet)! One must be aware that if he wants to get into native ATM technology (see

[DS99]), he goes towards high bandwidth networks but also internetworking issues due to the interaction between the "simple" connectionless IP and the connection-oriented ATM and its complex signaling procedures. That does not mean it is better to stay with classic IP routers in order to avoid interoperability problems.

2.4 IP and ATM: the Internetworking Solution?

Since late 1980's, much has been done to standardize ATM protocols in an effort to make ATM the new "killer technology" and the paradigm aimed at replacing IP. Unfortunately, idealism and standardization do not match together in the reality of the networking industry. IP connects approximately 18 million hosts on ... thousand interconnected networks spread over more than ... countries and has proven to be very robust and flexible: the death of IP will not come very soon! On the other side, despite all the efforts put by the ATM Forum and the ITU and the interesting properties of ATM[3], its role in the near future will probably be limited to some ISP backbones and corporate networks. With this in mind, the IP community began working on IP over ATM, ATM only being one subnetwork technology among others (like Frame Relay, SDMS or Ethernet). If one could merge the respective advantages of IP and ATM in a single solution, this would be as a serious progress towards fast IP and QoS support. At the same time, it could be the solution to all the problems encountered by IP. Unfortunately, the IP over ATM integration raises some serious challenges. The complexity of ATM signaling protocols makes it difficult to just run the connectionless IP over the classical ATM stack. This approach has been used in the following "solutions": LAN Emulation, Classical IP over ATM, Routing over large clouds and Multiprotocol over ATM.

LAN Emulation (see [LANE95]) emulates a physical shared medium (suppose Ethernet for our example) over an ATM subnetwork. The principle relies on the use of an address resolution server (LES for LAN Emulation Server) to convert MAC addresses of the stations into ATM addresses. Once the ATM address discovered, a point-to-point ATM VC (Virtual Circuit) between the two LECs (for LAN Emulation Client) responsible for their respective Ethernet segment is established via ATM signaling procedures (see Figure 2.4). Each LEC manages all VCs established from or towards one of its Ethernet segment stations. Suppose a station situated on Ethernet segment 1 (say station S) wants to communicate with another station situated on Ethernet segment 2 (say station D). Station S first sends an ARP request that is treated by LEC 1. LEC 1 has a configured VC with the LES. It sends via the ATM VC a message with the MAC address of the ARP request to the LES. This allows LES to update its resolution table with the MAC address of station S (associated with the ATM address of LEC 1). If the LES already knows the ATM address associated with station 2 (by example if LEC 2 has already registered station 2 to the LES), the LES replies to the request by giving the ATM address of LEC 2. If the LES does not know it, it broadcasts a request to all LECs to know whether one LEC has this MAC address on its respective Ethernet segment. Once the LES has learned the ATM address of the LEC responsible for station D (LEC 2), it sends a reply to LEC 1 with ATM address of LEC 2. LEC 1 then establishes an ATM VC with LEC 2 through the ATM switches by ATM signaling procedures. Now, LEC 1 can say to station S that it is able to send packets to station D. LEC 1 will thus receive frames from station D, place them in ATM cells, send the cells over the VC established with LEC 2. LEC 2 will receive the cells on the established VC, place the cells into Ethernet frames and send them to station D. The purpose of a LEC is therefore to register its stations to the LES and to manage all VCs from and towards its stations with stations situated on other Ethernet segments of the Emulated LAN. This approach has some serious drawbacks since the server side constitutes a single point of failure and scaling to large networks cannot be considered.

[3] High capacity, bandwidth scalability and ability to support multi-service traffic.

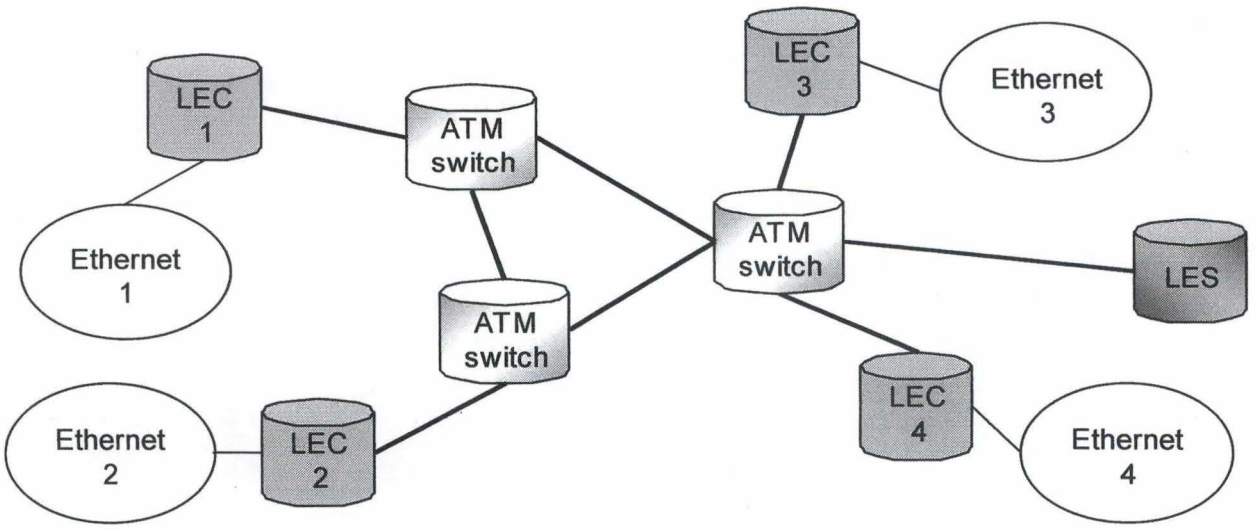

Figure 2.4: LAN Emulation

The next approach is Classical IP over ATM (see [Lau94]). The main concept is the LIS or Logical IP Subnet. Each LIS communicates with each other through the same ATM network. Each LIS is independent from each other and communication occurs through an IP router that is configured as an ATM endpoint. If two stations situated on different LISs want to communicate, they are obliged to pass through an intermediate router even if they are situated on the same ATM network. Figure 2.4 gives an example of several LISs connected through an ATM switch. Two LISs could very well be on a common ATM subnet without being able to communicate directly by using ATM signaling operations. Classical IP over ATM suffers from similar drawbacks to those of LAN Emulation. Routers interconnecting LISs are the bottlenecks and scaling to large networks is an issue.

Routing over large clouds (see [LKP+98]) was intended to address the communication problem between LISs in a large homogeneous ATM network (which is called cloud or non-broadcast multiple access network or NBMA). It consists in locating the exit point in the cloud nearest to a given destination, obtaining the ATM address of this exit point and establishing a VC across the ATM cloud to the exit point. The NBMA address resolution protocol (NARP) is a server-based solution similar to Classical IP over ATM. The NBMA next hop resolution protocol (NHRP) extends NARP with routers implementing address resolution services rather than forwarding services. This functionality allows ATM VCs establishment across multiple LISs.

Finally, Multiprotocol over ATM (MPOA) (see [MPOA97]) looks like a summary of the concepts of LAN Emulation, Classical IP over ATM and NHRP into a single protocol. The failure of the previous approaches comes from their desire to achieve direct connectivity across subnets. IP assumes that subnets interconnect at the network layer and that no host is able to communicate to another host located in a different subnetwork at layer 2. This opposition in the connectivity model leads to an opaque view of network topology since there are two separate routing protocols running at the same time (IP and ATM both have their routing protocols) at distinct layers. IP cares about network reachability while ATM cares about physical reachability: when connectivity loss occurs, which protocol knows the truth? A solution would be to strip the signaling procedures from ATM, conserving what makes ATM attractive (high speed through hardware and QoS support) and lay the IP stack on top to provide scalability and flexibility. Another challenge consists in making it work! This is why proprietary solutions have been proposed for the integration of IP over ATM like the "Cell Switching Router" from Toshiba, "IP Switching" from Ipsilon, "Tag Switching" from Cisco or "Aggregate Route-based IP Switching" from IBM (see Appendix 4). These proprietary solutions have been proposed in an attempt to satisfy the demand for IP over ATM products, waiting for a fully open and standardized

solution. All these approaches rely on proprietary protocols that implement all routing functionality on top of ATM hardware.

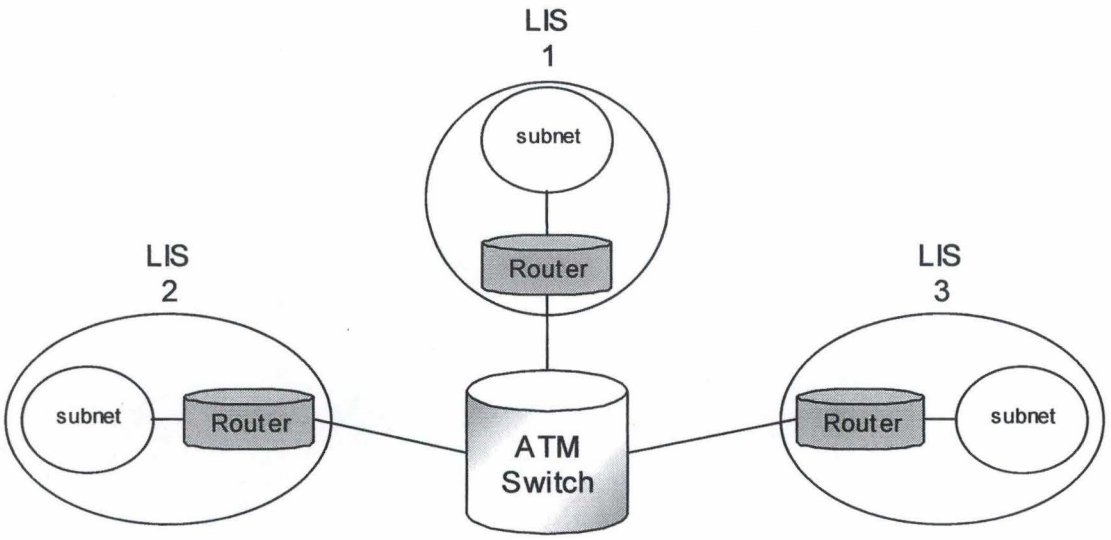

Figure 2.5: Classical IP over ATM.

Speaking about IP and ATM integration, two solutions are conceivable: the use of ATM to convey IP packets and the revision of IP to support the flow concept and resource reservation functionality. Either we continue to use ATM at layer 2 to convey IP datagrams or we emulate the connection-oriented nature of ATM on IP. The proprietary solutions already proposed make use of ATM to convey IP datagrams. They do not rely on internal IP mechanisms but support the flow concept and resource reservation via proprietary protocols (see Appendix 4). On the other hand, MPLS (see Chapter 3) extends the existing IP routing architecture by emulating the connection-oriented nature of ATM.

2.5 Label Switching – Some Theory

2.5.1 Labels

As we saw before, there are two flavors of labels: implicit or explicit. Each of them has advantages and drawbacks. Implicit labels have the advantage of permitting to make a forwarding decision without being obliged to know anything about the packet (or cell) at the intermediate nodes. This allows fast packet forwarding. On the other hand, explicit labels give the opportunity to learn network-level information like destination and source addresses, what kind of treatment the router should apply, or for which application this packet has been sent. We will see that it is possible to make "implicit-flavored label switching forwarding decision" with the advantages of "explicit-flavored label switching".

2.5.2 Forwarding Equivalence Class and Flow Classification

An important concept in label switching is the FEC or Forwarding Equivalence Class. A FEC is a subset of the packets received by a LSR (Label Switched Router) that should receive identical treatment within the LSR. The term LSR replaces the one of "router" or "switch". It designates a device that forwards packets or cells in the label switching context. The particular treatment allows differentiated handling of packets belonging to different FECs within an LSR. The FEC concept constitutes an

opportunity to define a granularity-dependent treatment by which the network can apply a traffic policy to transit flows (traffic that only passes across the network). The notion of granularity closely relates to the layer at which the differentiation occurs. It defines the criteria upon which an LSR determines the subset of the packets that belongs to a particular FEC. It could be the application sending the packets, the IP destination, the *<source host,destination host>* pair, the ingress LSR, etc. FECs allow the network to give distinct levels of priority (and protection) between packets. Implicit semantics of label means that granularity has no effect on the value of the label since the two concepts are orthogonal. The latter property ensures no side effect of label's value on the granularity-related capabilities of the FEC.

2.5.3 Network Layer Routing

Network layer routing is a fundamental component of label switching. It can be partitioned into two components: forwarding and control. The forwarding component operates within a particular LSR while the control component tackles the interoperability aspect of forwarding (consistency between forwarding tables).

2.5.3.1 Forwarding Component

The label switching forwarding component has to make a consistent mapping between labels and FECs. The forwarding component comprises the information from the packet and the procedures used to find the entry in the forwarding table. The forwarding process uses a single forwarding algorithm based on label swapping (ATM-like). Label swapping is the action of replacing the current label by the appropriate one in the packet (label swapping relies on the mapping between one input and one output label). The forwarding component defines the label as being the information used by an LSR to find the right entry in the forwarding table (exact match on the label).

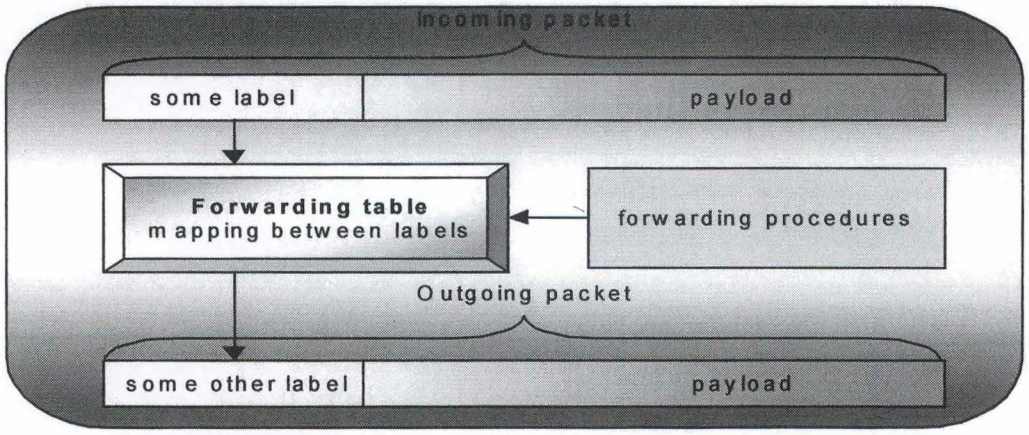

Figure 2.6: Forwarding component

2.5.3.2 Control Component

The control component provides a consistent distribution of routing information among LSRs and procedures to convert routing information to a forwarding table. The distribution of the routing information is executed by the distributed (in the sense that it is organized in some way) exchange of information of multiple routing protocols running on the LSRs.

Let us go a little deeper in the actions of the construction of the forwarding table. The control component has three things to do: make the binding between FECs and labels, inform others LSRs about local binding and use the two latter constraints to construct its forwarding table. The control

component verifies several properties that have an effect on the forwarding table construction, as seen in the following sections.

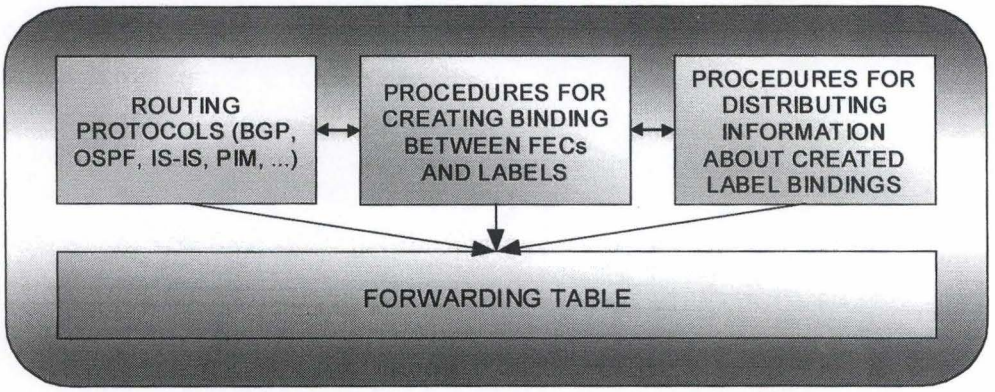

Figure 2.7: Label switching control component

2.5.4 Label Binding Methods

Label binding methods may be classified according to three criteria:

- Local or remote binding: either the label is chosen and assigned locally (local binding) or the LSR receives from other LSRs the label binding information corresponding to some label binding created by these other LSRs (remote binding);

- Upstream or downstream binding: downstream label binding means that the LSR that decided of the value of the label is located downstream with respect to the flow of (data) packets (binding information goes upstream). Upstream label binding is the opposite, which means the LSR that decided the value of the label is located upstream with respect to the flow of (data) packets;

- Control-driven or data-driven binding: a binding between a FEC and a label may be created and destroyed according to two techniques. The first consists in triggering the creation and destruction of a label binding by data traffic flows which is called data-driven because the label binding's existence depends upon the transit of packets in a particular FEC. The second method relies on control information coming from routing or resource reservation protocols. It is therefore called control-driven. The data-driven approach has the advantage to be more traffic-adaptive than the control-driven solution. However, it has the eventual drawback of generating much more control traffic for label binding distribution than the control-driven approach. Traffic characteristics might indeed prevent the data-driven approach from being effective. A very dynamic traffic pattern without sufficiently long-lived flows might imply an important label binding overhead. This phenomenon could make the data-driven solution non-viable since it would even not make sense to set up any label binding according to data traffic. The advantage of the control-driven method is that labels are pre-computed and binding information is only distributed in response to changes in FEC to next hop mapping. Data-driven label binding depends on a mix of control information (for label binding distribution) and data traffic when control-driven only needs its control information. In summary, data-driven means adaptive and more "useful" (since label binding decisions rely on experienced traffic) but more difficult to implement when control-driven means simpler but more arbitrary label binding decisions.

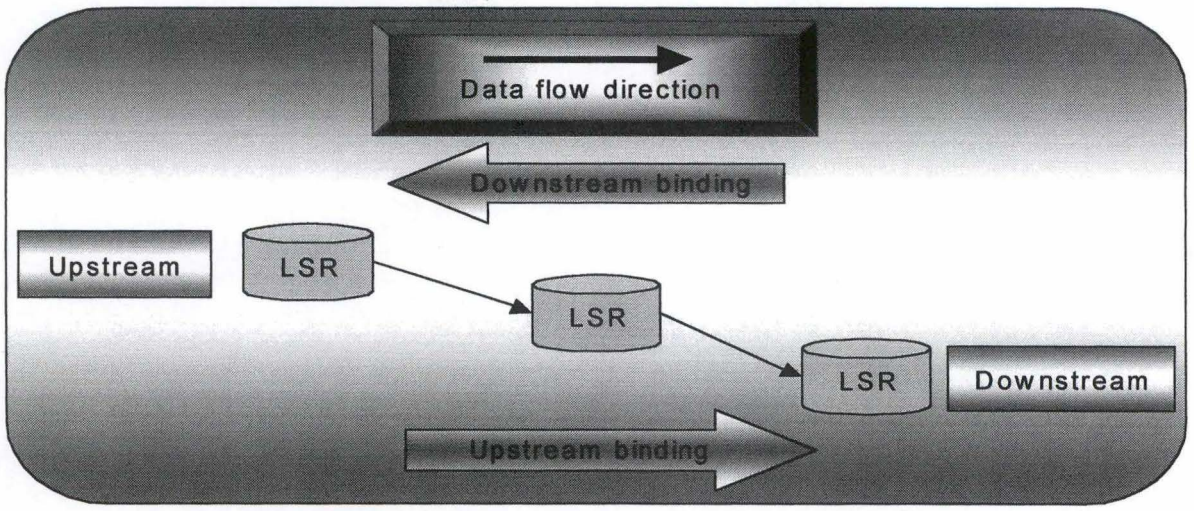

Figure 2.8: Upstream vs. downstream label binding

2.5.5 Label Binding Information Distribution

Two methods exist to distribute label binding information across LSRs: piggybacking on top of routing protocols or resorting to a label distribution protocol.

The first solution is only possible with a control-driven approach since label distribution is tied to the distribution of control information. The positive aspect of piggybacking concerns consistency: Because routing and label binding information are carried together, it avoids transient periods during which routing information is changing, resulting in inconsistency of label attribution within the network. It also simplifies the label distribution mechanism since it does not require a separate label distribution protocol. The problem with piggybacking relates to the protocols used to carry label binding information: not every routing protocol fits. Only some of them explicitly transport mapping between FECs and next hops. Even when the protocol permits to carry that information, another problem may arise: protocol extendibility capability. Some protocols allow changes to their message format while others do not. A desirable solution might therefore also be an infeasible one.

The second solution should be considered as a backup since its sole advantage is to make up for the unfeasibility of piggybacking. Because of the partitioning between routing information and label binding information, race conditions might appear. It is possible that an LSR be waiting for routing information from some routing protocol while label binding information is ready. The other tedious drawback relates to the necessity of making the label distribution protocol interoperate with the routing protocols. In fact, some protocols use incremental updates (e.g., BGP) while others use periodic refreshes of complete routing information (e.g., PIM). A unique label distribution protocol must cope with several different types of routing protocols, which use different methods to distribute their routing information. This makes the implementation a somewhat hazardous operation. A solution to this problem might rely on the definition of several label distribution protocols but this only aggravates the interoperability problem and adds complexity to the label distribution mechanism.

Piggybacking therefore constitutes the best solution. However, if piggybacking were unfeasible then a label distribution protocol would suit.

2.5.6 Creation of Forwarding Entries

There are two strategies for creating forwarding table entries: independent or ordered. The latter means that the LSR waits for the FEC to next hop mapping and the remote label binding information to create its forwarding entries. Independent creation means that the LSR does not need to have the remote binding information to create its forwarding entries. With independent creation, the LSR creates the local label binding in response to the receiving of FEC to next hop mapping and advertises it to other LSRs (thus providing the remote label binding). With ordered creation, even when the LSR has created its local label binding, it has to wait until it receives the appropriate remote label binding before advertising its local label binding.

The choice between independent and ordered requires some remarks. First, ordered creation means latency. The construction of forwarding entries are serialized among a set of LSRs while in the independent case, creation occurs in parallel with others LSRs. Second, ordered creation raises some interoperability problems since there exist dependencies between LSRs. This affects the robustness and scalability properties[4]. Finally, ordered creation has one advantage in that it simplifies configuration efforts. If one wants to restrict the FECs that are label-switched, he has to configure only a subset of the LSRs about which FECs to label switch. On the other hand, one has to configure all LSRs in the independent case.

[4] Independent creation minimizes interdependence.

Chapter 3

Multi Protocol Label Switching

3.1 Introduction

While the IP Switching approaches described in Appendix 4 tackle the IP/ATM integration, they "only" represent proprietary solutions that did not really intend to become long-term solutions for IP over ATM. A true standard and open solution had to emerge from all the concepts gathered from the previous approaches: it is MPLS. The interest formulated by several companies, not to mention Cisco's intention to pursue its standardization efforts of label switching[5], have without a doubt been an important motivation for the creation of the MPLS working group at the IETF. The MPLS Working Group aims at developing a solution that must work with existing datalink technologies based on high level requirements. According to the MPLS Working Group charter (see [MPLSCH]), the group is responsible for standardizing a base technology for using label forwarding in conjunction with network layer routing over a variety of media. The objectives are to develop a unicast Label Distribution Protocol as well as a Multicast LDP, Operation over ATM, Encapsulation and finally Host Behavior. Several documents describe what MPLS is all about as well as its architecture and its protocols. The content of these documents is described in the remainder of this chapter. We present in this section an overview of these MPLS drafts. The MPLS Framework document explains what MPLS is all about and sets the terminology used in the other documents from the MPLS working group. It explains the concepts of IP switching which we have previously discussed (label distribution issues, interoperability matters...). The MPLS Architecture document describes the parts of the protocols on which the Working Group came to a consensus. The Label Distribution Protocol document specifies the procedures used by the LSRs to communicate their label bindings. Note that these documents are to be considered as "work in progress". They should not be used as reference since their content might vary considerably. We will give as much insight as possible about MPLS so that the reader will have an accurate picture, without going into the technical details, since a consensus has not been found yet. For the reader further interested in specific details, see [IETF]. The description given here should reflect the state of the work of the MPLS working group as presented in the Internet-Drafts up to early 2000.

3.2 Motivations

3.2.1 MPLS Requirements

As stated in [MPLSFR]: "The primary goal of the MPLS working group is to standardize a base technology that integrates the label swapping forwarding paradigm with network layer routing. This base technology (label swapping) is expected to improve the price/performance of network layer routing, improve the scalability of the network layer, and provide greater flexibility in the delivery of (new) routing services (by allowing new routing services to be added without a change to the forwarding paradigm)". This means no less that the objectives are clearly the standardization of a new technology aimed at integrating all existing datalink layers with current layer 3 routing protocols. MPLS is thus due to be the "most general unifier" for layer 2 and 3 routing paradigms. The requirements for MPLS design are high-level properties the MPLS standard must verify.

[5] The first Tag Switching Internet-Drafts were published at that time.

The remainder of this chapter discusses the features and some expected benefits of MPLS. The following benefits only relate to ISP backbones and major corporate networks. Campus and LAN networks are out of the scope of this dissertation.

3.2.2 MPLS vs. Layer 3 Routing

MPLS allows for a more simple forwarding because it uses label swapping and labels that are simpler than typical layer 3 headers. This does not imply that MPLS will allow for higher speed than classical routers: implementation details will decide about it. While layer 3 routing supports explicit routing, its overhead makes its use prohibitive. This situation does not arise with MPLS. The explicit route information need not be carried in every packet but only at the establishment of the label switched path (LSP). A LSP may be compared to a virtual connection between two LSRs. Once the LSP established, every packet of the FEC follows the explicit route without the need for explicit information in the packet header. The main advantage from the use of MPLS probably relates to its ability to become a powerful tool for traffic engineering. Traffic engineering is about being able to balance the traffic load within the network. It allows selecting the paths followed by data traffic on the various links, routers and switches of the network. Traffic engineering encounters problems in our current IP networks because one often needs to manually configure the link metrics in order to balance the traffic on the multiple available parallel paths. Such a situation does not scale with big backbone networks where the traffic variability may turn out important. MPLS allows identifying and separately handling individual streams of packets. It provides a straightforward means to measure the traffic bounded to a specific $<ingress\ LSR, egress\ LSR>$ pair. This provides MPLS enough information to compute the best path a new flow should follow across the network to ensure an adequate traffic distribution. The challenge of traffic engineering amounts to choosing an appropriate technique (manual configuration, use of existing routing protocols or implementation of a dedicated protocol) to route the LSPs.

Another similar challenge is QoS routing. QoS routing means choosing a path across the network that guarantees one or several properties of the path (bandwidth, delay, jitter...). QoS routing may be seen as an extension of traffic engineering where paths are constrained not only by the overall traffic distribution but also by path properties. QoS routing requires more complete information about network link state than traffic engineering because QoS-related link state information might become very quickly obsolete. QoS routing and traffic engineering have similar information needs while distinct objectives. Traffic engineering is about network use and cost reduction while QoS routing aims at providing strict guarantees to flows. The advantage of MPLS for QoS routing relates to explicit routes: one can determine where resource shortage occurs in the network so that "backtracking" procedures (crankback) will find another path that satisfies the flow needs. Such information is not known in IP networks since only node reachability information is available, at least at the interdomain level. Support for complex "IP to FEC" mapping constitutes another advantage of MPLS: packet filtering occurs one single time at the ingress LSR. Complex packet filtering is impractical in today's IP networks because of the need to perform filtering at each node the packet pass through. The burden it would imply on routers seems too heavy for now. The partitioning of functionality between border and non-border LSRs gives the opportunity to spread more intelligently the load of complex tasks in the network. By making a lot of the work in the border of the network, non-border LSRs concentrate on forwarding functionality while border LSRs spend relatively more time on route calculation and packet filtering.

3.2.3 MPLS vs. IP over ATM

One of the main drawbacks of the IP over ATM approach is its lack of scalability. IP over ATM often requires $O(n^2)$ logical links between switches due to the connection-oriented nature of its layer 2 implementations. Since LSRs run standard IP routing protocols, the number of peers one LSR need to communicate with is limited to the LSRs directly connected to it. Another point in favor of MPLS is its

RSVP) and the creation of new protocols dedicated to that task (LDP and CR-LDP). LSRs may be label distribution peers for some part of their respective label space but not for some other part of it.

LSRs have two alternatives for what concerns label binding distribution: either they explicitly request a binding from their next hop (downstream-on-demand distribution) or they receive it from adjacent LSRs without having requested it (unsolicited downstream). LSRs are not required to implement any particular label distribution method as long as LSRs agree with their adjacent peers. Some LSRs will provide only one method while others could cope with both of them.

3.3.4 Label Retention Mode

There are two choices for a LSR when receiving a label to FEC binding from an adjacent LSR which is no more its next hop for the FEC: either discard the binding information (conservative label retention mode) or keep track of such binding for the eventual case where the LSR that sent it would become its next hop for that FEC (liberal label retention mode). The trade-off between the two methods is the following: liberal retention mode allows for quicker adaptation to routing changes while conservative label retention mode requires maintaining fewer label bindings.

3.3.5 Label Encoding

Two options may be used to encode the label stack and other MPLS control information: either define a separate protocol between the network and data link layer ("shim header") or use the existing possibilities provided by the underlying data link technology. In the first case, the shim header must be "protocol-independent" so that it may be used to encapsulate any network layer protocol. This "generic MPLS encapsulation" is defined in [MPLSSHIM]. The existence of a shim header also implies specific MPLS hardware or software. If label encoding in the data link layer technology is used, the encoding technique will depend upon implementation details of the particular datalink layer. MPLS should allow interoperation between the various encapsulation techniques even if some data link layer technologies are know to be quite restrictive about their ability to interoperate with others technologies.

Figure 3.1 shows the MPLS label stack encoding. The Label field contains the actual value of the label. The Exp field is reserved for experimental use. The S bit indicates the bottom of stack when set to zero. TTL encodes the time to live.

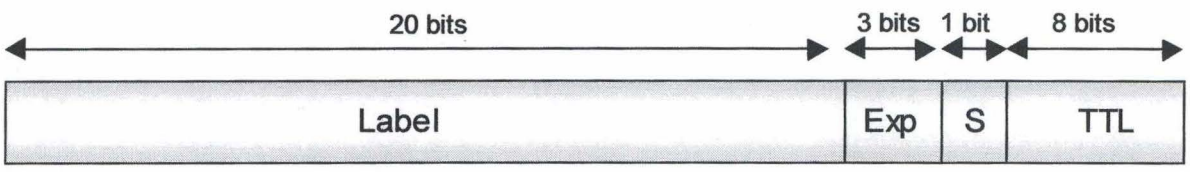

Figure 3.1: Label stack entry encoding

3.3.6 Label Stack

The designers of MPLS made a clever implementation choice by defining a label stack. Labels in MPLS are structured as a stack. It gives the opportunity to assign distinct labels for every level of the routing hierarchy that a labeled packet goes through. The presence of a label stack has no significant implication on the processing of the packets. The processing of a packet is based on the top level label indiscriminately of the past existence of labels on top of it as well as on the fact that several labels could be present above it. An unlabelled packet is a packet with a stack of depth 0 (so IP routing is a

relative independence with regard to the underlying datalink layer. This independence with respect to the layer 2 technology is also expected to simplify network management.

3.3 Architecture

We have seen so far requirements, motivations and potential benefits of MPLS. What about the real-life MPLS? We have had a flavor of MPLS in the previous sections: let us see what there really is in it!

3.3.1 Introduction

As previously stated every packet that enters an MLPS network is assigned to a particular FEC by means of a label sent along with it. When the packet travels across the MPLS network, the value of the label serves every LSR as an index into a table, which specifies the next hop and a new value for the label. The old value is then replaced with the new one so that the packet can be forwarded to its next hop. Once the packet has been assigned to a FEC, subsequent LSRs along the LSP will make no further analysis of the packet. An interesting point to note is that the assignment of the packet to a particular FEC can be based on any information available to the ingress LSR. The IP header, the transport protocol header, the data content of the packet or even information exterior to the packet[6] might be relevant to determine the particular FEC. This gives MLPS the ability to base the packet to FEC mapping on any level of granularity. Hence, the level of complexity of the mapping operation does not affect intermediate LSRs. Thanks to its high-level definition, the FEC concept may allow to express any conceivable routing constraint.

3.3.2 Labels

A label is a short, fixed length, locally significant identifier that identifies a FEC. The decision process by which a mapping between a packet and a FEC is made is said dynamic. The same packet might appear at the same ingress node at two distinct moments and receive different mappings due to different network state conditions. A topological or even a policy-related change may make the treatment received by the same packet at two distinct moments different. The value of the label may rely on a local or partially local decision because any characteristic of the FEC cannot be inferred solely based on the value of the label. The value of the label does not contain any semantics. A packet that was given a label's value is called a "labeled packet". Note that the label may be encoded in an encapsulation header for that specific purpose or in an existing layer 2 header (ATM, Frame Relay...). The way it is encoded does not matter as long as the sender and the receiver of the labeled packet do agree on the particular encoding technique.

3.3.3 Label Distribution

This label distribution mode used in MPLS is downstream relative to the concepts defined in Chapter 2. A particular label binding may have associated attributes. These attributes may be distributed between LSRs under certain conditions. A label distribution protocol is a set of procedures by which LSRs can communicate their respective label bindings. LSRs that exchange their label bindings are known as "label distribution peers" (we will use the shorter term "adjacent LSRs" in our discussion). The MPLS architecture does not assume the existence of one single label distribution protocol. Two distinct means are possible to distribute the label bindings: piggybacking on existing protocols (BGP or

[6] Like some QoS-related state of the network...

particular case of MPLS forwarding). Figure 3.2 gives an example of a packet with a label stack of depth n (with n > 3). Level 1 label is the one closest to the data. If the label stack is of depth m (m > 0), there are m-1 labels between the data and the top level label.

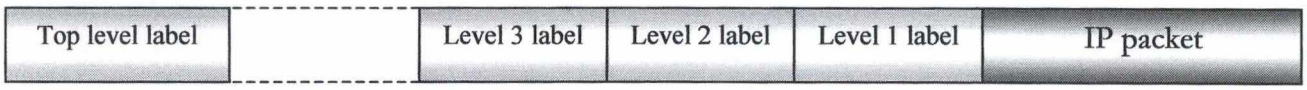

Figure 3.2: MPLS label stack

3.3.7 Forwarding

When a packet arrives at an input port, the forwarding decision exclusively relies upon the label at the top of the stack (if such label exists). The LSR must know two things in order to forward the packet: what is the next hop and what to do with the label on top of the stack (if there is one). The purpose of the Next Hop Label Forwarding Entry (NHLFE) is to provide this information. It contains the next hop of the packet and the operation to perform on the label stack of the packet.

The operation on the label stack consists in:

1) Replacing the label at the top of the stack with a new label or

2) Popping the label stack or

3) Replacing the label at the top of the stack with a new label and pushing one or more new labels onto the label stack.

Case 1 corresponds to the simple forwarding on the LSP so only the value of the label on top of the stack changes. Case 2 corresponds to the arrival of the packet at the egress endpoint of the LSP of depth equal to the depth of the popped label(s). The LSR that pops the label(s) is the egress LSR for the corresponding FEC. Case 3 arises when the LSP traverses additional levels in the routing hierarchy. A particular situation arises when the next hop for the FEC is the LSR itself. In that case, the operation must be to pop the stack.

If we had only one NHLFE per FEC, it would not be possible to use multiple paths in order to balance the traffic. The "Incoming Label Map" (ILM) makes the mapping between each incoming label and a set (containing one or more elements) of NHLFEs. When a packet arrives unlabeled, it cannot be attributed a NHLFE. Unlabeled packets use the "FEC-to-NHLFE" (FTN) to obtain a label stack before being forwarded. The FTN maps each FEC to a set of NHLFEs in the same way that the ILM does for a label.

The forwarding process of an unlabeled packet thus consists in the analysis of the network layer header in order to determine the FEC of the packet. The FTN is then used to find an NHLFE. The NHLFE gives the information of where to forward the packet and what operation to perform on the label stack (that does not exists yet). The operation cannot consist in popping the stack but only in either pushing one or more label(s) or simply forwarding it without pushing any label.

The forwarding process of a labeled packet begins with the examination of the label at the top of the stack. The ILM allows mapping this label to an NHLFE. The LSR determines based on the NHLFE where to forward the packet and the operation to perform on the stack before forwarding the result.

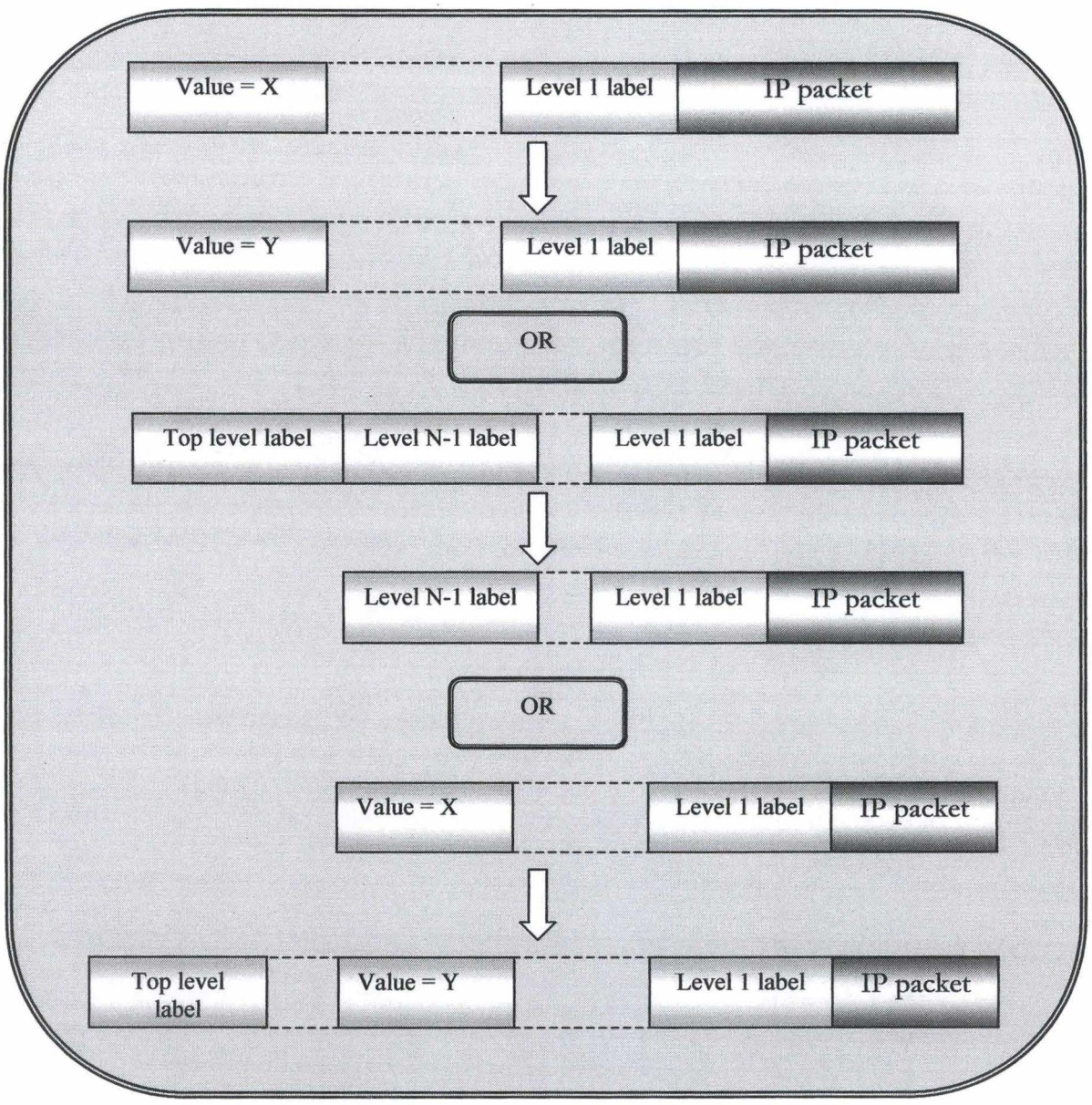

Figure 3.3: MPLS forwarding operation

3.3.8 Label Switched Path

We describe in this section the actions performed by LSRs on a path taken by a labeled packet, with and without stacking. A label switched path is an ordered sequence of LSRs $<R[1],\ldots,R[n]>$. The depth of the stack evolves according to the number of levels in the hierarchy the packet goes through. The more levels in the hierarchy in the actual portion of the LSP the packet is traversing, the more labels there are on the stack. Let us begin with a LSP across a flat hierarchy: At all steps between R[1] and R[n], the stack has depth 1 and the only action performed by intermediate LSRs is label swapping. Only the value of the label changes from one LSR to another. Figure 3.4 illustrates the evolution of the "stack" from R[1] (ingress LSR) to R[n] (egress LSR). Before the ingress LSR, the packet is unlabeled. The ingress LSR has to push a label on the empty stack and forward the labeled packet to the first intermediate LSR. From the first intermediate LSR to the egress LSR (or the one just before, see section 3.3.9), label swapping is performed at each step. Only the value of the label changes. At the egress LSR, the stack is popped. Forwarding is then based on the network layer header.

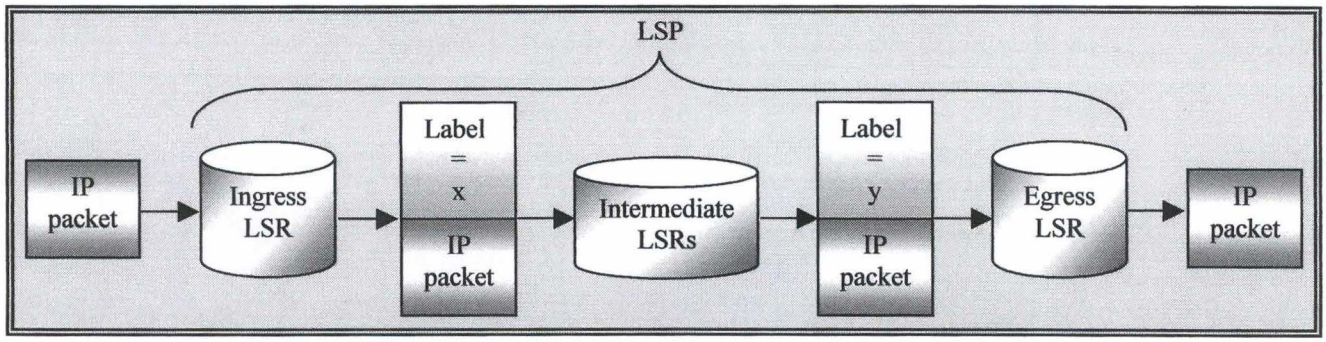

Figure 3.4: LSP without stacking

Suppose now that stacking is in use, by example the sequence <*R[2],...,R[n-1]*> corresponds to one more level in the hierarchy. In that case, we have two LSPs between R[1] and R[n]. The level 1 LSP is <*R[1],R[2],???,R[n-1],R[n]*> where "???" means that the packet enters an area where stacking is in use. The level 1 LSP as no idea about what happens between R[2] and R[n-1]. "???" is similar to a tunnel packets enter at R[2] and leave at R[n-1]. Only LSRs R[1], R[2], R [n-1] and R[n] see the level 1 label since a level 2 label is used between R[2] and R[n-1]. At every step between R[2] and R[n-1], the stack depth is at least of two since other levels could be traversed. Figure 3.5 shows the evolution of the stack from ingress R[1] to egress R[n].

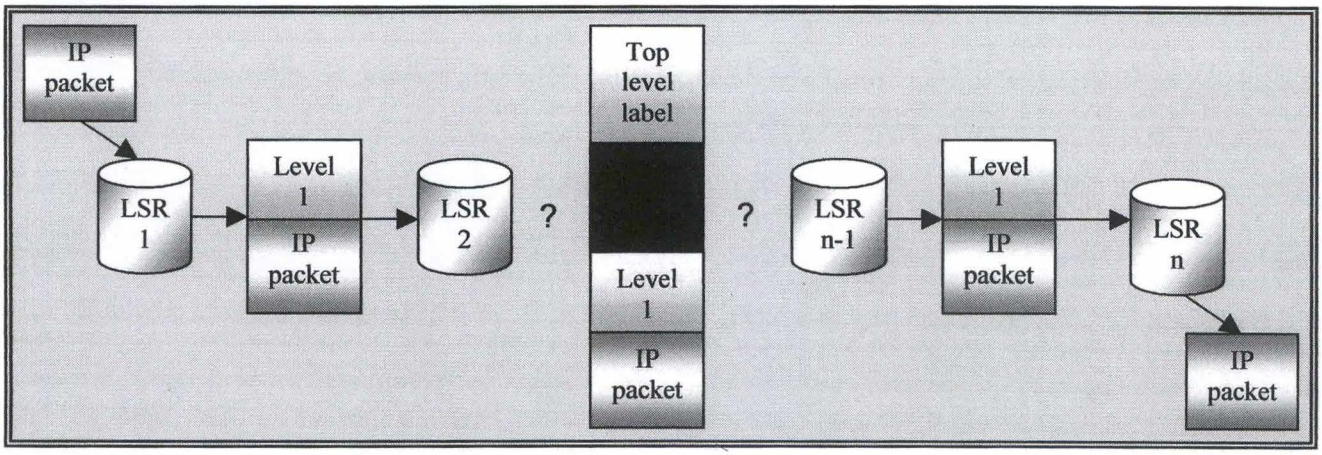

Figure 3.5: LSP with stacking

3.3.9 Penultimate Hop Popping

The reader may have noticed that the last section allows LSR R[n-1] to forward labeled packets with a stack of depth $m-1$ if the LSR has a depth of m. The rationale for penultimate hop popping is that once R[n-1] has decided to send the packet to R[n], the top label no longer has any function. The main advantage of penultimate hop popping is practical: it spares a label lookup at R[n]. Without this scheme, R[n] would have to look up at the top label to determine that he is the egress LSR for the level 2 LSP. After that, it would have to pop the stack and examine the top of the remaining stack to forward the packet. If the stack is empty after the popping operation, the forwarding decision is based on the lookup of the network layer header. On the other hand, when penultimate hop popping is used, R[n] needs only one lookup to forward the packet based on either the top label or the network layer header. Another non-negligible advantage of the scheme concerns code simplification. An LSR can assume that only a single lookup is ever required (not every LSR may be able to implement penultimate hop popping but some devices could get cheaper thanks to this method). The issue for penultimate hop popping will be to make an LSR discover whether its neighboring LSRs implement it or not.

3.3.10 LSP Control Issue

The reader knows from the previous chapter that two means exist to control label binding distribution: independent and ordered. The issue with the independent scheme comes from the fact that each LSR must rely on the appropriate (on a time basis) convergence of routing information to ensure effective delivery of each datagram. Because LSRs may create and advertise label bindings whenever they want to, synchronization with routing information might create oscillations problems within the label to FEC mappings. Such problems relate with the non-deterministic nature of the scheme. Ordered control on the other hand allows a LSR to bind a label to a FEC if and only if it has already received a label binding from its next hop(s) (for that FEC), except when the LSR is the egress LSR of the LSP. Both schemes ensure that traffic in a particular FEC follows a path that owns the desired properties[7]. This happens because the complete LSP must be established in order to be able to provide strict QoS. The two schemes might cooperate even if the overall behavior of the network will not be the ordered-like one as long as not every LSR uses the ordered scheme. The choice between independent and ordered is only implementation-dependent since both may provide the same guarantees. The main difference might reside in the convergence time for LSP establishment.

3.3.11 Aggregation

An important issue we had to cope with in IP routing was the growth of the routing tables due to the increasing number of networks in the Internet. The Internet growth will not stop with MPLS so the designers tackled this problem early in the MPLS development cycle. MPLS allows several FECs to be aggregated into one or several labels. Aggregation gives MPLS a wide scope to the granularity of a FEC. However, the advantages provided by aggregation do not come easily: each LSR wishing to aggregate traffic flows will be obliged to do it in a coherent manner with its LSR neighboring. When ordered control is used, LSRs should adopt the same granularity as their next hop. The situation differs for independent control: adjacent LSRs may aggregate their FECs differently as long as upstream LSRs FECs use a finer granularity than downstream ones. In the case of upstream having finer granularity, upstream LSRs will request more labels for a given FEC. If an upstream LSR has coarser granularity then it has two options. The first one is not really an option since it consists in adopting the next hop granularity. The second one consists in mapping its less important number of labels into a subset of its next hop's labels if it knows that it will produce the same routing. If it does not produce the same routing, then no other solution will ever resolve the problem anyway. In any case, each LSR will have to discover its neighbor's FEC granularity in order to make this scheme properly work.

3.3.12 Route Selection

Route selection refers to the method used for selecting the LSP for a particular FEC. MPLS supports two different schemes: hop-by-hop routing and explicit routing. Hop-by-hop routing is the method currently in use in IP networks where the routing decision is always a local matter for the current router (the one that has to make the forwarding decision). Hop-by-hop routing encounters huge difficulties when the path followed by a set of packets must own particular properties. Explicit routing allows the path to be completely (every transit LSR along the path) or partially (only the LSRs the path must traverse but there may be other LSRs traversed) selected before the LSP set-up process occurs. The latter scheme opens the doors for many traffic engineering methods including constrained path selection, traffic load balancing and policy routing. Traffic engineering will no more be a configuration game where extreme tuning and experimentation are the only means to make the packets follow a desired path across the Internet. Be careful not to confuse MPLS explicit routing with IP source

[7] For example, bandwidth, delay, jitter, explicitly specified paths or even more large-scale related properties like sound traffic balancing.

routing: IP source routing requires every packet to be processed by every router along the source route. MPLS explicit routing is like IP source routing but only at LSP establishment time. Once the LSP established, LSRs along the path label swap the packets.

3.3.13 TTL

The issue of the Time-To-Live field reappears with MPLS: every technology that tries to bypass the hop-by-hop routing scheme has to cope with it. Since the purpose of the TTL field is quite wide, MPLS needs to cope with TTL as a means for loop handling as well as a way to accomplish other functions (like multicast scoping and support for *traceroute*-based applications). A packet traveling along a LSP must emerge at the egress LSR with the same TTL value as if it had traversed a hop-by-hop routed path. Each LSR should be considered like an ordinary IP next hop. The encoding of the TTL field in MPLS will depend upon the use of a shim header for the MPLS header. If a shim header carries the MPLS information then it should contain a TTL field that every LSR would decrement along the LSP. On the other hand, if the MPLS information is encoded in a datalink layer header that does not explicitly contain a TTL field, a means must ensure the propagation of the path length information to the ingress node(s). Hence, an ingress node should be able to decrement the TTL from the IP header before inserting the packet in the LSP.

3.3.14 Label Merging

Label merging is the mechanism by which several incoming labels are bound to a single outgoing label. This can happen for instance when a particular LSR uses a wider granularity than upstream LSRs. If an LSR cannot perform label merging, two packets arriving with different incoming labels must be forwarded with different outgoing labels. This situation is not desirable in large networks since the number of outgoing labels per FEC could be as large as the number of nodes in the network. With label merging, the maximal number of incoming labels per FEC that a particular LSR needs is equal to the number of label distribution adjacencies. Label merging is not mandatory so interoperability between merging and non-merging LSRs is an issue. Label merging would not be an issue if packets were never fragmented since packet-fragments interleave would never happen. Two solutions have been proposed: the first is the support for non-merging LSRs while the second consists in procedures that allow a non-merging LSR to function as a merging LSR. In any case, LSRs should discover their adjacent LSRs merging capabilities by configuration. Nevertheless, there exists methods of eliminating cell interleave for the ATM case (see [MPLSFR]).

3.3.15 Label Distribution Peering

While MPLS allows for the existence of several levels of hierarchy, it must cope with means to keep label peering consistent with that hierarchy. MPLS supports two methods to distribute labels between peers: implicit and explicit peering. LDP peers that participate in the same IGP peering session are called "local label distribution peers". LDP peers that do not participate in the same IGP session are called "remote label distribution peers".

Explicit peering is about distributing label distribution protocol messages by sending them explicitly to the peers, i.e. by using the known address of the peer. This approach is best if remote LDP peers are few, or the number of higher-level label bindings is large or when remote LDP peers are located in different routing domains.

Implicit peering does not assume the knowledge of the address of the remote peers. Instead, higher level labels intended to remote peers are encoded as attributes of a lower level label. The local LDP

peers then distribute both the local and remote label binding information. This technique does not require an $O(n^2)$ peering mesh between remote peers thanks to the piggybacking approach. In return, intermediate nodes need to store some more information.

3.3.16 Label Distribution Protocol Matters

We said in section 3.3.3 that more than one solution to distribute label bindings was conceivable. One could ask: "Why more than one LDP? Is it not enough having just one LDP do the job?" The answer is no. The problem is that there are no universal rules that will decree the enforcement of one best method for every possible situation. There always exists a trade-off when choosing among several "least worst" solutions for the particular situation.

When only one standard routing algorithm distributes the interdomain routes, the best means to distribute label bindings consists in piggybacking it on the route distribution protocol[8]. On the other hand, for intra-AS routes, piggybacking on top of BGP cannot be used so LDP is the only means to distribute the routes within an AS (neither OSPF nor IS-IS do not allow to distribute labels).

In our traffic engineering context, explicitly routed paths will often require resource reservation. However, we assume that resource reservation will be done everywhere in the network. This is a strong hypothesis in our best-effort Internet. Consequently, either we start with RSVP and add support for explicit routing (see [MPLSRSVPE]) or we make use of an existing protocol for label distribution and add support for explicit routing and resource reservation (see [MPLSCRLDP]).

3.4 MPLS Nice Features

3.4.1 Hop-by-Hop Routing

The connection-oriented nature of MPLS requires the partitioning of all packets into several classes. All packets within a class receive the same treatment along their respective LSP. What property will be used to determine the different traffic classes? The actual property used today in the Internet to route packets is the address prefix of the destination. Routers determine the next hop for each packet by looking up in their routing table and finding the best-matching (longest match) entry for the particular destination address. Actual Internet FECs consequently correspond to all traffic which destination matches a longest-known prefix. The prefix length could eventually be zero. This means that no prefix is known for that destination (default routing). Such a situation may happen in small networks that do not participate in an Exterior Gateway Protocol peering session with other ASs. Default routes are uncommon in backbones or big ISPs routing tables. Today's metrics in use for routing is restricted to a "hop count". The only objective achieved by today's routing is to make the number of hops decrease towards the destination. This situation is clearly not suitable in order to achieve good network utilization. MPLS allows bypassing this stupid routing metric through explicitly routed paths. Figure 3.6 shows it for only one particular target prefix. In reality, LSRs should bind one or more labels to each address prefix appearing in their routing table and use a LDP to distribute the binding of a label to each of their LDP peers for that particular prefix. Given that a routing table contains in the order of tens of thousands in the number of entries, maintaining one LSP per entry seem impractical for interdomain routers[9]. Aggregation techniques such as topology-based multicast-like trees or traffic-based LSPs could resolve this issue. When such an hop-by-hop path is used as the LSP, care must be taken that this LSP

[8] BGP in the Internet today (see Appendix 6).
[9] See Chapter 5 for a quantitative evaluation of interdomain prefix variability.

can extend only to the point where a new "better" best-matching entry is found for the FEC. Aggregation is only possible up to the point where a "better-match" is found.

Figure 3.6 shows a situation where LSRs 1, 2 and 3 have LSR 4 as their next hop for prefix X. LSR 5 is the next hop for X for LSRs 1, 2, 3and 4. LSR 8 is next hop for X for LSRs 5, 6, 7 and 8. This situation illustrates prefix targeted hop-by-hop routing since an LSP traverses LSRs 4 and 5. The LSP for destination X extends across LSR 4 but a new LSP is established at LSR 8 since he has a better match for prefix X. We see that the LSP from LSR 4 could extend to the destination but a better match at LSR 8 obstructs the LSP establishment.

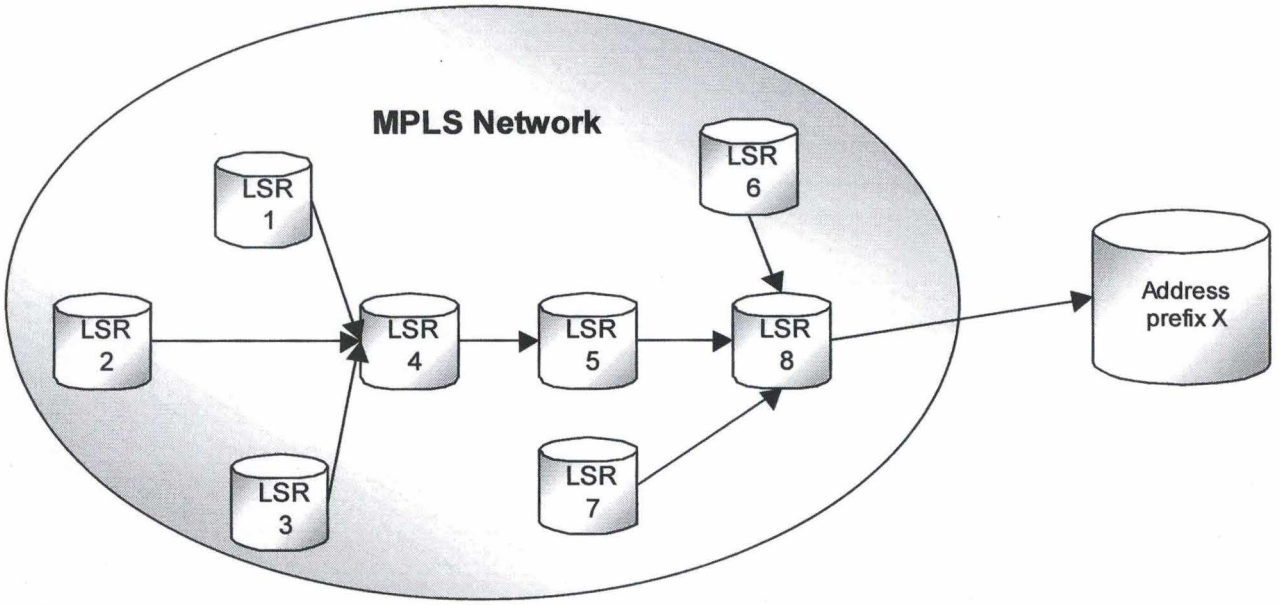

Figure 3.6: MPLS hop-by-hop routing with label assignment to address prefixes

3.4.2 Egress Targeted Label Assignment

There exist situations where an LSR Ro knows for whatever reason that packets of different FECs must follow the same LSP terminating at a known point Rc. In such a case, the best solution consists in routing all the FECs with a single label since distinct labels are not necessary. However, several conditions are required in order to achieve proper functioning of this scheme: the address of Rc is present in Ro's routing table and Ro must be able to determine that Rc is the egress LSR for all the FECs. If Ro binds a single label to all these FECs, we say that Ro performs "Egress-Targeted Label Assignment". Quite spectacular, isn't it? However, how might Ro gather that information, assuming that Rc is not just Ro's next hop for all these FECs?

There exists numerous ways by which this is possible:

- If the network is running a link state routing algorithm and all nodes support MPLS, the routing algorithm provides enough information to determine through which routers which packets of the FECs must leave the routing domain;

- When BGP is used, interior nodes may be able to determine that certain packets of a FEC must leave the network via a particular router (the BGP next hop for that FEC);

- One could think about a method that would provide the LDP enough information so that every LSR would be able to determine which prefixes are bound to which egress LSR (without requiring any link state information).

3.4.3 Explicitly Routed LSPs

We shall see in the next chapter the motivations for explicit routing. Many reasons may require the use of explicit routing instead of hop-by-hop routing. Policy routing and traffic-engineered routes are examples of such situations. Network administrators may want to forward specific traffic classes along specific and pre-specified routes that differ from the ordinary hop-by-hop path. The route might be manually configured as well as dynamically calculated. MPLS could perform these tasks. All it needs is:

- A means of selecting the packets that are to be sent into the explicitly routed LSP;
- A means of setting up the explicitly routed LSP;
- A means of ensuring that loops will not occur in the explicitly routed LSP.

3.4.4 Multi-Path Routing

An LSR could assign multiple LSPs for a particular stream. If so, it may assign multiple labels to the stream (one for each LSP). The reception of a second label binding from a neighbor for an already bound address prefix should mean that both labels represent that address prefix.

3.4.5 LSP Tunneling

Assume that an AS X carries transit traffic between other ASs. AS X possesses BGP border routers with a mesh of BGP peering sessions among them. We clearly do not want to distribute every route known by border routers to non-border routers since it would represent a tremendous burden. However, we also need that transit traffic be delivered between border routers. A solution would be to establish LSP tunnels according to the following rules:

- Each BGP border router distributes to every other border router within the same AS a label for each address prefix that it advertises to that router via BGP;
- The IGP used in AS X maintains a host route for each BGP border router and distributes its labels for these routes to each of its IGP neighbors;
- When a BGP border router receives a packet, it forwards it by means of a level 1 label (by changing the value of the top label) to the BGP next hop border router. A level 2 label allows the packet to traverse the tunnel across AS X's IGP routers.

Since BGP border routers exchange label bindings for address prefixes that are not known to the IGP routing, BGP border routers should become explicit label distribution peers. We can therefore say that hop-by-hop routed LSP tunnels exist between the BGP border routers.

3.4.6 VPN

A "Virtual Private Network" (VPN) is the abstraction of a user-defined interface onto a physical infrastructure. It defines a virtual topology between distinct sites so that every site has logical connectivity to the others sites of the VPN. The user located inside a site of the VPN sees the topology as if it were directly connected to every other site belonging to the VPN. In order to define a VPN, one needs to provide administrative mechanisms to designate members of the VPN. The membership of a site between several VPNs is not exclusive. A particular site or a subset of it may belong to several VPNs at the same time. This connectivity is implemented either through a full mesh (direct route between each site) or a partial mesh (only a subset of the sites have a direct route between them). The question of the responsibility of the VPN service is an open one. Some customers will want to get a full VPN service from their SP so that they will not be obliged to care about the policies that determine whether some sites belong to the VPN. This solution implies that the VPN connectivity should not change too often. Since these policies are to be provided by the customer presumably on a dynamic basis, the most realistic solution would be to share the responsibility of the implementation between the customer and the provider.

Several potential solutions appear to implement VPNs on MPLS. We will discuss two current propositions; the first uses BGP with MPLS (see [VPNBGP]) while the other only MPLS (see [VPNMPLS]).

The BGP/MPLS VPNs approach makes use of MPLS to forward packets and BGP to distribute the routes. The model assumes that each site of a VPN has one or more Customer Edge (CE) devices attached to one or more Provider Edge (PE) router. A PE router attaches to a particular VPN if it connects to a CE of at least one site belonging to the VPN. It attaches to a site if it connects to one CE of the site. CE routers at different sites do not exchange directly routing information. Instead, each PE router maintains one or more "per-site forwarding tables". The PE binds one such forwarding table with a particular site to which it attaches. If a site belongs to multiple VPNs, the forwarding table associated with the site will contain routes from all VPNs of which it is a member. When a packet coming from a site arrives at a PE router attaching to the site, the PE router looks up in the forwarding tables associated with the site. Not finding a matching entry means that the destination of the packet does not belong to any of the site's VPNs. If the SP provides Internet service, the packet will be routed by means of the Internet forwarding table of the PE router. If the site has not contracted any Internet service, the packet is discarded. Since a VPN emulates an internet, host IP addresses are unique within a particular VPN. The same IP address may be used within several distinct VPNs. The way BGP distributes the routes is out of the scope of this dissertation. For the reader further interested in BGP/MPLS VPNs, see [VPNBGP].

The second approach envisions a VPN service only by using MPLS based on the concept of a "virtual router". This approach does not require any modification of any existing routing protocol. A "virtual router" is a collection of threads running in a routing device that provides routing and forwarding services. A virtual router is logically equivalent to a physical one for the VPN customer standpoint. The virtual router allows implementing separate routing domains for each VPN. The main advantage of this approach is the ability to provide a flexible service without any hardware requirement. The virtual router is similar to the PE in the previous approach.

Figure 3.7 illustrates the virtual connectivity of a VPN provided over a public network. The way connectivity occurs between every site pair is not important at a user's point of view as long as communication within the VPN is secure. Virtual links across the public network could be implemented by MPLS LSPs or IP tunneling without any difference in the way users think about their VPN.

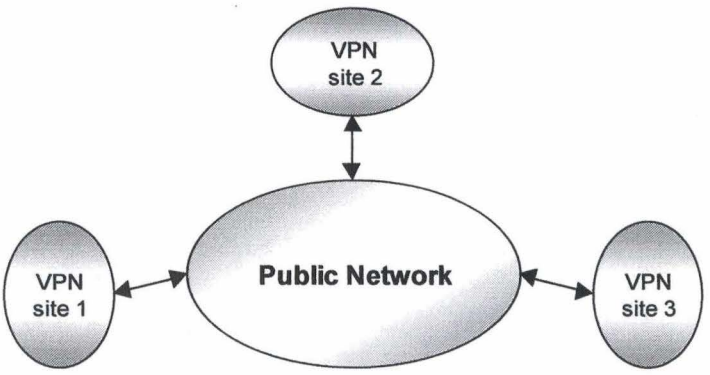

Figure 3.7: VPN over a public network

MPLS is not the only means to implement a VPN; a pure IP solution may be considered (see [VPNIP]) by means of IP tunnels between the sites. Whether MPLS will serve or not as a VPN solution depends on many factors including the demand for VPN service, the speed of MPLS deployment and many technical aspects.

3.4.7 Other Uses of Hop-by-hop Routed LSP Tunnels

Section 3.4.5 covered the specific case of intra-AS tunneling. Hop-by-hop routed LSP tunnels serves all situations where encapsulation tunnels would have otherwise been used. This scheme spares the additional header needed to tunnel a packet across a non-MPLS hop-by-hop routing network. Instead of encapsulating the packet with a new header specifying the tunnel's endpoint as the destination, the label corresponding to the address prefix that is the longest match for the address of the tunnel's endpoint is pushed on the packet's label stack[10]. If the ingress endpoint of the tunnel wants to put a labeled packet into the tunnel, it must first replace the existing top label by a label advertised by the egress endpoint of the tunnel. It then pushes on the label corresponding to the tunnel[11].

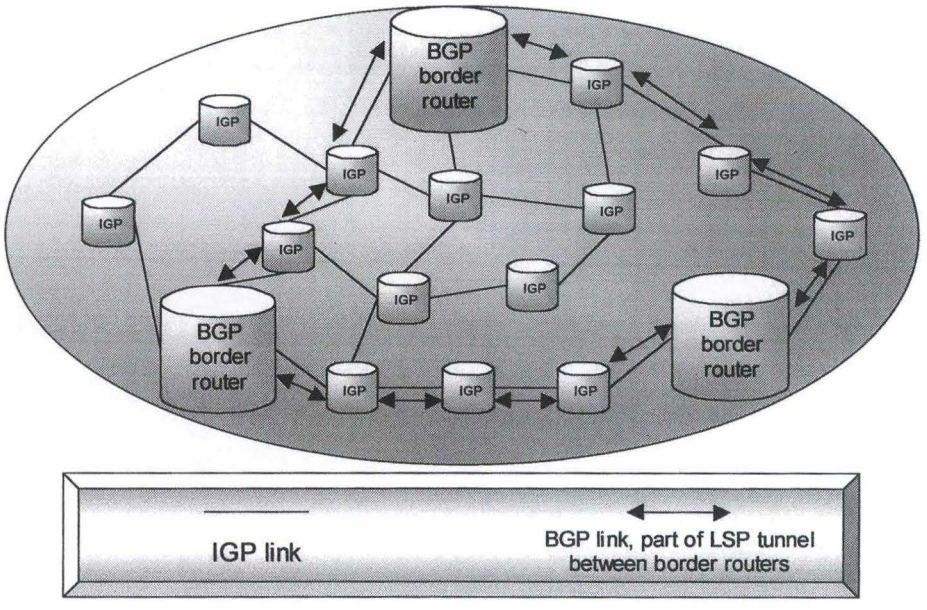

Figure 3.8: Hop-by-hop routed LSP tunnels

[10] Without regard to the fact that the tunneled packet be labeled or not.
[11] The tunnel endpoints must then be explicit label distribution peers.

Chapter 4

Traffic Engineering

4.1 Internet and QoS

While the IP Switching solutions presented in Appendix 4 aimed at resolving the IP over ATM integration, the emerging MPLS standard drastically changed the view people had or will have about routing capabilities. MPLS is not just another IP Switching implementation: it is a standalone technology that enables traffic engineering by integrating label swapping with network layer routing functionality. Since the beginning of the Internet in the early 80's, the clear motivation for IP routing was robustness. With the evolution and growth of the Internet, objectives clearly changed. Robustness is still important today but the core of the Internet problem is no more its ability to survive a nuclear attack. Operational problems like scalability and explicit route selection[12] constitute the chief point. The Internet has evolved into an operational network that needs some (traffic) engineering. The step that will make possible the transition from the "best-effort Internet" to the "QoS Internet" probably corresponds to MPLS even if strict QoS are far from being in sight.

4.2 Definition

Traffic engineering is about performance optimization in real-life networks. It has to facilitate network operation while at the same time optimizing resource utilization and traffic performance. One cannot achieve such objectives without measuring, modeling and controlling the relationship between network traffic and network performance. Traffic engineering arises from the trade-off that must be found between traffic guarantees and network resource utilization. Traffic-oriented performance tries to enhance the QoS of traffic streams (packet loss, delay, jitter...) while resource-oriented performance cares about the optimization of scarce and costly network resources (buffer space, bandwidth...). These objectives are often contradictory since Internet users care about their traffic "efficiency" while ISPs about their network costs. On one side, the user would like to get the most resources on the path to the destination. On the other side, the ISP would like to exactly know the traffic distribution (and being able to control it) in order to avoid too much resource over-provisioning or too many points of congestion. The user does not care about the path taken by its packets as long as performance is acceptable[13]. Apart for link cost reasons, the ISP does not care about traffic distribution as long as it can be predicted and controlled. Thus, traffic engineering must find a compromise between users and service providers objectives. ISPs have few means to control the user traffic before it enters their network. The two main solutions for this problem differ very much. The first one prevents specific traffic to enter the network by discarding the packets or applying a dissuasive billing policy. The second one lets the traffic enter and distribute it inside the network to minimize the likelihood of congestion. The former solution simply solves the problem by suppressing its cause while the latter represents what traffic engineering should accomplish. The classical solution (LAN) network managers apply when faced with potential bandwidth shortage is jerking off more bandwidth in place of trying to balance the load on all available links. Huge over-provisioning is the best-effort solution while waiting for operational traffic engineering tools. It would certainly be less expensive to use all available capacity before thinking about infrastructure upgrade. Even if bandwidth becomes an almost gratuitous

[12] QoS being the "long-term" objective?
[13] Excepted for security reasons; for example VPN traffic.

resource in the future, applications will always manage to consume it so arguing that over-provisioning is the best solution cannot be true. It is admittedly the easiest one!

4.3 User vs. ISP

We saw in the previous section the two players of the traffic engineering game: the user and the ISP. All the approaches we will see in the remaining of this chapter are mostly "ISP-centered". The reader might thus innocently ask: "Why not try to make users and ISPs cooperate?" The following question illustrates the problem of cooperative behavior between users and an ISP: "Could you try connecting later, please? We have currently problems to find a path inside our network for your connection." Do you imagine being said to try connecting later to get your e-mail at your local POP server or while trying to connect at the Internet Gaming Zone[14]? Users want their connections now, not within 5 minutes! Therefore, the only option left is to optimize the ISP's network so that it will accommodate the user's traffic. After all, ISP's are "service" providers so it would seem natural for them to try to provide the best service they can to their users (reality sometimes turn out to be quite different). Of course, an ISP cannot ensure 100 % availability at every moment without over-provisioning. Sometimes, connection requests may be refused to ensure that already established connections are getting their required level of QoS. Depending on the network load, different methods may be used to attain the desired level of QoS. By example, best-effort might work when the load is light. When the load increases, traffic engineering would be used to balance the load so that network utilization is better. However, when network load is high, even traffic engineering cannot compensate for lack of resources. Connection request rejection is the only solution to ensure that already established connections receive a satisfactory treatment.

4.4 Congestion

Congestion problems inside the network arise from two reasons:

- Insufficient resources or inadequate resources to accommodate the demand;
- Inefficient mapping of traffic streams onto available resources (parts of the network are under-utilized while at the same time other parts are over-utilized).

The first source of congestion may be resolved via extension of capacity and/or classical congestion control techniques. Therefore, either we add sufficient resources so that congestion does not appear or we try to control the flows (even if it might be too late in some cases). Classical congestion control techniques include rate limiting, window flow control, router queue management... The second source of congestion arises from inefficient resource allocation. Traffic engineering addresses this kind of problem. Load balancing constitutes an essential component of network performance optimization. It provides a means for minimizing congestion and optimizing resource allocation. By minimizing congestion through efficient resource allocation, packet loss decreases, transit delay decreases and aggregate throughput increases. The perception of network service quality as experienced by the user becomes better due to the smoothing of the traffic distribution inside the network.

Figure 4.1 illustrates the "fish problem": classical IP routing uses shortest path metric in order to get through the destination. Suppose we want to get from device 1 to device 8. For IP routing, paths 3-4-6-8 and 3-5-7-8 have the same metric value if we use the hop count as the routing metric. So, at a particular moment, IP routing will choose between one of the two available paths (depending on

[14] Even if it would be better for your precious time never getting the latter connections.

policing) but only one of them at a particular moment. An important issue concerns route fluttering: whenever congestion occurs on one path (4-6 or 5-7), IP routing will choose the other path and redirect all traffic on this second path. If congestion did occur when routing traffic on one path, redirecting is likely to generate congestion on the other path. This in turn will make IP routing redirect all traffic on the original path.

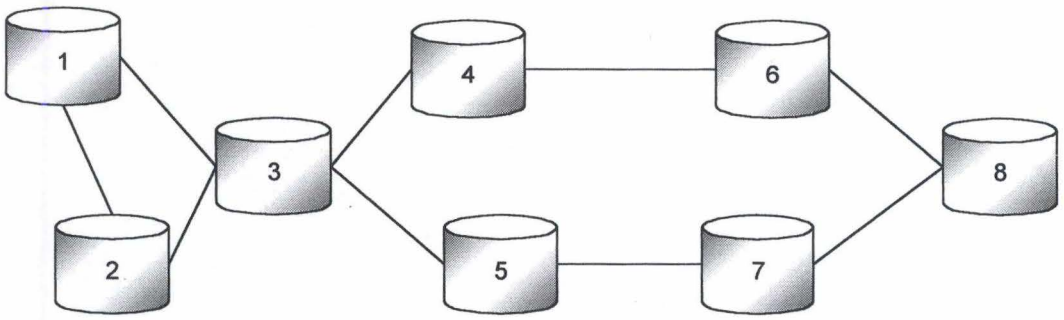

Figure 4.1: The fish problem

One would then find a means to split traffic between 4-6 and 5-7 so that congestion does not occur and both paths utilization is smooth. Unfortunately, load splitting requires some knowledge of the "flows" within the traffic because splitting packets over two paths is going to interleave packets from the same "source-destination flow". In addition, device 3 must have knowledge of the load on both paths 4-6 and 5-7 at every moment so that he can play on the relative load of both paths to minimize variability in path utilization.

4.5 Network Control: Theory

Performance optimization in operational networks is a control-related matter. Trying to preserve the network in a steady and stable state on a time-dependant basis must cope with real-time processes. Since one cannot deterministically model traffic patterns, traffic statistics analysis and near-real-time adaptability are required. Such processes require maximal automation and minimal human intervention. The control process should match the real-time constraints the current traffic imposes and the state of the network one would like to reach. Network state monitoring and network behavior prediction can together achieve that. They might enable reacting to traffic distribution changes within a sufficiently short period. The control process may be abstracted into the following steps:

1. Formulation of a control policy;
2. Observation of the state of the system through monitoring;
3. Characterization of the traffic as seen by the monitoring system;
4. Application of control actions in response to traffic characterization and current state of the system in order to verify the constraints established by the control policy.

Step 1 consists in the definition of the properties that the network must verify at every moment. These properties could be the average link load, the buffer space occupancy within network routing devices or the end-to-end delay between two egress nodes. Any constraint over the state of the network could be part of the control policy. The control policy could be compared to an ideal state of the network (this state could be a function of time).

Step 2 enables the network traffic engineer (be it a human and/or an automaton) to get feedback about the near-real-time state of the system via links state statistics collection or any parameter giving information about the state of the network.

Step 3 translates the monitoring action into a higher-level state system characterization related to the criteria established by the control policy. It provides a quantitative or qualitative difference between the monitored parameters and the state to attain.

Step 4 injects into the system the specification of the actions required to make the system verify the control policy in the future (short and/or long-term). This step also performs the actions corresponding to the aforementioned specification. Remark that these steps form a retroaction (or pro-action) loop. The actions defined in step 4 should either make the system return to equilibrium (retroaction) or prevent the system from leaving the desired state (pro-action), according to the control policy.

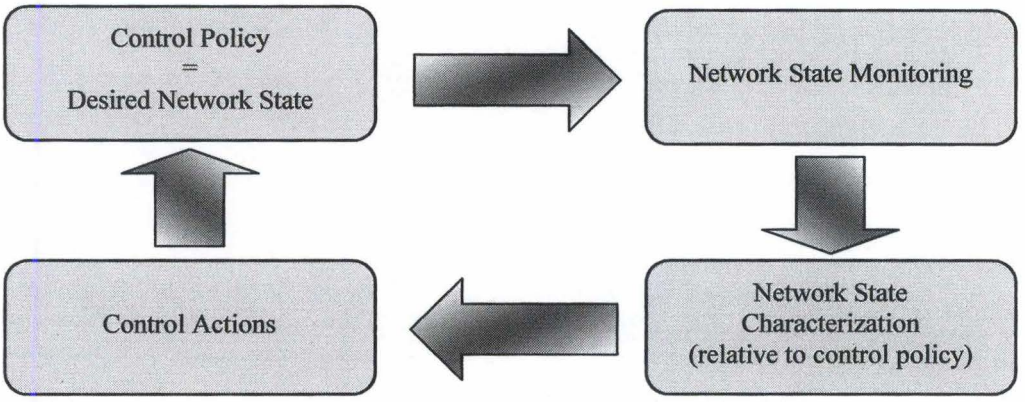

Figure 4.2: Network control loop

4.6 Network Control: Reality

The description of network control we made in the previous section gives a somewhat "idealistic" view from the network control that we have in our real-life networks. We innocently presented the logical steps required for near-real-time network control but we did assume that:

- A formulation of the control policy exists;
- Existing monitoring tools provide means to determine the state of the network at near-real-time granularity;
- Translation between monitoring information and traffic characterization can be made;
- Actions may be performed on the network within a sufficiently short time interval.

Requirements are not achievements! We are in a complex system where not all variables are mastered and where time is an enemy. Getting the information needed to react against a change in the state of the system is not the main concern as long as time does not come into play. Nevertheless, if we need this information in order to react within a minor interval we will not be able to get all the information. If we want all the information, it is going to take time to retrieve. We can get some information within a

short time interval but is it the information we need in order to maintain the system's equilibrium? Not sure at all. Ubiquity and immediacy are not real! One can easily develop network control schemes that are looking good and tasty, but when trying to implement the stuff: what a mess! The logical world of concepts and properties works very well at a theoretical point of view. Unfortunately, once translated into a real system, things are becoming odd. By partitioning network control according to the time granularity at which problems arise, one can use several techniques that fit to the size of the problem. Packet-level issues should not be resolved through huge over-provisioning. At the same time, long-term link load issues should not be resolved only with new buffer management and scheduling techniques.

4.7 Traffic Engineering and Routing

In our definition of traffic engineering, two conflicting aspects seem to oppose. The resource utilization aspect would like to minimize resource consumption by the traffic. On the other hand, traffic would like to use all available resources to attain the destination as fast as possible. One cannot satisfy both wishes at the same time. Hence, as much traffic as possible will be accepted to the point of eventual congestion collapse. Such a situation should (or preferably must) be prevented so that minimal loss occurs. At the same time, the network should accept as much traffic as possible. To achieve this, one logical standpoint would be to let the traffic consume as much resource as it needs as long as congestion does not occur. If congestion happens, balancing the load would allow circumventing the point of congestion. Since congestion did occur, some part of the traffic got lost: this should not occur. If we can prevent any loss, we should do it. The rationale for this comes from the long-term objective of trying to provide the best QoS guarantees to every packet inside the network. Preventing QoS degradation is the best means to ensure the fulfillment of QoS! It is not by trying to compensate for the already lost quality that real QoS can be achieved. Our approach therefore tackles the problem before it arises. When the load is heavy, resources are scarce: try to spare as much resource as you can. When the load is light, resources are abundant: consume them. Routing the traffic inside the network through a least resource-consuming path does sparing resources. Resource consumption on the other hand is "best" achieved by balancing the traffic along all available paths (see [Ma98] for a more detailed evaluation of routing strategies and resource consumption).

Classical IP routing does not allow such routing behavior: only the shortest path known to the destination is used. If a link on the shortest path becomes congested, packets get lost and routing does not care about it. This is one reason why retransmission procedures are included within transport protocols in order to provide delivery guarantees. If the link goes down, routing will find another shortest path to the destination. Nevertheless, routing protocols convergence times are not short enough to ensure that an alternate path will be found. At least not before a quite important amount of packets are lost.

4.8 Service Models

While we covered the "essential" control aspects of traffic engineering, network control is of no use alone: the traffic engineering objective relates to network utilization optimization but not just for the sake of it. The motivation for traffic engineering is traffic, not engineering. Traffic has inherent characteristics that closely relate to the profile of the application using it. Since we do not intend to present a taxonomy of the various applications, we will use the service definition standpoint. This standpoint permits to determine the type of guarantees the network has to provide given that network control will use this standpoint to perform its operations. The first step towards QoS guarantees is the definition of service models. Service models specify the service one can expect from the network considering the behavior of the network as a black box. In order to provide the service to its customers, the network service provider will implement different mechanisms in its network nodes, by

defining the router's behavior. Several mechanisms permit to achieve an identical service. There are roughly three types of guarantees.

The first is the one we have today in the Internet: almost no guarantee! The source sends packets and they magically appear at the destination (in the best case). If the packet does not arrive at the destination, this is what best-effort is all about. It has been lost somewhere between the source and the destination for an unknown reason[15]. The only guarantee users get is that every packet is treated as soon as possible depending on current network load.

The second is Differentiated Services. Differentiated Services defines a per-hop behavior that should suffice for the applications using it. It provides a relative or absolute priority between packets so that different packet classes are treated according to the priority of the class.

Finally, Integrated Services are about strict guarantees provided by the network in terms of end-to-end delay (it could also apply to bandwidth guarantees). They consist in ubiquitous flow metering and scheduling that enable strict guarantees for every flow.

4.8.1 Differentiated Services

The Differentiated Services (DS) architecture (see [DS98]) is based on numerous requirements including avoidance of per-flow (layer 4) state within core routers, aggregated classification within routers, simple packet classification implementation...

The architecture relies on a number of functional elements implemented in routers, including a small set of per-hop forwarding behaviors, packet classification functions and traffic conditioning functions (metering, marking, shaping and policing). DS achieves scalability by implementing complex classification and conditioning at the boundary nodes of the network. Per-hop behaviors are applied in the core of the network to aggregates of traffic that have previously been marked using the DS field in the IP header. A single DS codepoint (DSCP) identifies a behavior aggregate (BA). Packets are forwarded within the core of the network according to the per-hop behavior (PHB) associated with the DSCP. A DS domain is a contiguous set of DS nodes, which operates under a common service provisioning policy. Each node implements a set of PHB groups. The DS boundary nodes classify ingress traffic and packets that transit the domain are marked. This ensures that a PHB from one of the PHB groups supported within the domain may be selected for every packet. Nodes within the domain select the forwarding behavior for packets based on their DSCP value. Boundary nodes and interior nodes therefore constitute a DS domain. DS boundary nodes interconnect the DS domain to other DS or non-DS domains. DS interior nodes only connect to other boundary or interior nodes of the same domain.

In order to connect several DS domains (or a DS domain with non-DS domains), service level agreements (SLA) must be established between upstream networks and downstream DS domains. Each SLA may specify packet classification, re-marking (attributing a new DSCP to an already marked packet) rules, traffic profiles and actions to traffic streams that are in- or out-of-profile. The packet classification policy identifies the subset of traffic that may receive a differentiated service by being conditioned and/or mapped to one or more BA (through re-marking). Traffic conditioning is about metering, shaping, policing and/or re-marking entering traffic so that it conforms to the rules specified in the traffic conditioning agreement (TCA) in application between the domains. Packet classifiers look at the content of some part of the packet header and classify packets matching some specified rule to an element of the traffic conditioner for further processing.

[15] Knowing the reason would not help anyway since sources cannot influence routing yet.

A traffic profile specifies the temporal properties of a traffic stream. It provides a means to determine whether a particular stream is in- or out-of-profile. It gives a correspondence between a DSCP value and a specification of the traffic stream (a token bucket for example). A profile indicates that all packets associated with it are to be measured against the traffic stream specification (be it a token bucket or something else). The conditioning actions associated with an "in-" or "out-of-profile" state of the stream can vary from one service to another. Packets from an out-of-profile stream can be marked so that the network can attribute lower scheduling priority to them. The packets may also be discarded. Figure 4.3 shows the logical view of a packet classifier and traffic conditioner. When a packet enters a DS domain, it is first selected by the traffic classifier, which steers the packet to a logical instance of a traffic conditioner. A meter is used to measure the traffic stream against a traffic profile. The resulting state of the meter with respect to the packet may be used to affect a marking, dropping or shaping action. When the packet exit the traffic conditioner of a DS boundary node, the DSCP must be appropriately set.

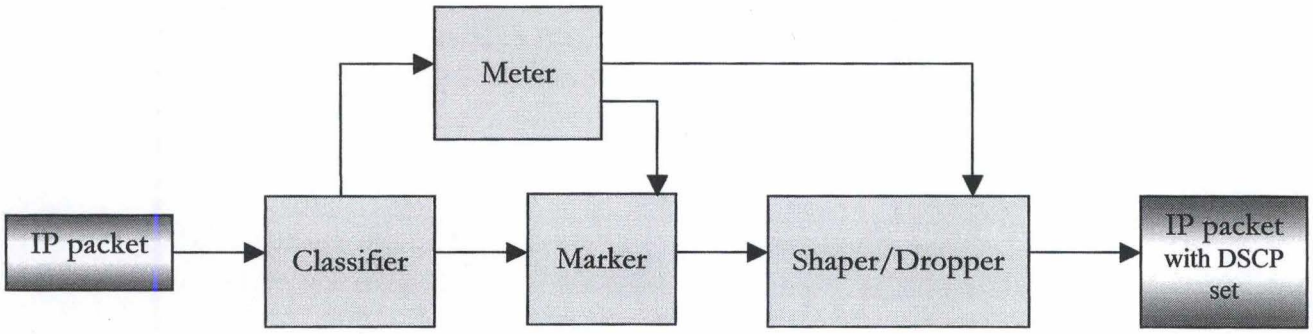

Figure 4.3: DS classification and conditioning operations

The packet marker sets the DS field, associating the packet to a DS behavior aggregate. The traffic shaper delays some or all packets of a traffic stream in order to bring the stream into compliance with its traffic profile. The dropper discards some or all the packets from a traffic stream in order to bring it into compliance with its traffic profile. Note that the shaper may also discard packets due to finite buffering space. The number of delayed packet might increase up to a point where some of them have to be discarded.

A PHB is a description of the externally observable forwarding behavior of a DS node applied to a particular DS behavior aggregate. The forwarding behavior depends on the relative load of the observed link. The PHB is the means by which a node allocates resources to BAs. Differentiated Services can be constructed upon this basic block. PHBs may be specified in terms of resource priority relative to others PHBs or in terms of their absolute or relative observable traffic characteristics (delay, jitter, loss). The two following sections present two examples of PHB groups.

4.8.1.1 Assured Forwarding

The Assured Forwarding (AF) PHB group, as defined in [AF99], proposes a general use PHB group providing delivery of IP packets in four independent forwarding classes. Each packet belonging to a particular AF class can be assigned one of the three drop precedence values. Packets of a particular layer-4 flow cannot be reordered as long as they belong to the same AF class. AF is a means for a provider DS domain to offer different forwarding assurances for IP packets received from a customer DS domain. There are four AF classes; each gets resources (buffers and bandwidth) allocated in the provider DS domain. IP packets, within each AF class, are marked with one of the three possible drop precedence values. This drop precedence value determines the relative importance of the packet within its AF class in case of congestion. A node experiencing congestion tries to protect packets with lower drop precedence from being lost from those of higher drop precedence. The level of forwarding

assurance in a particular DS node depends on not only the drop preference of the packet but also on how much forwarding resources have been allocated to the AF class and the current load of the AF class in the node. The amount of AF traffic entering and exiting a DS domain may be controlled through traffic conditioning actions. The allowed traffic conditioning actions comprise traffic shaping, discarding of packets, changing the drop precedence of packets and reassigning packets to other AF classes.

The objective of AF is to minimize long-term congestion within each class while at the same time handling short-term congestion. Long-term congestion is thus avoided by dropping packets while short-term congestion is handled by queuing them. The dropping algorithm must treat all packets within a single class and precedence level equally, for example by trying to give equal dropping probability to flows that have the same long-term behavior but different short-term burstiness.

An example of an implementation the AF group is the Olympic service. Three service classes are defined called the gold, silver and bronze service. Packets in the gold class experience lighter load than those of the silver class, the silver class packets receiving better service than those of the bronze class. Drop precedence could also be defined within each service class. The drop precedence could be implemented via a distinct token bucket that would be less constraining for lower precedence flows. For example, lower precedence flows could get more tokens than higher precedence ones but the same bucket size.

4.8.1.2 Expedited Forwarding

The Expedited Forwarding (EF) is another example of an implementation of a PHB group. It is defined in [EF99]. Its objective is to provide a low loss, low latency, low jitter, assured bandwidth, and end-to-end service through a DS domain. Since loss, latency and jitter occur due to queues in the network devices, ensuring that no queues will be experienced by the traffic is equivalent to bounding the time spend by packets within network nodes. This is possible by controlling that the maximal arrival rate never exceeds the departure rate at each node. EF traffic should receive the same treatment independently of any other traffic transiting the same node. A possible implementation could allow unlimited preemption of other traffic while at the same time managing the damage inflicted to preempted traffic. Managing the preempted traffic means that some EF packets could be discarded within the network. The mechanism used by EF consists in shaping the EF traffic at the boundary nodes so that packets within the network are forwarded immediately. Shaping should ensure that EF packets that are within the network strictly comply with the service specification (they consume the right amount of resources) so that they should be forwarded immediately by interior nodes. If an excessive number of EF packets get into the network, it must be due to an erroneous condition. EF packets should thus be discarded and not other PHB ones even if they seem to have lower drop precedence. It could seem contradictory that higher priority packets be discarded first but the faulty situation has been caused by the EF class so the EF class is punished.

4.8.2 Integrated Services

The Integrated Services model as presented in [IS94] does not propose a new routing architecture. Instead, it defines extensions in order to carry real-time traffic across the best-effort Internet. The model defines two types of services: guaranteed and predictive service. Guaranteed service means that there will be an absolute upper bound on the network delay. Predictive service objective is to give a delay bound that is as low as possible and at the same time stable enough to be evaluated by the receiver (its use is clearly directed to real-time applications). Integrated Services main components are resource reservation and admission control. These components are a consequence of a will to explicitly manage network resources in order to meet applications requirements. The current Internet architecture relies on the assumption that end-systems should maintain all flow-related states. The

Integrated Services model imposes that flow-specific states be maintained by routers. The main side effect arising from resource reservation in routers relates to the need for administrative control and enforcement of policy because some users are getting privileged service. Routers must therefore be able to identify users requesting resources and packets using these resources.

The architecture of Integrated Services relies on four components: the packet scheduler, the admission control routine, the classifier and the reservation setup protocol. The first three components constitute the traffic control. Traffic control is the function by which a router creates different QoS. The packet scheduler serves at managing the forwarding of the packets inside a router. It uses mechanisms like distinct queues, timers and priorities. The classifier maps each incoming packet to a class based on local (internal to the router) or external (packet's header or input port) information. The concept of "class" refers to a particular flow or any aggregation of several flows. The admission control mechanism determines whether a new flow can be granted to access the required resources, without impacting on the reservations made previously. Admission control is invoked at each node traversed by the reservation path. Do not confuse admission control with policing or enforcement. Policing (or enforcement) occurs at the edges of the network for every packet to ensure the user conforms to its traffic contract. Admission control relates to a flow or an aggregated flow establishment. The reservation setup protocol creates and maintains flow-specific state all the way from the source to the destination (be it within routers as well as endpoint hosts). It is up to the application to specify its resources requirements, which are carried via the reservation setup protocol. Admission control then proceeds for a test for acceptability concerning the reservation information. In case of success for admission control, the reservation is translated into parameters for the packet scheduler.

Like every other service model, Integrated Services relies on a core of service commitments (i.e., a traffic contract) that specify a response from the network (in terms of service delivery) to a service request. In order to know what type of service the model should provide, a good approach consists in characterizing the QoS requirements of the flows. The service model is almost uniquely concerned for the per-packet delay so that quantitative QoS are relative to maximum and minimum packet delay. With this assumption in mind, characterization of application needs becomes quite simple: applications are inelastic (real-time) or elastic. Inelastic applications require that packets arrive within a certain time interval. If not, they become worthless due to real-time needs. An example is "playback" applications that bufferize packets and play the (audio and/or video) signal at the receiver's side. If one or more packets arrive too late at the destination, the application will not be able to give the user an acceptable quality. On the other hand, elastic applications do not require data to arrive before a specific time. They just wait for the data and continue when it is there. Examples of such applications are interactive data transfer (FTP) and asynchronous data transfers (mail, FAX). Admission control clearly differs for inelastic and elastic applications since elastic ones do not require any delay nor jitter guarantee. Elastic applications do not need any admission control due to their ability to adapt to any network resource state.

The previous applications taxonomy gives an idea about what service the network should provide to a particular flow. However, routers need to cope with a collection of such flows. Resource sharing therefore comes into play. While only inelastic applications need resource reservation, they cannot use all resources. One has to choose the part of the resources each application type can reserve. Link sharing addresses the problematics of how to share the aggregate bandwidth between all existing flows on individual links. An example of such policing might attribute for each traffic class (real-time flows, interactive elastic, non-interactive elastic...) a minimal fraction of the link bandwidth in order to prevent greedy flows from consuming all available resources. Ideally, all flows from a given class should be given $1/n$ of the link fraction attributed to the class where n is the instantaneous number of flows of the class. This model describes an idealized fluid model with instantaneous proportional link sharing. It has been introduced in [DKS89] and further explored in [Par92] but this is out of the scope of this dissertation.

The service model says what to expect from the network when access to a particular service has been granted to a flow. It does not say anything about the way an application negotiates for a QoS level yet. It is up to the reservation model to tackle this issue. Several options exist for the negotiation procedure. The network may either accept all terms of the QoS requirements or reject the request. If the network cannot provide the required QoS, it might propose (or directly grant) a lower resource reservation service. Given that most applications properly work with a range of QoS, they could adapt to degraded service. For example, they could use different encoding techniques or vary the amount of buffering at the receiver's side.

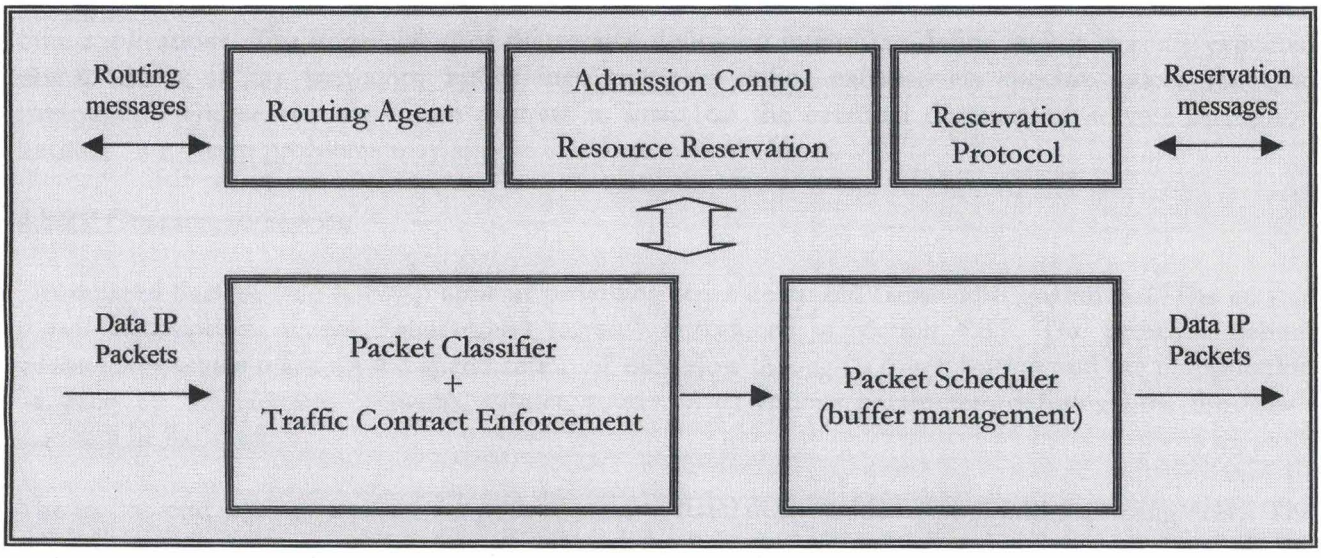

Figure 4.4: Integrated Services router architecture

What we have seen about Integrated Services gives an idea about what to expect from the network. However, how do the routers ensure the fulfillment of such guarantees? This is the job of the traffic control mechanisms. Traffic control uses basic router's functions in order to match the effective treatment packets obtain within every router along the path with the QoS guarantee ensured by the network. Its main components comprise packet scheduling, packet dropping, packet classification and admission control. Packet scheduling is about choosing which packet to send to the output port. Several schemes exists from the simplest FIFO queue to the complex Weighted Fair Queuing (see [Par92]) to determine which packet send first, each method having advantages and drawbacks in terms of operational complexity, fairness and efficiency. Packet dropping should never happen but routers do not have infinite buffering capacity. Sometimes, buffers get full so a choice is necessary: which packet to drop? First, note that a full buffer actually relates to the buffer occupancy going beyond a defined threshold. Second, classification could have led to distinct packet priorities so either one single threshold exists for every priority class or a single threshold serves for the whole buffer. Finally, not all packets are equal in terms of the consequences of dropping. With the "over-lying" TCP performing congestion control, dropping a TCP segment means choosing a TCP source to throttle. Be aware that local packet dropping could relate with achieving the desired end-to-end QoS. If the buffer queue length increases, dropping one packet reduces the delay experienced by all packets situated logically after it in the queue. A local loss (drop) might lower the delay of many flows.

4.8.2.1 Controlled-Load

The Controlled-Load service, as defined in [CL97], corresponds to the "predictive service" introduced in section 4.8.2. It aims at providing to each flow a QoS similar to the one that would be received under a lightly loaded network condition. However, Controlled-Load has to provide it under any actual network condition. Applications using the Controlled-Load service are assured that:

The service model says what to expect from the network when access to a particular service has been granted to a flow. It does not say anything about the way an application negotiates for a QoS level yet. It is up to the reservation model to tackle this issue. Several options exist for the negotiation procedure. The network may either accept all terms of the QoS requirements or reject the request. If the network cannot provide the required QoS, it might propose (or directly grant) a lower resource reservation service. Given that most applications properly work with a range of QoS, they could adapt to degraded service. For example, they could use different encoding techniques or vary the amount of buffering at the receiver's side.

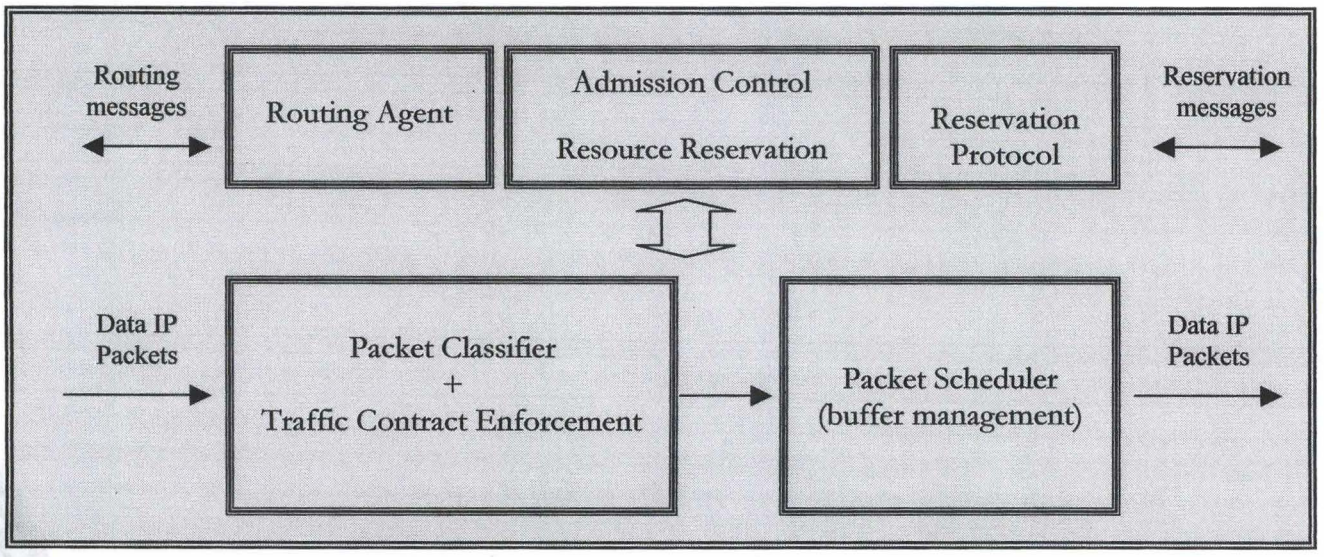

Figure 4.4: Integrated Services router architecture

What we have seen about Integrated Services gives an idea about what to expect from the network. However, how do the routers ensure the fulfillment of such guarantees? This is the job of the traffic control mechanisms. Traffic control uses basic router's functions in order to match the effective treatment packets obtain within every router along the path with the QoS guarantee ensured by the network. Its main components comprise packet scheduling, packet dropping, packet classification and admission control. Packet scheduling is about choosing which packet to send to the output port. Several schemes exists from the simplest FIFO queue to the complex Weighted Fair Queuing (see [Par92]) to determine which packet send first, each method having advantages and drawbacks in terms of operational complexity, fairness and efficiency. Packet dropping should never happen but routers do not have infinite buffering capacity. Sometimes, buffers get full so a choice is necessary: which packet to drop? First, note that a full buffer actually relates to the buffer occupancy going beyond a defined threshold. Second, classification could have led to distinct packet priorities so either one single threshold exists for every priority class or a single threshold serves for the whole buffer. Finally, not all packets are equal in terms of the consequences of dropping. With the "over-lying" TCP performing congestion control, dropping a TCP segment means choosing a TCP source to throttle. Be aware that local packet dropping could relate with achieving the desired end-to-end QoS. If the buffer queue length increases, dropping one packet reduces the delay experienced by all packets situated logically after it in the queue. A local loss (drop) might lower the delay of many flows.

4.8.2.1 Controlled-Load

The Controlled-Load service, as defined in [CL97], corresponds to the "predictive service" introduced in section 4.8.2. It aims at providing to each flow a QoS similar to the one that would be received under a lightly loaded network condition. However, Controlled-Load has to provide it under any actual network condition. Applications using the Controlled-Load service are assured that:

- Packet loss rate will be close to the transmission medium error rate;
- Transit delay experienced by packets will be close to the minimum transit delay.

To make such guarantees possible, the user is required to provide the intermediate network elements an estimation of the traffic he will generate. If the user does not comply with the specification of its estimated traffic pattern, the QoS experienced by the non-conforming packets can exhibit characteristics of a heavily loaded network (packet loss or important delay).

The applications targeted by this service are those very sensitive to network load degradation like real-time applications. The imprecision of the service definition intends to define events that are expected not to occur at any frequency rather than trying to define exhaustively specific values for QoS parameters. The service definition permits to insist on the eventual duration of service disruption because short-term problems may appear due to statistical effects.

4.8.2.2 Guaranteed Service

Guaranteed Service (see [GS97]) aims at providing strict delay and bandwidth guarantees. The service relates corresponds to the "guaranteed service" introduced in section 4.8.2. The principle behind guaranteed service relies on the specification of each flow through a token bucket and the computation by every service element (network, subnet, router...) of various parameters defining how the flow's data will be handled.

The end-to-end delay of a particular path contains a fixed delay (transmission) and a queuing delay. The fixed delay only depends on the chosen path, essentially its length and the transmission media. Only the setup mechanism influences the fixed delay[16]. However, two parameters under the application's control allow working on the queuing delay: the token bucket and the data rate. This means that an application is able to accurately estimate the queuing delay delivered by Guaranteed Service. If a too important delay is experienced, the application may modify its token bucket definition to get satisfying services. The end-to-end behavior provided by the network elements on the path ensures an upper bound on the delay with no queuing loss for all conforming packets (assuming no transient routing periods are experienced).

The targeted applications are those needing packets to arrive before a specific delay like "play-back" applications. The service does not attempt to minimize the jitter but it just controls the queuing delay. This leads to a "fluid model" service where a logical dedicated wire of a specific guaranteed bandwidth exists between the source and the destination.

4.8.3 Integrated Services over Differentiated Services

Integrated Services (Intserv) and Differentiated Services (Diffserv) do not form a partitioning of the service model space. Each provides a particular service but integrating them may ensure a "better" service (see [ISDS]). The idea consists in implementing Intserv over Diffserv networks. The Diffserv networks support the end-to-end QoS needs of Intserv. The role of Intserv, Diffserv and the reservation mechanism (which is due to be RSVP in the Intserv context) is to facilitate the deployment of real-time applications (IP Telephony, video-on-demand) and other non-multimedia critical applications in a scalable manner. On one hand, Intserv will allow per-flow resource requests as well as admissibility feedback for them. On the other hand, Diffserv provides scalability across the network. Each Diffserv network is treated as a virtual link.

[16] By making a choice with regard to the path of course, not by changing its physical properties.

The advantage of combining the two models relates to the scalability of Diffserv and the ability of Intserv to perform some strict traffic conditioning at the boundaries of Diffserv networks. This would make the architecture more robust while at the same time provide end-to-end guarantees to specific flows. The framework assumes the method to work even with some of the transit Diffserv networks are not RSVP-aware in which case reservation messages should traverse them transparently.

The principle is that Intserv service requests be mapped at the ingress of every Diffserv transit network by selecting the appropriate PHB corresponding to the requested service. Since Diffserv does not use flow-specific state within routers, every transit Diffserv network is likely to perform aggregate traffic control. Note that such a method does not exclude end-to-end guarantees because a loose path exists for each flow between the source and the destination thanks to RSVP. Of course, the service will not be as strict as it could be with end-to-end Intserv but one cannot realistically expect per-flow state reservations in the Internet today.

The main issue will probably relate to resource management in Diffserv networks. A wide number of options are possible: statically provisioned resources, dynamically provisioned resources by RSVP or other means. Static provisioning implies the negotiation of a contract between Intserv and Diffserv regions so that Intserv routers only inject the amount of traffic allowed by the contract. The contract thus enables provisioning to be made in the Diffserv network to ensure performance guarantees in the Diffserv part. With dynamic provisioning, some kind of an "oracle" might perform admission control based on the network state knowledge. The lack of explicit reservations at the edges of the Diffserv networks could be bypassed through some kind of "magic" consisting in distributed knowledge of the network state (within routers) or a centralized temporal database maintaining statistics about time-dependent and topology-dependent load trends.

4.8.4 Evaluation

Even if the previous service models provide some more guarantees than best-effort service, their use may be questioned. Differentiated services might be deployed in today's routers thanks to their simple requirements. Simplicity also arises with limited guarantees: a PHB does not provide strong end-to-end guarantees. Although the Olympic service implementation of the AF gives some relative service (compared to other service classes), it is mainly directed to Intranets. It provides the assurance that some traffic will get better service than others will. No strict service may be ensured so it is difficult to find a monetary correspondence for a given service class due to the relativity of the service specification. EF may provide some kind of "virtual leased line" service assuming its implementation gives actual preemption over all other traffic classes. Integrated Services may provide enough guarantees for most applications but implementation cost is high: classification, reservation state, scheduling and policing need to be made for each layer-4 flow. The burden seems far too heavy for high-speed routers. Hybrid schemes (see section 4.8.3 for an example) are more likely to be implemented in the Internet in a close future.

4.9 Resource Reservation Mechanisms

While the definition of service models constitutes a good point, everything relies on the existing routing architecture. Trying to provide guarantees without thinking about resource reservation mechanisms is useless. This is when RSVP and CR-LDP enter the QoS arena.

4.9.1 RSVP

Resource reSerVation Protocol (RSVP), as its name implies, is a resource reservation setup protocol designed for an Integrated Services Internet (see [RSVP97]). It provides receiver-initiated setup of resource reservations for multicast as well as unicast data flows. Applications may select a specific QoS among the services the network proposes. RSVP reserves resources along the path from the source to the destination. Note that RSVP flows are simplex flows. RSVP is not a routing protocol but only a control protocol: it has been designed to operate on every routing protocol, be it unicast or multicast. RSVP itself uses routing protocols to obtain routes. We saw in section 4.8 the place of the reservation protocol part within Integrated Services.

When a RSVP QoS request arrives at a router, it is submitted to two decision modules: admission control and policy control. The admission control module determines if enough resources are available to accept the reservation. The policy control module checks whether the user has administrative permission to make the reservation. If the request passes the two tests, the QoS parameters for the flow are set in the classifier so that incoming packets belonging to the flow get the treatment corresponding to the QoS. The policy control and QoS parameters are carried by RSVP as opaque data, i.e. they are only interpreted by the appropriate modules within routers. RSVP does not care about the parameter's values.

RSVP uses soft-state reservation that needs to be periodically refreshed. Otherwise, the reservation is cancelled. This feature enables RSVP to efficiently manage multicast groups. Another advantage concerns good scalability for interdomain flows.

RSVP defines the notion of "session" to be a data flow with a particular destination and transport-layer protocol. Each session is treated as a distinct entity. More formally, an RSVP session is defined by the triple *<Destination address,Protocol Id [,Destination Port]>*. *Destination address* is an IP unicast or multicast address. *Protocol Id* is the IP protocol ID. *Destination Port* is an optional parameter used for demultiplexing at the transport or application layer.

The reservation model relies on the concepts of a "flowspec" and of a "filter spec". These two concepts together form a "flow descriptor". The flowspec specifies a desired QoS. When a session specification is added to a flowspec, it defines the flow that will receive the QoS defined by the flowspec. The flowspec serves at setting the parameters in the router scheduler. The packet classifier module only uses the filterspec. When packets addressed to a session do not match any of the filter specs for the session, they are forwarded as best-effort traffic. The model relies on a "one pass" reservation style: a receiver sends a reservation request upstream and each node in the path either accepts or rejects the request. This reservation style is clearly not the best in order to find a path verifying the requested QoS. A solution consists in making the control packets flow downstream, following the data path, thus gathering information that might serve to evaluate the end-to-end QoS. This does not always give a path having the desired QoS properties but it allows the receiver to eventually lower its QoS requirements so that a satisfying path may be found[17].

RSVP defines several reservation styles comprising two options. The first option concerns the sharing of the reservation between sender's packets. The second one specifies which senders are allowed to use resources of a particular flow. Three reservation styles exist: Wildcard-Filter (WF) style, Fixed-Filter (FF) style and Shared Explicit (SE) style.

The WF style means reservation sharing between all senders. At every router on the path, all upstream senders share the same common reservation. The FF style creates a distinct reservation for data packets

[17] At least there will be some reservation made.

from a particular sender, protecting them from the ones belonging to other senders of the session. The SE style allows an explicit subset of the senders to share the reserved resources. Shared reservation styles (WF and SE) target multicast applications for which the number of active senders is relatively small compared to the absolute number of senders. Unicast flows may easily reserve resources with the FF reservation style. A single sender may use the reserved resources without sharing them with other sources.

4.9.2 CR-LDP

CR-LDP stands for Constrained-based Label Distribution Protocol (see [MPLSCRLDP]). While LDP is used to establish LSPs through a MPLS network, it does not ensure by itself that the selected path verifies some defined constraint other than the ones implicit from the routing protocol used (shortest path or least-cost path). The setup of an LSP may be based on explicit route constraints, QoS constraints, ... The difference between a classical LSP (one established through LDP only) and an CRLSP is that the constrained-based route is calculated at one point at the edge of the MPLS network based on several criteria, not solely on the routing information. This enables to associate a service class with every CRLSP, be it in terms of QoS (bandwidth, delay, jitter...) or a guarantee that physically separate alternative paths exist and are ready to carry the traffic.

An explicit route constraint is defined by a list of nodes or groups of nodes the CRLSP is allowed to go through. When the CRLSP is established, all or a subset of the nodes in a group may be crossed by the LSP. The ability to specify a group of nodes allows LSPs to be established with imperfect information about the network topology. The constrained-based route is encoded as a list of abstract nodes. An abstract node is either an LSR or a group of LSRs.

The CR-LDP LSP establishment procedure signals the resources required at every hop of the path. If a satisfying route cannot be found in the network, already established LSPs might be rerouted to free enough resources so that a satisfying route is found. The path pre-emption process allows existing path to be pre-empted and eventually rerouted if an LSP that holds a higher priority needs already reserved resources. Another feature of CRLSPs is route pinning: loosely routed LSPs segments may use route pinning so that the loose part of the LSP does not change if a better path is found after the LSP setup.

The CRLSP over LDP solution aims at enabling traffic engineering functionality and performing more general constrained-based routing than what is required for traffic engineering only.

4.9.3 Evaluation

So what? CR-LDP or RSVP? Probably both. To see why both seem better compared to the use of only one of them, let us look at their respective advantages. The soft-state RSVP provides scalability. Interdomain and big traffic LSPs may constitute its main targets. The hard-state "LDP + CR-LDP" approach on the other hand allows fast-restoration, re-routing and on the fly LSP optimization functionality that are best achieved in a quite small network. The more realistic solution probably consists in using CR-LDP within a particular domain (intradomain) and RSVP for interdomain path setup. These protocols also require that routing protocols provide the necessary QoS and path availability information. Be aware that end-to-end guarantees require extensions to existing intra and interdomain routing protocols. While a lot of work has already been done for what concerns intradomain routing protocols QoS extensions (see [QoSOSPF]) and intradomain traffic engineering extensions (see [OSPFTE] and [ISISTE]), interdomain routing seem to have been forgotten. In order to provide a coherent routing architecture, the integration of QoS in the Internet should cope with both intra and interdomain.

4.10 Traffic Engineering with MPLS

We saw in the previous chapter a description of MPLS and its main features. Now that basic traffic engineering concepts have been introduced, let us see why MPLS is presented as the new traffic engineering tool (see [MPLSTE]).

4.10.1 Induced MPLS Graph

An induced MPLS graph is a set of LSRs that compose the nodes of the graph and a set of LSPs that connect the nodes, providing logical connectivity between the nodes. The graph may be composed of logical nodes, which are not a single LSRs but a set of LSRs, thus forming a logically hierarchical graph, by using the label stack mechanism. The reader could ask why we did introduce this concept. Just because the problem of managing a QoS constraint relates to being able to map such a graph with the physical topology!

4.10.2 The Problem

The problematics of traffic engineering over MPLS can be decomposed into three fundamental sub-problems:

- The mapping of packets into FECs;
- The mapping of FECs into traffic trunks;
- The mapping of traffic trunks onto the physical topology through LSPs.

The first sub-problem tackles the definition of a flow. More precisely, at which level of the OSI model the flow definition occurs. It deals with the granularity at which the traffic will be decomposed. It is essentially a performance matter and eventually a billing one. The finer the granularity, the better the potential control over traffic. The coarser the granularity, the lesser the burden placed on border LSRs due to a restricted number of simultaneous flows.

The second sub-problem concerns the aggregation of individual flows for performance purposes. Every FEC at ingress LSRs might be mapped to an individual LSP in the network. That would let an open door for the well-known problem of scalability. Some aggregation is therefore required inside the network in order to perform fast label swapping.

The last sub-problem is about finding a path in the network for traffic trunks constrained by:

- Properties inherent to the flows that constitute the trunks;
- Constraints related with network resource distribution (the placement of trunks among themselves).

4.10.3 Traffic Trunk

Strictly speaking, a traffic trunk is a unidirectional "aggregate" of traffic flows belonging to the same class (in the context of a service model). This definition might be extended to the support of multiple classes within a traffic trunk. Because a traffic trunk differs from the LSPs it traverses, it may be physically re-routed. A traffic trunk is characterized by its ingress and egress LSRs, its FEC and a set of

attributes that defines its behavioral characteristics (relative to the service model from which the class emanates).

4.10.4 Traffic Trunk Attribute

An attribute of a traffic trunk is a parameter that influences its behavioral characteristics. Basic attributes comprise traffic parameter, generic path selection and maintenance, priority, pre-emption, resilience and policing.

A traffic parameter captures the characteristics of the FECs that constitute the traffic trunk. The characteristics may express themselves via a token bucket specification[18]. Generic path selection and maintenance attributes serves at selecting the route taken by traffic trunks and at determining the rules for maintenance of already established paths. Paths may be selected administratively by the network operator or automatically computed by a routing protocol (using QoS-based topologies eventually constrained by policy restrictions). A very important aspect of path maintenance concerns the adaptability of traffic trunks. Since the network state change over time, some resources become available while others are allocated. Some traffic trunks could therefore obtain a better path through re-optimization. The re-optimization mechanism must be an optional adaptability attribute of the particular traffic trunk. The priority attribute enables to specify the relative importance of traffic trunks. It influences the order in which path selection occurs at establishment time and under faulty situations. The pre-emption attribute determines whether a traffic trunk can pre-empt another traffic trunk (be it from a specified path or from any path). Pre-emption permits to ensure that high priority traffic gets the most favorable paths through the network. The resilience attribute determines the behavior of the traffic trunk under fault conditions. It specifies the recovery procedure to apply on traffic trunks in the case of a faulty path. Several options for recovery schemes include re-routing the traffic trunk to an already provisioned LSP, re-routing to a feasible path with enough resources if one exists, rerouting to any available path, and many other schemes. The policing attribute indicates the actions to apply when a traffic trunk does not conform to its traffic contract.

4.10.5 Constrained-Based Routing

Since MPLS is due to be an additional element of the current Internet routing architecture, constrained-based routing in the MPLS context should logically co-exist with current topology-driven IGP routing protocols (preferably augmented by QoS extensions). Based on traffic trunk requirements (QoS and policy), resource availability information and topology information, a constrained-based routing process may compute explicit routes at each node for each traffic trunk originating from the node. The question of either every node or only border nodes will implement this process is an architectural matter. At least, we may expect from the constraint-routing framework to provide a solution for the traffic trunk placement problem. Even if it does not provide an optimal solution (due to inaccurate information provided by the routing protocol or some real-time calculation constraint), it should provide a basic solution based on some heuristic algorithm. The constrained-based routing problem has been proved to be NP-complete for more than two (independent) constraints. Hence, the following heuristic may be used to find a feasible path:

1. Start with the resource-independent graph given by the routing protocol;
2. For each resource constraint, prune the residual graph for links that do not satisfy the resource constraint required by the traffic trunk;
3. Run a shortest path algorithm on the residual graph.

This algorithm should allow finding a feasible path if such exists. If several satisfying paths exist, the one due to minimize either congestion or call-blocking rate should be chosen. The drawback of this algorithm lies in not considering simultaneously the routing of all traffic trunks. It implies that the algorithm will not always find a path for each trunk even if such a solution exists. Figure 4.3 shows the expected architecture of the constrained-based routing process in an LSR running both "classical" IGP routing protocol and constrained-based routing.

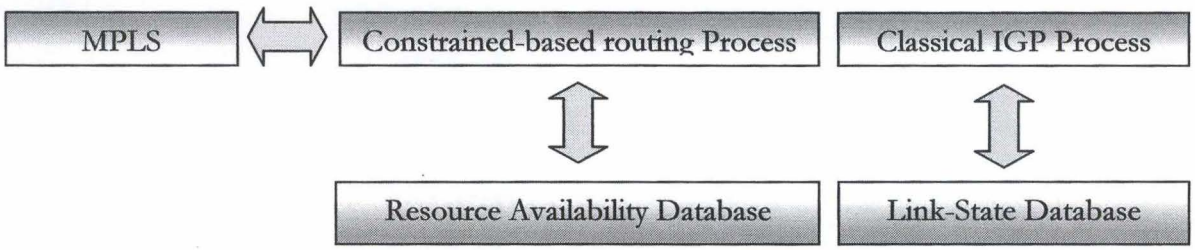

Figure 4.5: Constrained-based routing process on an LSR

4.11 PASTE

PASTE stands for Provider Architecture for Differentiated Services and Traffic Engineering (see [PASTE]). We have seen that DS require maintaining per-flow state in every DS router along the path. This raises some scaling problems for large ISPs. Large ISPs need the ability to perform traffic engineering by aggregating the flows and balancing their load across the network. The architecture defined by PASTE uses MPLS as packet forwarding mechanism while RSVP carries control information as well as QoS requests. In order to include DS in the architecture, traffic-handling operations (queuing, dropping, scheduling...) are supported on a per-trunk basis.

A traffic trunk abstracts the aggregation of several flows. It allows limiting the aforementioned scaling problem by decoupling the overhead of the infrastructure (network control) from the size of the network and the amount of traffic. As the traffic and the number of flows increases, only the total traffic carried by trunks increases and not the number of trunks. The worst case concerning the number of trunks that need to be simultaneously instantiated is $(N \times (N-1) \times C)$ where N is the number of border routers and C the number of service classes. This number decreases if sink trees aggregate trunks physically sharing a subset of their respective path.

RSVP has been designed to allow a host to make a resource reservation, on behalf of an application data stream, to request a QoS from the network. PASTE makes use of RSVP in some different way: RSVP serves at installing state that applies to a collection of flows that share a common path and common resources. In addition, RSVP installs label switching information. RSVP does not pass QoS parameters only but it manipulates explicit route objects and label bindings. This new functionality of RSVP integrates the QoS request with the LSP establishment mechanism.

[18] A token bucket specifies in a very simple and general manner the temporal properties of a traffic stream.

4.12 Evaluation

We did present in this chapter a non-exhaustive "state-of-the-art" of today's traffic engineering. It is obvious that strict guarantees are far from being in sight. DS only gives the user a taste of QoS while IS seems too complex to implement in today's Internet. Whether or not MPLS might allow a scalable implementation of IS in the core of the network is an open question. Traffic engineering is still a concept, not a reality. Throughout this chapter, we saw that traffic engineering could become somewhat "more real" with today's tools for controlling and characterizing the traffic.

Traffic control will be very difficult to achieve in a heterogeneous Internet. While intradomain traffic engineering may be resolved in parts via the tools mentioned in this chapter, interdomain traffic engineering might be a real problem. Interdomain routing must cope with scalability-related matters. Size and lack of information are inherent to the interdomain level. Flow aggregation may partially resolve the interdomain problem. However, the characteristics of aggregated traffic will determine whether QoS can be provided to interdomain flows.

Chapter 5

Interdomain Traffic Traces Analysis

5.1 Introduction

While today's Internet is a best-effort one, trying to build the QoS Internet requires a thorough understanding of the way actual traffic behaves. This chapter intends to provide a detailed analysis about the way interdomain traffic behaves in the Internet today. The focus of our analysis concerns interdomain traffic variability. We evaluate the cost of LSP establishment techniques by studying the implications of carrying today's traffic through MPLS at the interdomain level. The chapter is structured as follows.

We first present the measurement environment that served at gathering our traffic traces. We then present several possible LSP establishment techniques to carry interdomain flows and discuss their limitations. After that, we study the performance of the LSP establishment techniques on the traffic traces. Because LSPs not only serve at routing purposes, studying the behavior of bandwidth reservations made on these LSPs will show us what guarantees can be provided to interdomain flows. We thus evaluate the performance of bandwidth reservation techniques in terms of the ratio of traffic carried on the guaranteed bandwidth LSPs and the ratio of bandwidth that has not been used. Finally, we present our conclusion concerning the analysis carried on the traffic traces and give some ideas of further work.

Note that in order to understand the analysis presented in this chapter, we assume the reader to have a basic knowledge of how interdomain routing works in the Internet today. Readers that do not possess this background are referred to Appendix 6.

5.2 Measurement Environment

5.2.1 Topological Context

To evaluate the variability of interdomain traffic, we consider traffic traces from two completely different ISPs. By basing our work on two different networks, we limit the possibility of measurement biases that could have been caused by a particular network. The two ISPs had different types of customers and both were multi-homed.

The first ISP, WIN[19], was at the time of our measurements a new ISP offering mainly dialup access to home users in the southern part of Belgium. We call this ISP the "dialup" ISP in the remainder of this chapter. The dialup ISP was connected through E1 links to two different transit ISPs and at the Belgian national interconnection point BNIX[20], having peering agreement with about ten ISPs there. The dialup ISP balanced the incoming traffic with its two upstream ISPs by announcing different address prefixes to them. When we performed our analysis, the dialup ISP had about 4.5 MBPS of external traffic (mainly incoming) during peak hours and it only received unicast traffic. The dialup ISP did not maintain a full BGP routing table in its border routers.

[19] http://www.win.be
[20] http://www.bnix.net

The second ISP, Belnet[21], provides access to the commodity Internet as well as access to high-speed European research networks to universities and research institutions in Belgium. We call this ISP the "research" ISP in the remainder of this chapter. The Belnet national network relies on a 34 MBPS backbone linking major Belgian universities. The research ISP differs from the dialup ISP in several aspects. First, the "customers" of the research ISP are mainly researchers or students with direct high-speed connections to the 34 MBPS backbone of the research ISP, although some institutions also provide dialup service to their users. Second, the research ISP is connected to a few tens of external networks with high bandwidth links. Its access points consist in high bandwidth links to two transit ISPs, an E3 link to the Dutch research network (SURFNET) and 1.5 DS3 links to the TEN-155 European research network. In addition, the research ISP is present at two national interconnection points: in Belgium and in the Netherlands. The research ISP has local peering agreement with about 40 different networks. It maintained a full BGP routing table in its border routers and used this routing table to balance the traffic between its directly connected peers. When we performed our measurements, the research ISP had about 80 MBPS of external traffic during peak hours from which 75 % was incoming.

5.2.2 Collection of Traffic Traces

To gather interdomain traffic traces, we relied on *Netflow* (see [Cis99]) measurement mechanisms supported on the border routers of the two ISPs. *Netflow* provides a record at the flow level. For a TCP connection, *Netflow* will record the timestamp of the connection establishment and connection release packets as well as the amount of traffic transmitted during the connection. For UDP flows, *Netflow* records the timestamp of the first UDP packet for a given flow, the amount of traffic and relies on a timeout for the ending time of a UDP flow. *Netflow* traces are less precise than *tcpdump* traces used in many measurement papers (see [NML98], [FRC98] and [NEK+99]) since it does not provide information about the arrival time and size of individual packets within a TCP connection. In this chapter, we assume that an M bytes flow, which is active during d seconds, is equivalent to a continuous M/d bytes per second flow. This approximation does not cause problems for our study since we work on aggregated flows with a one-minute granularity. Because *Netflow* data collection has been configured with one-minute summarization intervals, all traces recorded traffic statistics with a one-minute time unit. This means that all packets belonging to a particular flow were attributed to the current minute interval like a fluid flow linearized over one-minute intervals. Figure 5.1 illustrates the *Netflow* summarization process leading to an underestimation of the real burstiness of the traffic. For both ISPs, the *Netflow* information was transmitted in real-time by the border routers to a monitoring workstation inside the ISP. The monitoring workstation of the research ISP used *cflowd* (see [CAIDA]). In both cases, the border routers exported the *Netflow* information for the external unicast traffic, incoming and outgoing.

Netflow traces were treated thanks to *arts++* libraries (see [CAIDA]), enabling to transform *cflowd* binaries into ASCII files. These ASCII files were in turn submitted to several scripts written in Perl aggregating flow-related information according to routing objects granularity (BGP prefixes and ASs). The tools that served for the traffic traces analysis are presented in Appendix 5.

[21] http://www.belnet.be

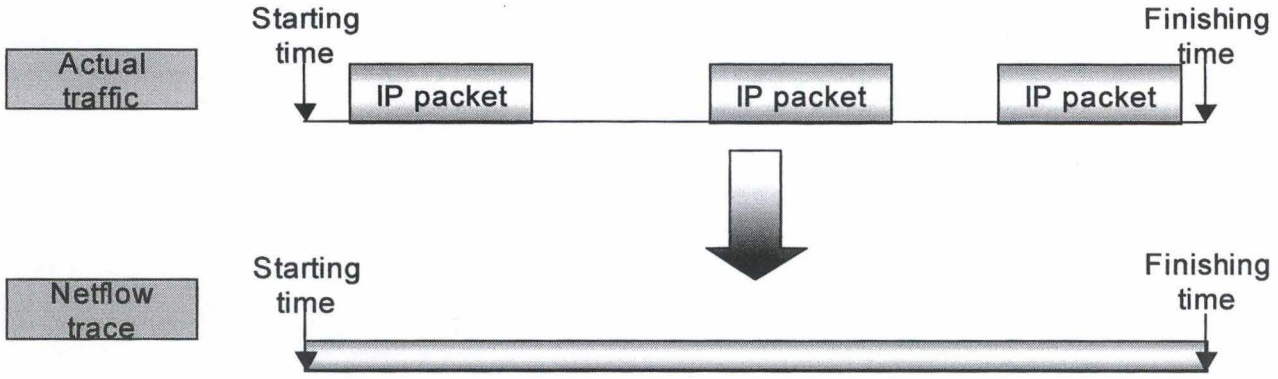

Figure 5.1: *Netflow* summarization process

5.3 Daily Traffic Evolution

The two ISPs exhibit very different behaviors, especially for daily traffic evolution. Figure 5.2 shows incoming traffic evolution for the dialup ISP while Figure 5.3 incoming and outgoing traffic evolution for the research ISP, both for one day. 37 Gbytes of incoming traffic were observed for the dialup ISP. The research ISP experienced 390 Gbytes of incoming traffic and 158 Gbytes of outgoing traffic during the analyzed day. The percentage of TCP for the research ISP was 97.5 % for incoming traffic and 95.8 % for outgoing traffic. This high percentage of TCP proves that current traffic in the Internet is best-effort. Figures 5.2 and 5.3 illustrate the difference between the two ISPs for what concerns not only total traffic variability but also peak hours location within the day. Mean incoming traffic for the dialup ISP was around 3.5 MBPS while the research ISP had a mean around 36 MBPS for incoming traffic and 14.5 MBPS for outgoing traffic.

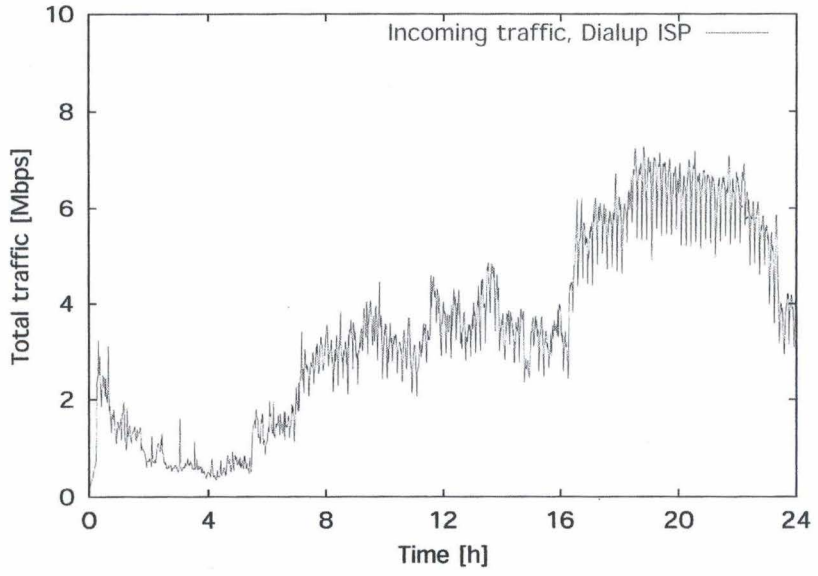

Figure 5.2: Daily traffic evolution for dialup ISP

Daily traffic evolution exhibits the influence of customer profile on peak hours location within the day. Dialup ISP traffic has its peak in the late evening while research ISP traffic is highest in the middle of the day. Peak hours location is hence highly correlated with the typical user profile. Researchers and students mainly connect to the Internet during the day while dialup users mostly in the evening.

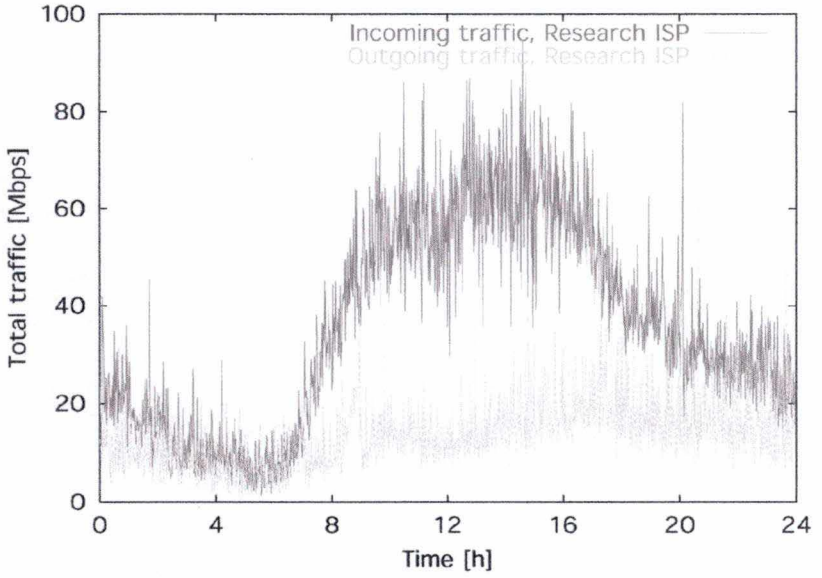

Figure 5.3: Daily traffic evolution for research ISP

5.4 Flow Establishment Techniques

Two main techniques have been proposed in the literature to establish flows (e.g. ATM VCs or MPLS LSPs) to carry TCP traffic (see [FRC98], [KLP+95], [NEK+99], [NML98] and [SRS99]). With a "traffic-based" flow establishment mechanism, a new flow is established once traffic is seen on a particular path. Such mechanisms were initially proposed in LAN environments to better integrate TCP/IP in ATM switches. In this case, one ATM VC was established and released for each (TCP or UDP) flow. Although this mechanism has been studied in LAN environments, it has not been analyzed in details in large networks where the amount of flows may be too large. With a "topology-based" flow establishment mechanism, LSPs are created based on the network topology. A typical example of the utilization of such mechanisms is backbone ISPs where LSPs would be established between all pairs of border routers. Although this mechanism has been recommended for internal use inside large networks, its utilization for interdomain traffic has not been studied in details.

5.4.1 Topology-based LSP Establishment Techniques

The topology-based LSP establishment technique is simple. It consists in associating to every entry of the routing table a dedicated LSP. In that case, every time the network topology changes, LSPs also change. If a routing entry is erased from the routing table, the corresponding LSP needs to be released. If a new entry is created in the routing table, a new LSP is established for that entry. With such a technique, every time a packet needs to be sent, the routing entry that matches the routing algorithm determines the LSP on which the packet will be sent.

Figure 5.4 provides the pseudo-code of the topology-based LSP establishment technique. It shows the operations required in order to send packets on LSPs. For each packet, we first search for the corresponding routing table entry. Then, depending on the existence of an LSP for the previously found routing table entry, either we directly find the LSP associated with the routing entry or we establish a new LSP for it. We then send the packet on the LSP.

```
foreach (packet) do {
      begin
      routing_table_entry = find_routing_entry(packet);
      if (exists_LSP(routing_table_entry))
      then {target_LSP = find_LSP(routing_table_entry);}
      else {target_LSP = establish_new_LSP(routing_table_entry);};
      send_packet(packet,target_LSP);
      end;
}
```

Figure 5.4: Pseudo-code for topology-based LSP establishment technique

The problem that arises with a pure topology-based approach is that every router needs to maintain as many LSPs as it has routing entries in its routing table. In addition, many routing entries could be unused for some quite long periods (several days). This approach requires many LSPs compared to the traffic-based approach. To reduce the number of LSPs that need to be established, one could use objects coarser than routing table entries. In our interdomain routing context, we could use aggregated routing entries instead of individual routing entries. For example, we could use routing entries that have a maximum prefix length. If we have many successive 24-bit long routing entries, it is possible to aggregate them into a shorter prefix entry (a 16-bit prefix for example) and associate LSPs to these more aggregated prefixes[22].

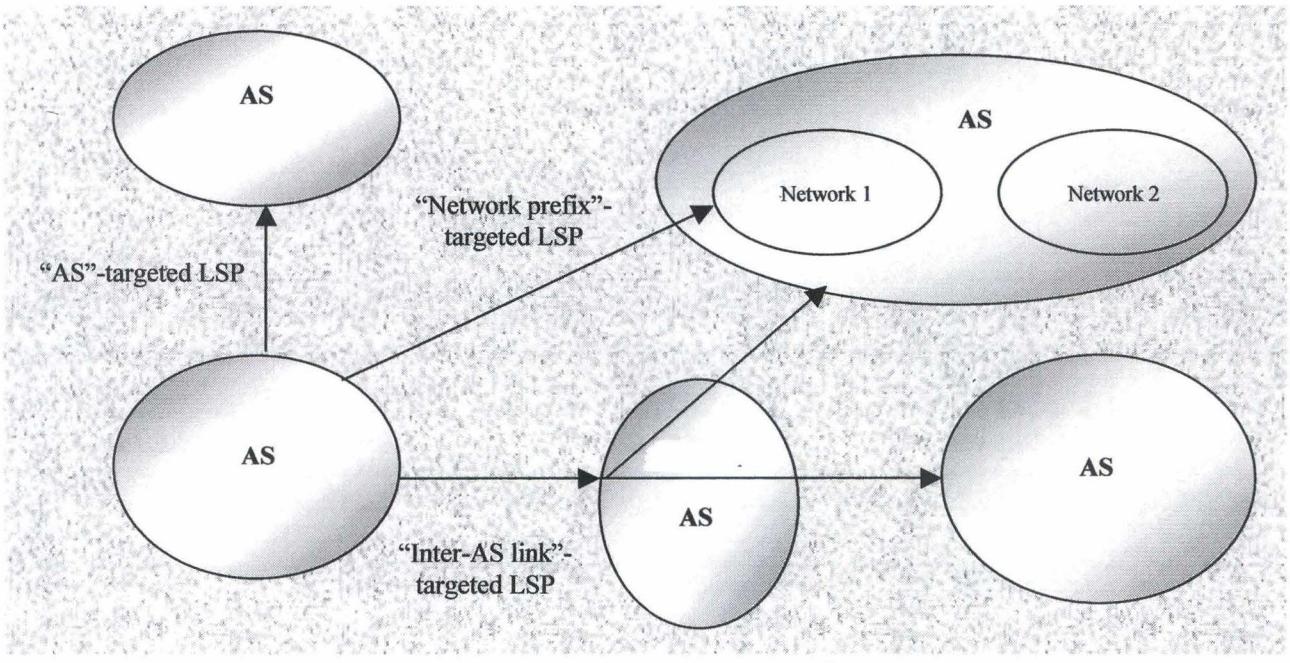

Figure 5.5: Interdomain topology-based LSPs

We could also use the information provided by the interdomain routing protocol. For example, BGP provides AS and inter-AS link information. AS stands for autonomous system and relates to a routing object much more aggregated than a simple routing entry. An AS represents a set of networks under a single administrative authority. Therefore, it is generally a more aggregated object than a particular routing entry. The concept of an "inter-AS link" is specific to BGP because BGP provides for every routing entry the complete path in terms of ASs to attain the destination. An inter-AS link is defined as

[22] This is true only if we assume that all network prefixes we aggregated are topologically located in the same area.

two different and strictly consecutive ASs appearing on an AS path of a BGP table entry. The concept of inter-AS link provides a more aggregated object than an AS because the same inter-AS link can be used to attain many different ASs. Figure 5.5 illustrates the topological difference between LSPs based on a routing entry, an AS or an inter-AS link. Interdomain LSPs can thus be "network prefix"-targeted, "AS"-targeted or "inter-AS link"-targeted.

5.4.2 Traffic-based LSP Establishment Techniques

While the topology-based approach probably requires maintaining too many LSPs, the traffic-based LSP establishment scheme relies only on actual traffic to establish LSPs. LSPs are thus established when traffic is seen for a particular traffic flow. A traffic flow may be an end-to-end flow or any "network topology"-based flow (between network prefixes, ASs...). Traffic-based techniques often use some timeout to release a LSP. When a packet needs to be sent, one has first to determine the LSP associated with the packet. The result of this operation depends on the granularity used for the traffic flow. If an LSP already exists for the traffic flow, the packet is sent on the corresponding LSP. If no LSP already exists for that traffic flow, a LSP needs to be established before sending the packet. Because traffic does not flow all the time over a particular LSP, its lifetime is limited according to a specific timeout. The timeout can rely on a configurable idle period during which no traffic is seen. Some specific information provided by the flow (application, transport protocol...) may also serve as an indication that the flow should be released.

Figure 5.6 provides the pseudo-code for a generic traffic-based establishment scheme. It illustrates the operations required to send a packet using a generic traffic-based LSP establishment technique. We first determine the flow associated with the packet to send. Depending on the existence of an LSP for that flow, either we find an LSP on which to send the packet or we establish a new LSP.

```
foreach (packet) do {
      begin
      flow = determine_flow(packet);
      if (exists_LSP(flow))
      then {target_LSP = find_LSP(routing_table_entry);}
      else {target_LSP = establish_new_LSP(routing_table_entry);};
      send_packet(packet,target_LSP);
      end;
}
```

Figure 5.6: Pseudo-code for generic traffic-based LSP establishment technique

The main issues that arise with traffic-based LSP establishment techniques relates with the LSP's lifetime and traffic variation over an LSP. Because traffic flows can be very dynamic, the overhead associated with pure traffic-based establishment schemes can be important. The other problem with traffic-based techniques concerns the traffic volume variations experienced by a particular LSP. The following section tackles these problems.

5.4.3 Hybrid LSP Establishment Techniques

All LSP establishment techniques seen so far have important problems. Topology-based techniques essentially require maintaining too many simultaneous LSPs. Traffic-based techniques on the other hand exhibit a behavior too dependent on the traffic flows dynamics (in volume and duration). This is

why "hybrid" establishment techniques are required. Hybrid LSP establishment techniques leverage the respective advantages of the topology-based and the traffic-based techniques. The two techniques on which we can rely to reduce the number of LSPs are "aggregation" and "triggering". Aggregation consists in multiplexing end-to-end flows to obtain LSPs carrying more traffic. For example, end-to-end flows can be aggregated based on the routing table information. End-to-end flows can become network-prefix-to-network-prefix LSPs. ASs or shorter network prefixes can also serve to aggregate flows. Triggering is a technique that imposes a constraint on the traffic volume. This technique is due to filter small flows that would require the use of a LSP for an insignificant traffic volume.

Neither aggregation nor triggering is expected to provide good results alone. However, combining both techniques could provide LSPs that are stable topologically and in terms of traffic volume. The remaining of this section thus presents algorithms implementing hybrid LSP establishment techniques.

The hybrid technique we present in this section use topology-based flows (between network prefixes, ASs or for an inter-AS link). The variable we call "flow" in the following pseudo-code represents any topology-based aggregation object.

```
foreach (packet) do {
    begin
    flow = determine_flow(packet);
    if (traffic_seen_during_last_interval(flow) ≥ trigger)
    then {begin
          if (exists_LSP(flow))
          then {target_LSP = find_LSP(flow);}
          else {target_LSP = establish_new_LSP(flow);};
          send_packet(packet,target_LSP);
          end;
    }
    else{begin
         if (exists_LSP(flow))
         then {release_LSP(flow)};
         send_packet_IP_routing(packet);
         end;
    };
    end;
}
```

Figure 5.7: Pseudo-code for constant trigger hybrid LSP establishment scheme

The hybrid LSP establishment technique uses a constant trigger filtering flows based on the traffic volume seen during the last time interval. Figure 5.7 provides the pseudo-code of such a technique. For a LSP to be established or maintained, at least trigger bytes need to be seen for the flow during the last time interval. Every time a packet needs to be sent over a LSP, the scheme first checks whether a LSP exists for that flow. If so, the trigger is applied against the LSP to verify that more than trigger bytes have been seen for that LSP during the last time interval (the past seconds corresponding to the time interval for example). If no LSP exists for the flow, we check whether enough traffic has been seen to establish a new LSP. If no LSP has been found or established for the flow, the packet is sent as best-

effort traffic by means of classical IP routing. If a flow already has a LSP and the traffic seen for that flow during the last time interval is less than trigger bytes then the LSP is released.

The previous technique could be enhanced by changing the length of the time interval. This time interval could depend on the flow type, on the amount of traffic seen for the flow so far, on the current LSP duration... The trigger could also be parameterized by the flow type or any other property of the LSP that is due to change its variability (in duration and traffic volume). Many optimizations may be considered to enhance the performance of the establishment scheme. However, the two main criteria are the lifetime of the LSP and the percentage of traffic carried on the LSPs. Traffic dynamics implies that LSPs duration could be short so that the number of establishment and release operations make the cost of the scheme prohibitive. The other performance criterion relates with forwarding performance. As previously mentioned, the main advantage of using LSPs is forwarding speed. Carrying traffic on an LSP only requires a lookup in the switching table while classical IP routing requires many additional lookup operations in the IP routing table. This implies that carrying more a higher percentage of the traffic on LSPs relates with better forwarding performance.

5.5 Switching Interdomain Flows

While section 5.3 presented the broad characteristics of the traffic traces in terms of the evolution of total traffic, this section studies the implications of using interdomain LSPs to carry best-effort traffic like the one of the traffic traces introduced in section 5.2. This section is structured as follows.

We first study the number of simultaneous LSPs and their signaling overhead. We then discuss the implications of using aggregation techniques to reduce the number of simultaneous LSPs. After that, we present the impacts of applying a trigger on LSPs in terms of traffic carried on the LSPs and the corresponding signaling overhead. The problems related with LSP duration are also briefly discussed.

5.5.1 Simultaneous LSPs

This section studies the implications of using interdomain LSPs in terms of variability. We assume that every packet is sent on a "network prefix "-targeted LSP. We use a hybrid LSP establishment technique with aggregation at the network prefix level, a traffic trigger of 1 byte and a time interval of one minute. Because *Netflow* does not provide information at the packet level, we were obliged to consider full one-minute intervals to determine whether a LSP was active or not. We thus modified the hybrid LSP establishment scheme presented in section 5.4.3 to work on one-minute interval traffic rather than incoming packets time granularity. While the hybrid LSP establishment scheme presented in section 5.4.3 considers the traffic seen during the last time interval for the flow, this section uses one-minute time intervals that begin at the same second for all LSPs. It means that LSPs are established for at least one minute. Seeing one byte towards a network prefix suffices to consider the corresponding LSP active. We study the total number of active LSPs and the corresponding signaling overhead (establishment and release operations).

Figures 5.8 and 5.9 show the evolution of the total number of active LSPs during one day for both ISPs. Daily traffic evolution seems to constitute a significant factor for explaining the evolution of the total number of active LSPs. All curves exhibit a behavior close to the one of total traffic evolution. However, the dialup ISP and the research ISP have quite different variations in the total number of active LSPs. The number of active LSPs for the dialup ISP remains quite stable during the whole day, even if the influence from the total amount of traffic appears.

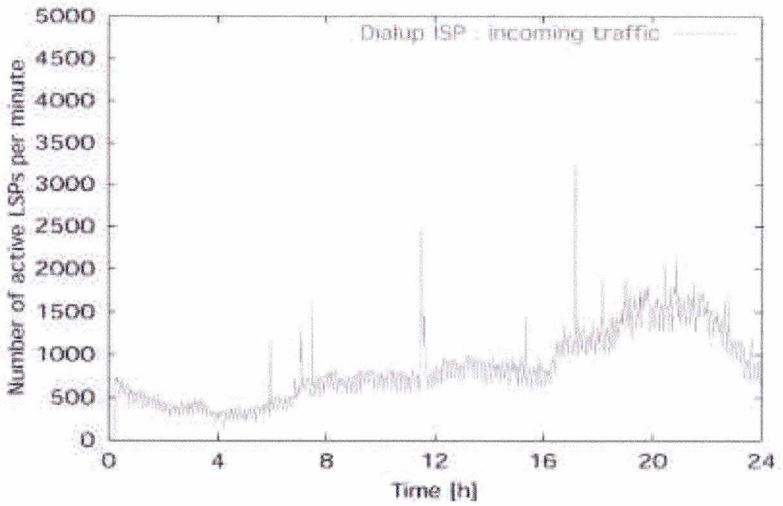

Figure 5.8: Number of active prefixes for dialup ISP

The research ISP experiences wide variations in the number of simultaneous active LSPs. This difference in LSPs number variations comes from the limited bandwidth of the dialup ISP, which does not allow many interdomain flows to be established at the same time. The very important link capacity available for the research ISP on the other hand allows many simultaneous flows to be active at the same time. This is why the research ISP exhibits a behavior more similar to its total amount of traffic.

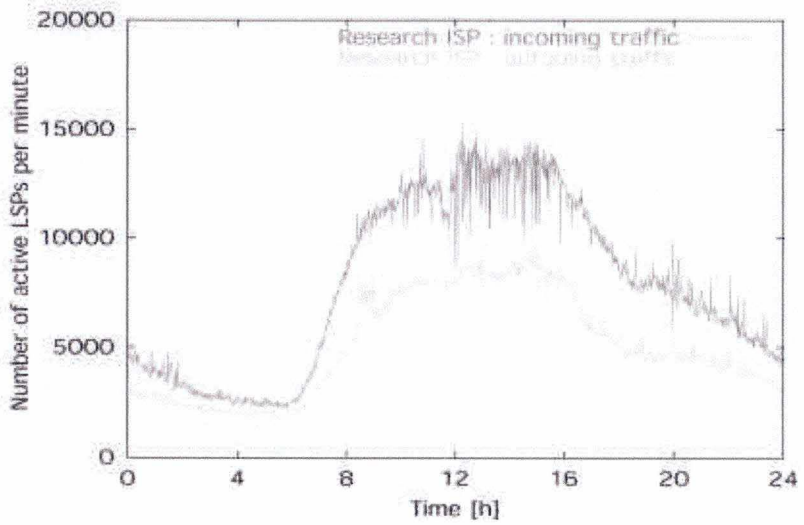

Figure 5.9: Number of active prefixes for research ISP

We now consider the signaling overhead generated by LSPs establishment and release operations. Figures 5.10 and 5.11 illustrate the evolution in the number of signaling operations (establishment and release) on a per-minute basis for the dialup ISP and the research ISP respectively. When comparing with absolute numbers of active LSPs, the signaling overhead appears substantial even if the ratio of signaling operations divided by the total active LSPs decreases when the total number of active LSPs increases.

Signaling overhead for the dialup ISP represents about 50 % of the number of active LSPs. This means that when comparing two consecutive minutes, only 50 % of the interdomain flows (active network prefixes) are identical. Such flows hence have a very short lifetime considering the fact that they apply to interdomain flows, not end-to-end flows.

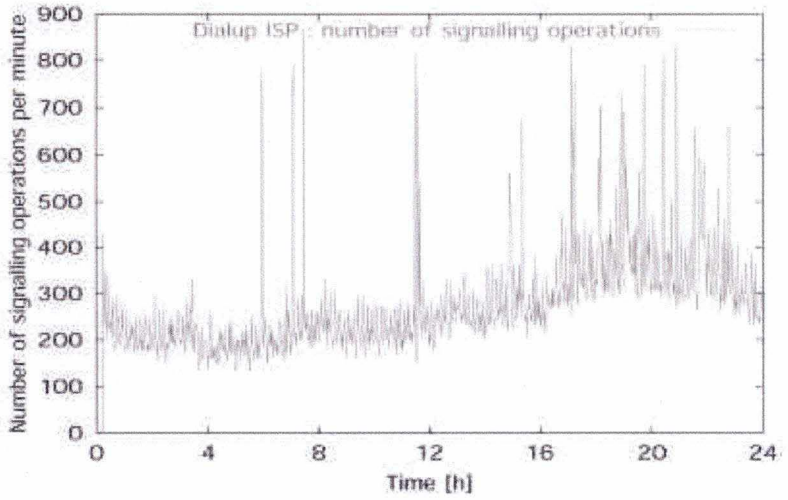

Figure 5.10: Signaling overhead for dialup ISP (prefixes)

Signaling overhead for the research ISP shows some better results. It remains well under 20 % of the total number of active LSPs during peak hours. This implies a better stability for interdomain flows. Remark the low variation of signaling overhead compared to the one of the total number of active LSPs. It means that signaling overhead does not depend too much on the number of active LSPs.

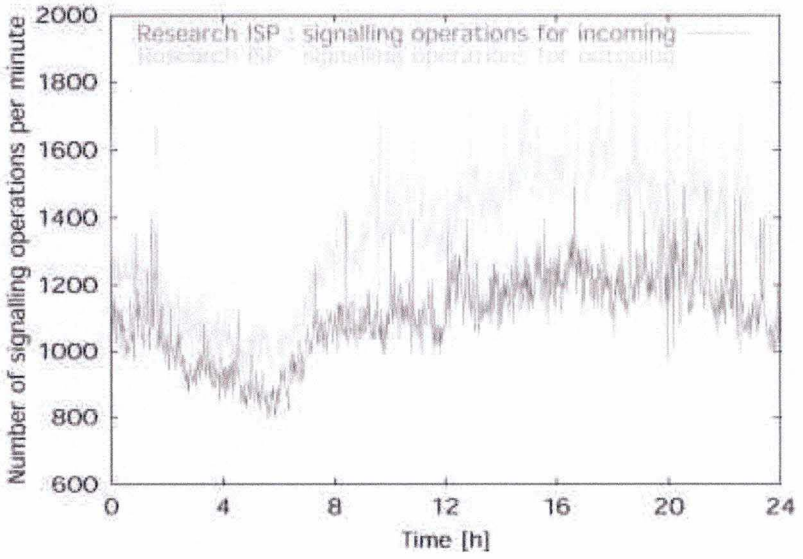

Figure 5.11: Signaling overhead for research ISP (prefixes)

5.5.2 Aggregation Techniques

We could consider several solutions to lessen LSPs number and signaling overhead. For example, working on ASs-targeted LSPs instead of network prefixes leads to a substantial cut in the number of LSPs and signaling operations. Figure 5.12 shows daily evolution in the number of active ASs for the research ISP. Using ASs instead of network prefixes clearly reduces the number of simultaneous LSPs from several thousands to less than one hundred simultaneous LSPs. As for all LSP establishment techniques, daily traffic evolution significantly influences LSPs evolution.

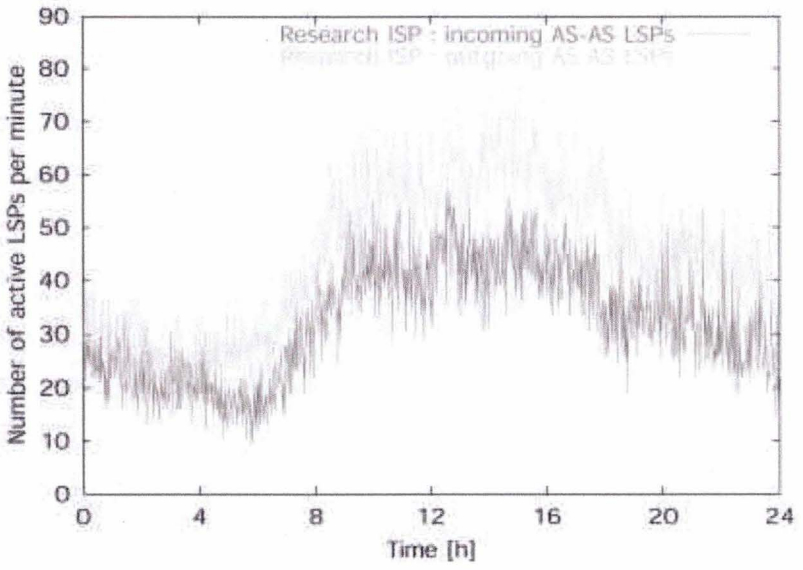

Figure 5.12: Number of active ASs for research ISP

The cut in the number of active AS-targeted LSPs is much more impressive in the case of the dialup ISP where it stays always below 30. Unfortunately, using ASs does not provide stability for what concerns signaling overhead.

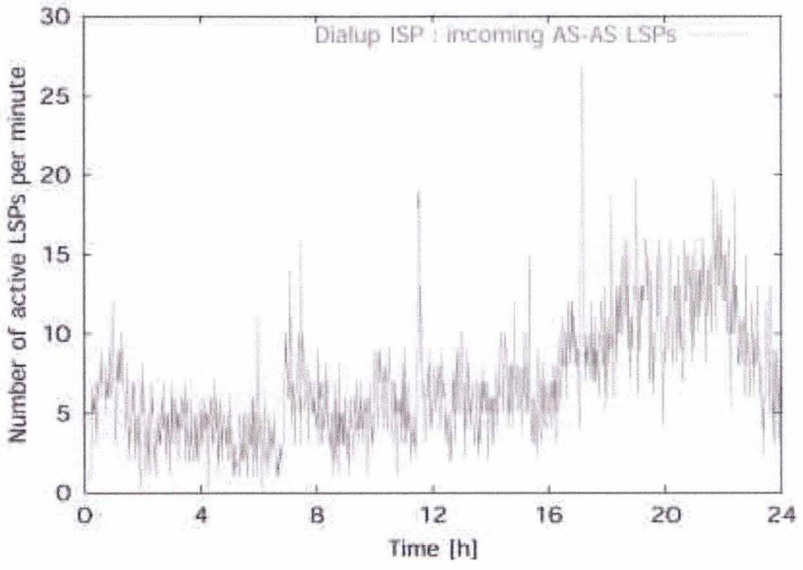

Figure 5.13: Number of active ASs for dialup ISP

Figure 5.14 shows the signaling overhead for AS-targeted LSPs and for the dialup ISP. It proves that using coarser LSPs does not resolve the signaling problem. The magnitude of the problems remains the same with a signaling overhead representing about 50 % of the number of active LSPs. More aggregated LSPs do not exhibit a better stability. Such results belie our expectations since we would have expected that more aggregated LSPs be more stable.

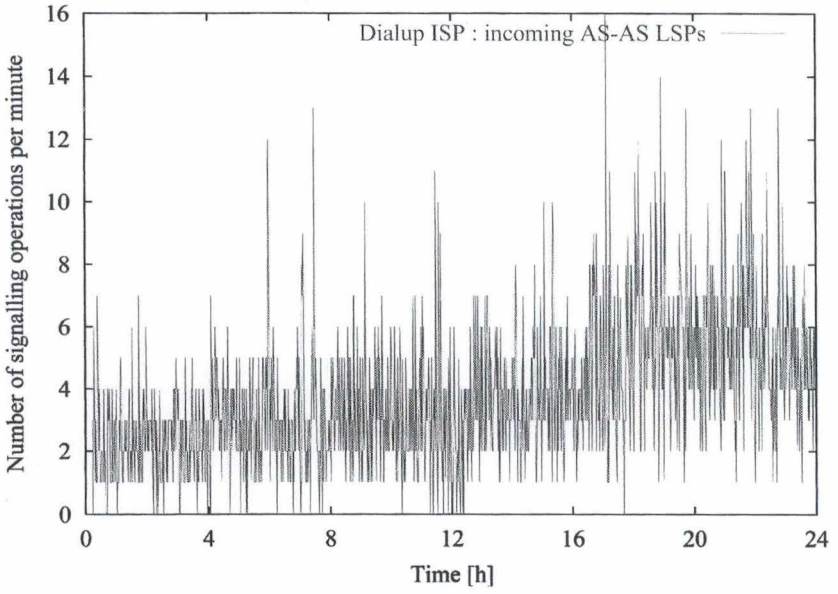

Figure 5.14: Signaling overhead for dialup ISP (ASs)

5.5.3 Triggered LSPs

Another solution to reduce the number of LSPs would be to divide the traffic into several "classes": low bandwidth flows handled as regular IP packets and high bandwidth flows carried over interdomain LSPs. Figures 5.15 and 5.16 show the difference between the number of network prefixes and ASs that are seen over the whole day and for incoming traffic. These figures also show the cumulative number of ASs and network prefixes as a function of the total traffic that has been seen for them during the whole day. While the overall gain at using ASs instead of network prefixes for LSP establishment seem convincing, it diminishes when considering ASs and network prefixes that carry more traffic over the day.

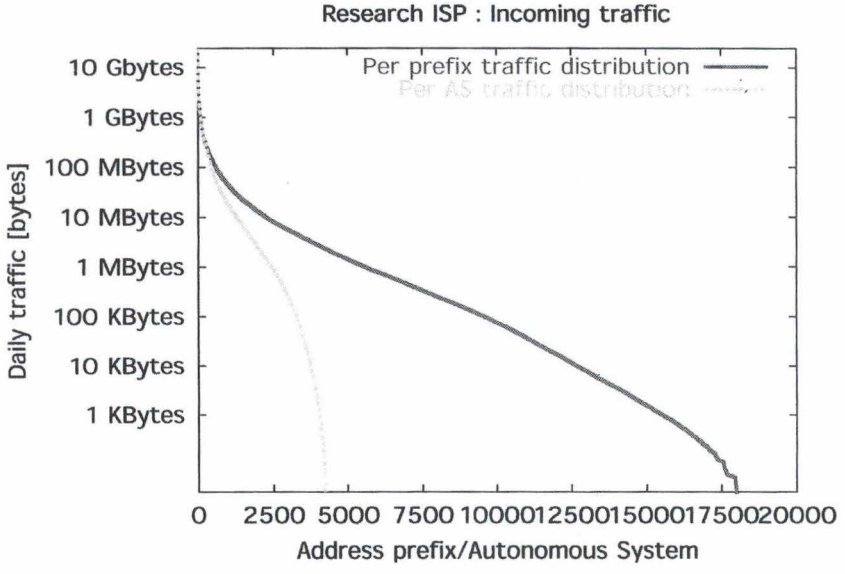

Figure 5.15: Daily "geographical" traffic distribution for research ISP

While Figures 5.15 and 5.16 exhibit a similar behavior for what concerns the total number of network prefixes and ASs, traffic carried by ASs and network prefixes over the day is quite different for the

research ISP and the dialup ISP. Each AS and network prefix among the top 2500 for the research ISP carry respectively more than 1Mbytes and 10 Mbytes of traffic during the day. On the other hand, the top 2500 ASs and network prefixes for the dialup ISP carry an amount of traffic around one order of magnitude inferior to the case of the research ISP. Such behavior is consistent with the difference in the total amount of traffic seen be each ISP.

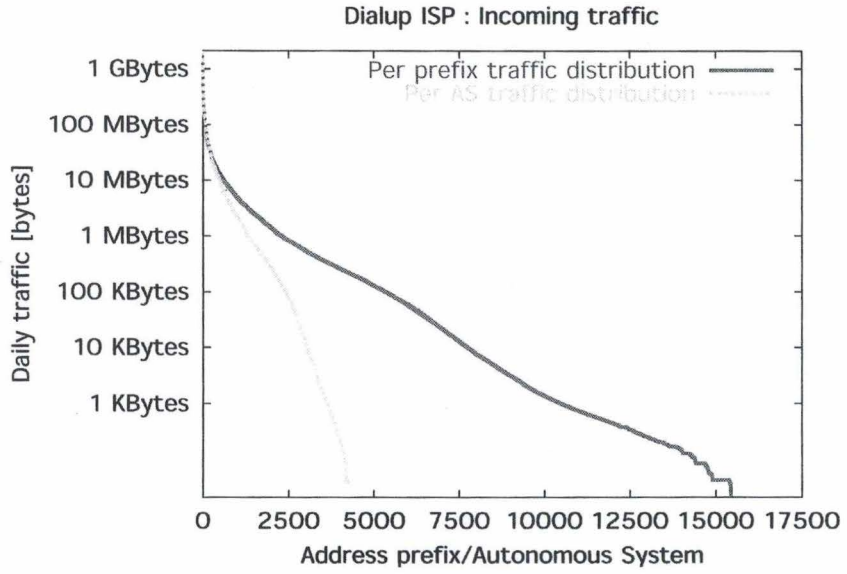

Figure 5.16: Daily "geographical" traffic distribution for dialup ISP

The two previous figures have shown the gain obtained from using ASs and network prefixes carrying more than a particular amount of traffic over the day in terms of the total number of ASs and network prefixes seen over the day. This requires to know before establishing a LSP whether the corresponding AS(s) or network prefix(es) will see an important amount of traffic over the day. The problem is that we do not have this information. What we know is the amount of traffic one LSP has already carried. This is why we now evaluate the performance of the hybrid LSP establishment technique presented in section 5.5.1. This technique uses "network prefix"-targeted LSPs and one-minute time intervals that begin at the same time for all LSPs. We study the performance of such a scheme in terms of the percentage of traffic carried on these LSPs and the fraction of signaling operations required by comparing different values of the trigger used to filter the flows.

Figures 5.17 and 5.18 display the fraction of traffic carried by LSPs and the corresponding fraction of signaling operations for the incoming traffic of the dialup and the research ISP. The fraction of traffic carried represents the ratio of the traffic carried over LSPs (over the whole day) established for a particular trigger (x-axis) divided by the total traffic seen during the day. The fraction of signaling operations gives the number of signaling operations on LSPs required when using a trigger of x bytes compared with the signaling overhead of the same establishment scheme with a 1-byte trigger (the scheme used in section 5.5.1).

The first interesting fact appearing from the traffic capture information is that using a trigger does not significantly influence the capture until a certain point. The research ISP (Figure 5.17) shows a capture superior to 95 % until the 10 Kbytes trigger. After the 10 Kbytes trigger, the traffic capture starts to slowly[23] decrease to attain a little more than 25 % at the 10 Mbytes trigger. A trigger superior to 10 Kbytes thus seems ideal. The second interesting fact concerns signaling that decreases from the beginning to represent about 30 % of the initial overhead at the 10 Kbytes trigger. The combination of traffic capture and signaling overhead is positive since an important trigger does not reduce too much

[23] Be aware that what we call "slow" is not linear because Figures 5.17 and 5.18 have an x-axis with a logarithmic scale.

the traffic that is carried over the LSPs while at the same time the signaling overhead decreases. It calls for the use of an important trigger. If we had to use a trigger for the research ISP, we would probably take a value of more than 10 Kbytes.

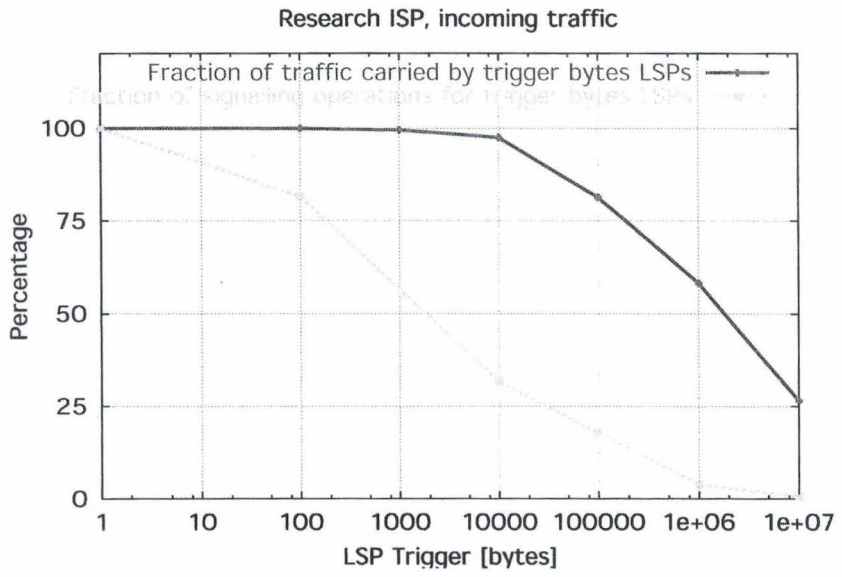

Figure 5.17: Impact of LSP trigger for research ISP incoming traffic

The case of the dialup ISP is similar but only if we look at trigger values below 10 Kbytes. The problem with the dialup ISP is that its customer profile coupled with its limited access link capacity limits the size of the LSPs. Big data transfers like in the case of the research ISP are not possible for dialup customers. This is why the fraction of traffic carried by LSPs triggered by more than 10 Kbytes decreases very quickly. While the research ISP has more than 50 % of traffic captured by LSPs bigger than 1 Mbytes, the dialup ISP is already below 10 % of traffic capture. However, the 10 Kbytes trigger seem to be a constant for both ISPs. This value gives the best results when considering both traffic capture and signaling overhead.

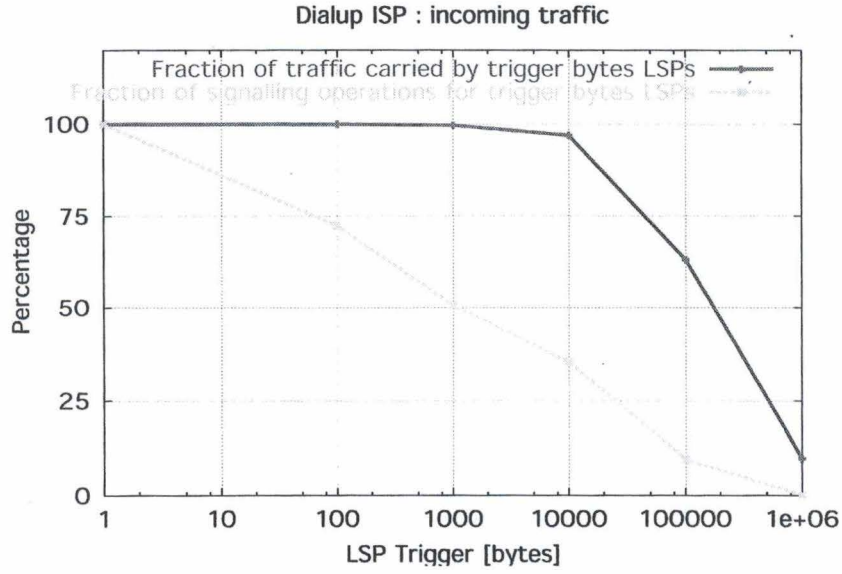

Figure 5.18: Impact of LSP trigger for dialup ISP incoming traffic

5.5.4 LSPs Duration

Now that we have studied the performance of applying a trigger with the hybrid LSP establishment scheme, one interesting question one could ask relates to the possibility such a simple technique could leave on the ability to rely on long-lived LSPs to carry an important fraction of total traffic.

One could have thought before looking at Figure 5.19 that "big-traffic" flows last longer than small traffic ones. Figure 5.19 shows the cumulative percentage of traffic carried on LSPs as a function of LSP duration for our hybrid LSP establishment scheme and for the research ISP. Different values of the trigger are compared. Simply having shown the 0 byte trigger on Figure 5.19 could raise into one's mind the idea that almost 80 % of the flows have a lifetime longer than 15 minutes. Of course, these "0-byte"-triggered flows are long-lived but considering flows constrained by a 0-byte trigger is comparable to considering every single packet as a flow! Interdomain flows should be made of aggregated traffic, not by flows of a few bytes per second. The other curves give results more representative of the LSP duration (for trigger values of 10 Kbytes, 100 Kbytes and 1 Mbytes). The 1 Mbytes trigger curve shows that a little less than 40 % of the traffic is made of big traffic flows lasting less than 1 minute. Hence, big interdomain traffic flows are rather short-lived. This constitutes quite important an issue because such bursty behavior would imply the inability to provide any realistic QoS guarantee to the corresponding traffic. Even if this traffic were carried over already established LSPs, traffic variations of this importance would not enable reacting quickly enough to ensure a low packet loss rate inside the intermediate routers. Of course, topological aggregation could ensure a good multiplexing of these big flows. The main issue concerns the stability of big traffic flows. If we could rely on some important percentage of the traffic that would exhibit some stability, we would be able to balance smaller flows to complement big flows. This would allow providing stability and at the same time capturing a substantial part of the traffic. Note that the 1 Mbytes trigger curve has its percentage of traffic ranging between a little less than 40 % to less than 60 %. Roughly, long-lived flows carrying more than 1 Mbytes per minute represent about 10 % of the total traffic. This number seems too small to use an aggregation technique based on big-traffic flows.

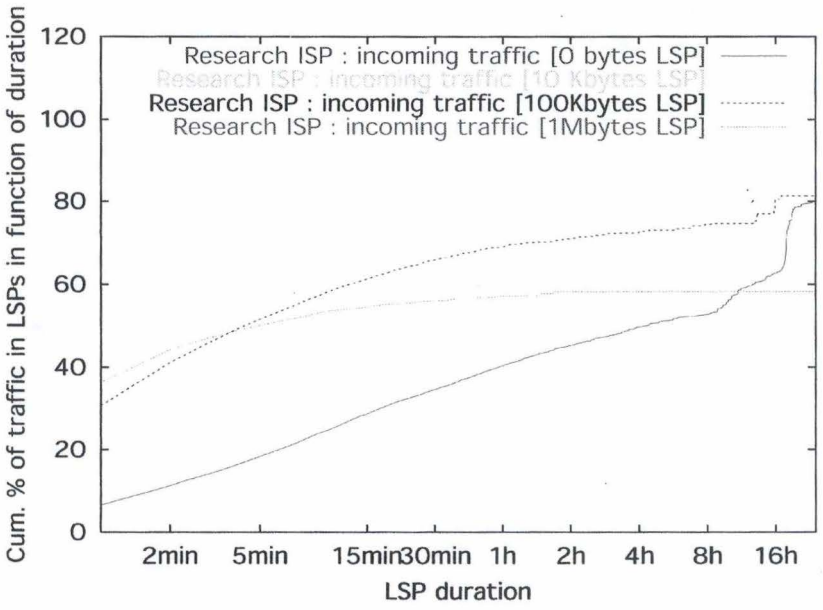

Figure 5.19: LSP lifetime for research ISP incoming traffic

Figure 5.20 displays the cumulative percentage of traffic carried in LSPs in function of LSP duration for the dialup ISP. The 1-byte trigger curve shows that using a pure MPLS solution (switching all packets) would make LSPs lasting more than 5 minutes capture about 80 % of the total traffic. Five minutes is probably a too short duration for interdomain LSPs. However, if we consider LSPs lasting more than

30 minutes, they capture only 47 % of the total traffic. If in addition we constrain these LSPs by the 100 Kbytes trigger, only 15 % of the total traffic remains.

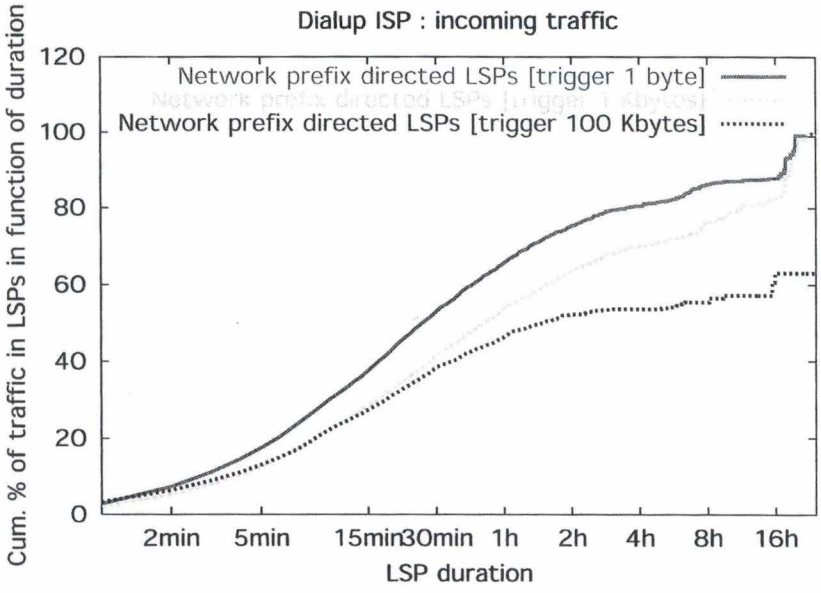

Figure 5.20: LSP lifetime for dialup ISP incoming traffic

The main difference between the research ISP and the dialup ISP concerning LSP duration appears for LSP duration inferior to 30 minutes. While the curves for the research ISP begin at quite high percentages of traffic, this is not the case for the dialup ISP. The "big trigger" curves for the research ISP begin around 40 % of traffic while all curves for the dialup ISP start near 0 %. This happens because the research ISP has almost unlimited bandwidth so that LSPs can carry an important amount of traffic. On the other hand, the dialup ISP does not permit big flows to be established because its customers mainly use modems. Most flows of the dialup ISPs are thus long-lived while they are rather short-lived for the research ISP. Another interesting characteristic that both ISPs have in common is the implications on the traffic capture of using big triggers. Not only relying on important trigger values for the research ISP makes the LSPs shorter but it also limits the ability to capture an important part of the traffic (80 % for 10 Kbytes and 100 Kbytes, 60 % for the 1 Mbytes trigger) because most flows do not carry much traffic. This inability to capture an important part of the traffic with big traffic LSPs also appears for the dialup ISP. However, the dialup ISP does not suffer from important trigger values for what concerns LSP duration. It can be viewed on Figure 5.20 for small values of the LSP duration. This phenomenon finds its cause in the "dialup property" of the customers. Constraining LSPs to big traffic flows only does not change anything for the dialup ISP because of its quite small bandwidth capacity. Aggregating many dialup customers traffic does not increase access link capacity anyway so that LSPs are rather long-lived for the dialup ISP.

The difference between the two ISPs with regard to the LSP lifetime can probably be explained by the differences between the two networks. Most of the incoming traffic of the dialup ISP is received through its two transit ISPs. The links to these transit ISPs are typically heavily congested during peak hours. Furthermore, the typical user of the dialup ISP receives data via a modem, which severely limits the interdomain capacity that a single user can utilize. On the other hand, the external links of the research ISP, especially those towards the research networks and the interconnection points, experience less congestion than the links of the dialup ISP. In addition, the typical user of the research ISP can easily receive interdomain traffic at several MBPS. These two factors explain the smaller duration of the interdomain LSPs of the research ISP.

Consequently, applying a trigger on interdomain flows does not provide a satisfying solution to the LSP duration problem. Traffic-based establishment techniques do not allow solving the traffic capture issue.

Be aware that the scheme we used exhibits idealistic results since flows can be instantaneously established and released. Actually, the first minute of flow's life captures about 15 % of total traffic. Such results lead to the conclusion that LSP establishment techniques should use topology-based management to increase big-traffic LSPs duration. Because we relied on BGP network prefixes (best-matching prefixes), LSPs might have been too fine-grained. Aggregating LSPs at the AS or inter-AS link level (see Appendix 6 for the definition of these concepts) could reduce LSPs variability. We studied such aggregation levels without being able to resolve the problem. The main issue encountered with interdomain traffic resides in the variability in volume within a particular LSP. The next section evaluates two bandwidth reservation techniques for LSPs, showing how deep the LSP traffic volume variability issue really is.

5.6 LSPs with a Guaranteed Bandwidth

Most of the papers dealing with LSP establishment mechanisms have mainly studied the instantaneous number of LSPs passing through one switch (or router). This is an important performance number to design switches. However, across interdomain boundaries, using lower cost links could use traffic engineering to optimize traffic. In this case, the cost associated with one LSP will always be an important factor to optimize. In some cases, e.g. for voice traffic, factors such as delay will be equally important.

If billing of interdomain flows were based on the total volume of traffic sent on a particular flow, then LSPs could remain active even if they did not carry any traffic. We do not expect that to be the only way to use LSPs. Another way for resource usage billing could rely on associating a minimum guaranteed bandwidth with each flow and to bill the flow according to the amount of reserved resources (i.e. bandwidth x duration). In that case, an important issue would be to determine the bandwidth required by a flow at LSP establishment time as well as during its lifetime.

The purpose of this section therefore consists in evaluating the possibility one has to use LSPs with bandwidth guarantees. The question we try to answer is the one of the performance of bandwidth reservations for interdomain LSPs. The performance aspects considered are the ratio of traffic carried on these "guaranteed bandwidth LSPs" and the ratio of bandwidth that has been wasted.

We define the ratio of traffic carried by

$$Capture = \frac{Traffic\ carried\ on\ "guaranteed\ bandwidth\ LSPs"}{Total\ traffic\ seen\ for\ all\ flows\ during\ the\ day}.$$

We define the ratio of bandwidth that has been wasted by

$$Waste = 1 - \frac{Traffic\ carried\ on\ "guaranteed\ bandwidth\ LSPs"}{Total\ bandwidth\ reserved\ for\ "guaranteed\ bandwidth\ LSPs"}.$$

Figure 5.21 presents the results of using a fixed reservation for each LSP for incoming traffic of the research ISP. The rules for determining whether a flow is active or not are the same as previously, i.e. hybrid LSP establishment scheme with "network prefix"-targeted LSPs and a one minute time interval starting at the same time for all LSPs. By fixed reservation, we mean that the flow can use an "amount of bandwidth" equivalent to "reservation" bytes per minute (x-coordinates on Figure 5.21). Capture measures the performance of the reservation technique in terms of the percentage of total traffic that could benefit from a guaranteed bandwidth. Waste measures the performance of the reservation

scheme in terms of the ratio of resources that have been actually used for all LSPs, given that a high waste relates to poor performance.

Results shown by Figure 5.21 are quite astonishing. Trying to capture any interesting percentage of traffic requires a high wasting of the reservations. This is due to the high burstiness of best-effort traffic that has very short peaks with long periods of low traffic with an overall mean that is very low compared to the peaks. Trying to capture more than 50 % of total traffic requires the waste of more than 90 % of the reserved resources. Such a high waste percentage implies that fixed reservations do not suit best-effort traffic. LSPs should thus be allowed to renegotiate their reservation to adjust themselves to their respective needs.

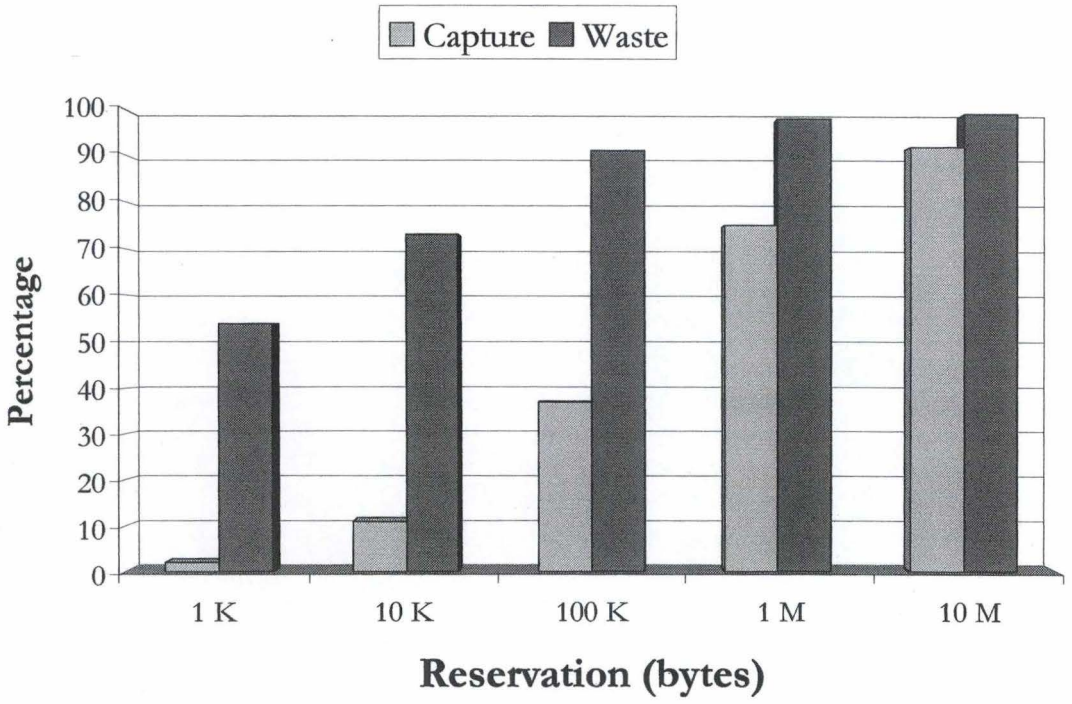

Figure 5.21: Performance of fixed reservation scheme

The discouraging results of the fixed reservation technique may be an artifact of the wide distribution in flow size. Each LSP might require a specific amount of traffic to be reserved in order to better match the reservation with the effective amount of bandwidth used. Figure 5.23 shows the performance of a reservation technique using a mobile mean over a parameterized period for incoming traffic of the research ISP. Instead of basing the amount of reservation on a constant number, we give the LSPs the opportunity to instantaneously renegotiate for every minute its amount of reservation based on the mean of the traffic seen for the flow during some past time interval. We call the latter time interval the "memory" of the scheme. The length of the period over which the mean is computed is a parameter. One can therefore adjust the memory of the scheme. If the flow is not active for at least "memory" minutes, the reservation equals the largest amount of traffic seen for the flow for the time it was active. We suppose that the newly established LSP carries all traffic during the first minute the flow is active. In addition, flows are constrained by a *trigger* filtering small traffic flows and enabling to study the performance of the scheme for "big-traffic" flows[24].

Figure 5.22 shows the pseudo-code of the mobile mean reservation technique. Reservations are made on a one-minute basis and renegotiation occurs every minute for every flow. When a flow passes the trigger constraint, it is granted a LSP and the corresponding guaranteed bandwidth.

[24] Big traffic flows are flows for which we need to see more than *trigger* bytes during the considered minute.

```
foreach (minute) {
    foreach (flow) do {
        if (traffic_seen_last_minute(flow) ≥ trigger) then {
        # Flow is considered as active
              if (last_active_periods(flow) ≥ memory) then {
              maintain_LSP(flow)
              reservation(flow) = mean_last_memory_periods(flow)
              }
              elseif (last_active_periods(flow) ≥ 1) {
              maintain_LSP(flow)
              reservation(flow) = max_traffic_since_active(flow)
              }
              else {
              establish_LSP(flow)
              reservation(flow) = traffic_seen_last_minute(flow)
              }
        }
        else {
        # Flow is considered as inactive
              if active(flow) then {release_LSP(flow)}
        }
    }
}
```

Figure 5.22: Pseudo-code for mobile mean reservation scheme

Figure 5.23 gives some insight about the "magnitude" of the problem: achieving a good capture percentage without over-reserving seem impossible. One could think that some more sophisticated technique could give better results. In reality, the problem is so severe than using sophisticated techniques is ineffective at the interdomain level. Roughly said, the best-effort paradigm is responsible for this ineffectiveness of reservation techniques. Allowing all sources to send as much traffic as they can generates a very important burstiness. At high levels of aggregation[25], interdomain traffic presents "self-similar" characteristics. Several other papers have already shown such a behavior for network traffic (see [PF95], [LTW+94], [TMW97] and [Kus99]).

The reason for the bad results shown on Figure 5.23 is displayed on Figure 5.24 where we see the "chaotic" behavior of the biggest traffic (incoming) network prefix of the research ISP. This network prefix has a total traffic over the day of 13.5 Gbytes or equivalently 1.25 MBPS during the whole day. See how the mean of 1.25 MBPS is far from being sufficient to have a chance of capturing an important part of the traffic. Even worse, the burstiness experienced by such a network prefix is not too severe since it transmits almost continuously during the day. Smallest traffic network prefixes exhibit a much more bursty behavior because they do not transmit traffic most of the time.

[25] This is true for all aggregation levels we used in our study: network prefixes, ASs or inter-AS links.

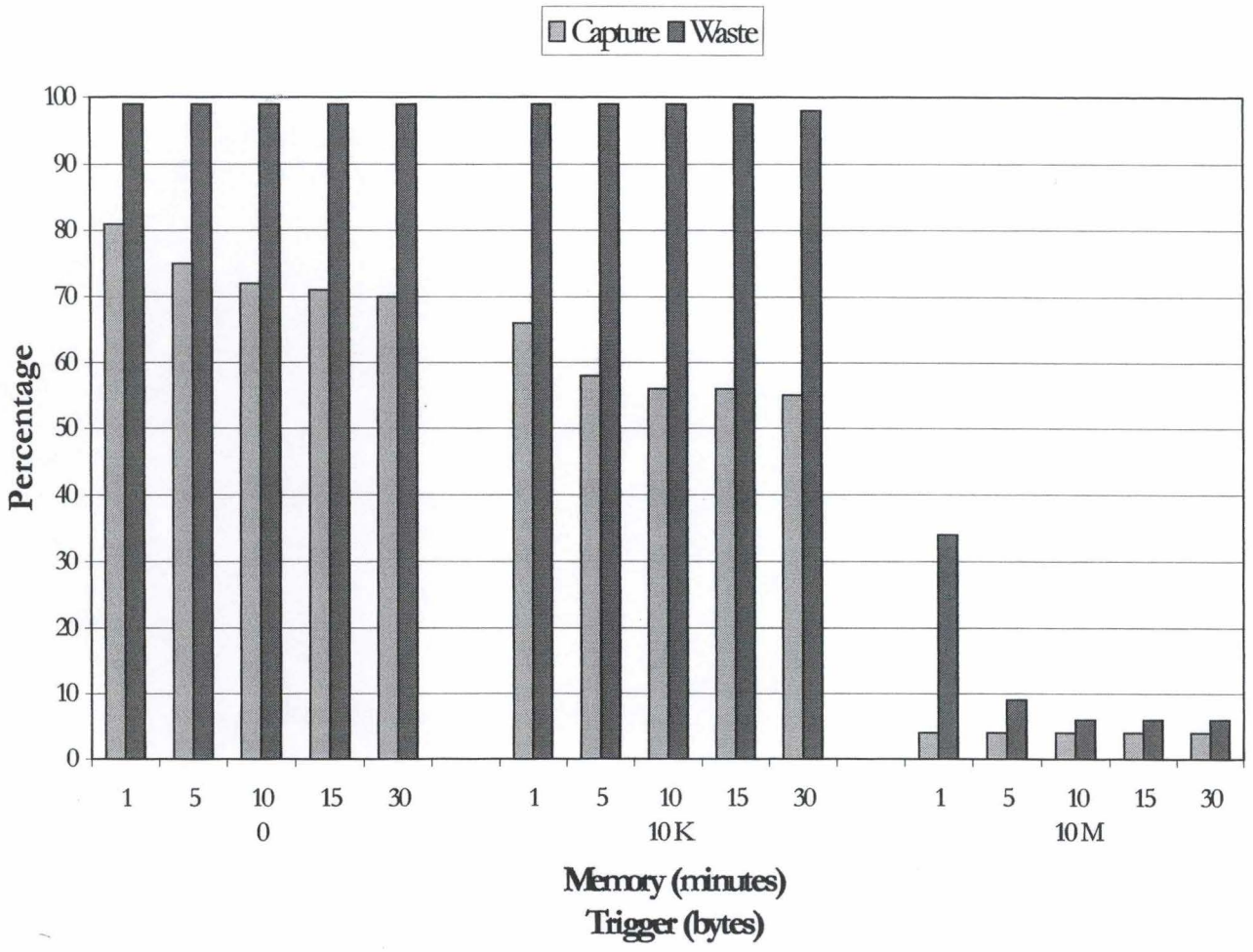

Figure 5.23: Performance of mobile mean reservation scheme

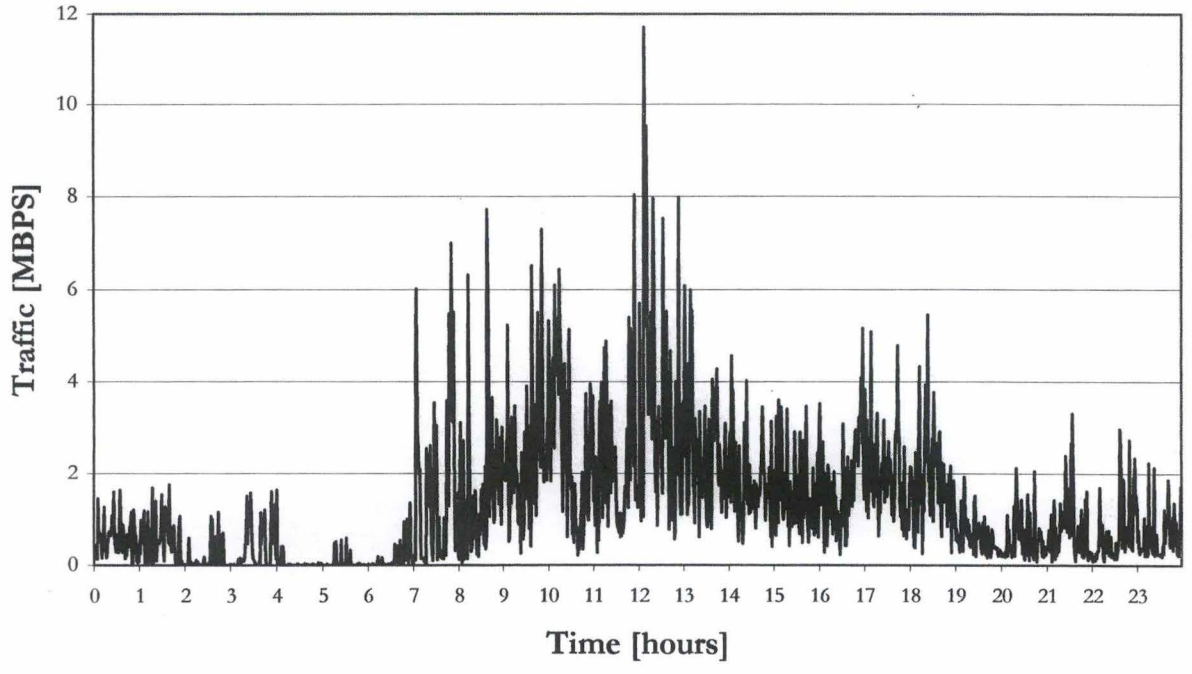

Figure 5.24: Daily incoming traffic evolution for "biggest" network prefix

A solution to this problem consists in shaping the traffic at the source-level. However, one has first to ensure the technique is effective through simulations. If "source shaping" does not reduce the burstiness at the interdomain aggregation level, changing the functioning of the TCP/IP stack might be necessary!

Another solution might be found in regression techniques performed for every LSP. Regression techniques (see [BD96]) allow predicting the forthcoming bandwidth needs of an LSP by using the traffic seen during some given past period. The main limitation of this scheme lies in its computational complexity. In fact, this technique requires the inversion of a matrix of order equal to the length of the past period on which the prediction relies. Performing a matrix inverse might be a somewhat unrealistic operation in a routing device. Even if most clever linear filters exist to predict bandwidth usage[26], they rely on the knowledge of the traffic distribution within flows. Unfortunately, such information is not known for best-effort data. However, real-time applications often provide enough information concerning internal flow traffic distribution (thanks to data compression techniques). [GGM+97] presents an evaluation of linear regression techniques that predict bandwidth usage for VBR data.

Some strange fact about the high level of burstiness experienced in our study concerns the absence of perception of it as a problem by the Internet community. The analysis of the reservation techniques has shown how deep the problem really is. This certainly constitutes more than just an issue when QoS are considered: "How could anyone provide any guarantee to specific flows when such bursts occur at high levels of aggregation?". When only "best-effort-like" guarantees need to be provided, loosing some traffic does not constitute a problem. Nevertheless, the arrival of applications needing strict end-to-end delay, jitter and loss guarantees could change many things in the way people think about source behavior and its implications on interdomain traffic variability.

5.7 Conclusion

We have seen throughout this chapter an analysis of interdomain traffic variability. We relied on traffic traces from two very different ISPs to minimize the chance of bias that would be due to the characteristics of a particular ISP. We first presented the traffic traces gathering environment and explained the limitations of our measurements. We then presented the broad characteristics of the two studied ISPs. We followed with the actual traffic traces analysis. After that, the broad characteristics of the daily traffic of both ISP have shown the influence of customer profile on peak hours location within the day. Evaluating the performance of interdomain LSPs to carry best-effort traffic required some introduction on the subject of LSP establishment techniques. We thus presented the two main LSP establishment techniques, topology-based and traffic-based, and explained their limitations. The limitations of the two main LSP establishment techniques drove the need for a new scheme that we call "hybrid" LSP establishment technique. This scheme permits to combine the advantages of the two main establishment techniques while at the same time minimizing the signaling overhead. This somewhat theoretical introduction allowed us to evaluate the performance of the LSP establishment techniques on the actual traffic traces. We studied the performance of using LSP to carry interdomain traffic in terms of the total number of LSPs and signaling overhead (LSP establishment and release operations). We studied aggregation techniques as well as triggering techniques in order to reduce the LSPs variability. It showed us the many problems that would appear if MPLS were used with best-effort traffic like the one we currently have in the Internet. After that, we tried to perform bandwidth reservations for interdomain LSPs to see whether guarantees could be provided. We first evaluated a simple reservation scheme by using a fixed amount of bandwidth to every LSP. The bad results of the previous reservation technique obliged us to use a more sophisticated reservation scheme based on a mobile mean over a parameterized time interval. LSPs were allowed to renegotiate their bandwidth

[26] Linear filters remove the need of a matrix inversion by using a particular function that requires to only maintaining the last value of the function and the value of the traffic experienced during the last time interval. However, finding the right function relies on the knowledge of the particular domain of application (see [BD96]).

reservation by using the mean traffic over a parameterized time interval. Even this technique did not provide satisfying performance so that somewhat had to be wrong with traffic sources behavior. We then explained the reason of the problem: best-effort. Allowing the sources to send as much traffic as they could generates a level of burstiness such that interdomain traffic presents "self-similar" characteristics. The problem with "self-similarity" concerns the fact that traffic exhibits a "chaotic" behavior at all levels of aggregation, so that trying to perform multiplexing on LSPs will not work. Finally, we proposed several solutions to the traffic variability problem, namely traffic shaping, regression techniques and modification of the TCP/IP stack. Among the previous solutions, only the first seem to have some future. Traffic shaping is an already used technique. Regression is far too complex to implement in routing devices and modification of the TCP/IP stack implies too many problems to be considered.

5.8 Further Considerations

Be aware that the analysis we presented in this chapter is a summary of what has been actually performed on the traffic traces. We showed only the most interesting facts in order to provide the reader with an overview of interdomain traffic characteristics without giving an overwhelming amount of numbers that would rather disturb than inform. While the guaranteed bandwidth part of our analysis may have been perceived too strange to be true, many existing studies confirm our results. Several papers have shown the self-similar nature of network traffic: [LTW+94] for Ethernet traffic, [PF95], [TMW97] and [Kus99] for Wide-Area traffic. However, we are not aware of any study about ISP-centered interdomain traffic analysis. All wide-area traffic studies relate to TCP connection evaluation, packet size distributions and protocol distribution within traffic.

Our work tried to look at the traffic characteristics using an ISP's point of view. Traffic engineering does not only concern backbones and transit service providers. Non-transit multi-homed ISPs also need to traffic engineer their access points. While traffic engineering objectives intend to balance traffic to provide guarantees for both incoming and outgoing traffic, we showed several problems that need to be resolved first, before thinking about network traffic optimization: LSPs variability in terms of establishment and release operations as well as traffic evolution within a given LSP.

Chapter 6

Conclusions and Future Work

This dissertation has presented an interdomain routing centered view of traffic engineering. We followed the successive steps of the evolution of IP routing and we have explained the limitations of IP routing that have driven the need for traffic engineering, also called network traffic optimization.

We have seen in Chapter 2 the control-related aspects of routing and switching. We have also discussed why classical IP routing cannot scale to the current Internet and why label switching is required to provide a scaleable routing function as well as to ensure better forwarding performance.

The emerging MPLS standard from Chapter 3 has then shown that the integration of layer 2 and layer 3 routing was possible, at least at a theoretical point of view. MPLS possesses many of the necessary features to enhance IP routing and at enabling the implementation of traffic engineering in IP networks. What is yet to determine is whether the Internet can make use of MPLS for traffic engineering purposes.

Chapter 4 has then provided a view of what traffic engineering looks like in the drafts and the specifications. Traffic engineering is only at an early stage of its development. Many things need to be achieved before being able to talk about effective traffic engineering. We evaluated the many approaches that have been developed in order to control the traffic flow as well as to provide guarantees.

The contents of the first four chapters were only a prerequisite allowing us to present our interdomain traffic analysis in Chapter 5. We studied in Chapter 5 the cost of using MPLS at the interdomain level. [UB00] also discusses the implications of using MPLS LSPs at the interdomain level. We have analyzed in details many aspects of the performance of using interdomain LSPs to carry best-effort traffic like the one we have today in the Internet. We have showed the many problems that would appear if MPLS were used with today's interdomain traffic. We have evaluated several LSP establishment techniques and discussed their performance. We also showed the problems of the signaling overhead and proposed several techniques aimed at reducing LSPs variability. We have then studied the performance of guaranteed bandwidth LSPs, i.e. LSPs for which we reserved a certain amount of bandwidth. Two different reservation techniques were evaluated, the first relying on a fixed amount of reservation while the other allowed the LSPs to renegotiate the amount bandwidth for every minute on the basis of their mean traffic. The analysis of the performance of both techniques uncovered the "self-similar" characteristics of the traffic. "Self-similarity" implies the inability to provide guarantees to interdomain flows. Having found the cause of this behavior has led us to propose several solutions: traffic shaping, regression techniques and modification of the TCP/IP stack. We explained why traffic shaping is the only realistic solution to the traffic variability problem.

It looks like if something had changed in the way we thought about traffic engineering and Internet traffic behavior! Resolving a problem requires an initial intuition of the constraints inherent to it. While we thought there were no significant issues at traffic engineering traffic streams, specifications and drafts alone do not change the way packets flood along network links. We now have some better intuition of how the real traffic behaves at the interdomain level. Maybe the reader is thinking our measurements were biased by some superior being so that it appears impossible to engineer anything in

such a mess. Many factors influence the way real traffic behaves, from psychological matters to routing paradigms. Beyond all this chaos[27], clear reasons explain such trends.

6.1 Reasons

The trends we did experience in the interdomain traffic without a doubt find their cause in one concept: best-effort. While TCP cares about congestion by decreasing its rate, available bandwidth generally means consumed bandwidth. No scheme in use today in the Internet prevents applications from greedily consuming available resources. The problem arises from the conflicting objectives of the user and the network manager: the former uses the most resources as it can while the latter would like to make its network cost minimal. One cannot satisfy both at the same time. A choice has to be made between service guarantees and operational cost.

If the traffic characteristics do not change, over-provisioning appears as the only way to provide guarantees to most traffic flows. A more realistic approach would probably provide guarantees to some part of the traffic by means of a service model like Diffserv or Intserv and best-effort for the remaining. Billing could also influence source behavior by smoothing it.

Our position may appear somewhat idealistic but QoS guarantees for all flows should be the one and only objective. We are conscious it cannot be a short-term goal but only like a direction to look at. Best-effort is no solution; it is an unfortunate situation while waiting for traffic engineering tools. The Internet today does not allow playing on routing and traffic distribution in order to significantly change the way resources are used.

6.2 Towards Interdomain QoS Guarantees?

Our intent has nothing to do with trying to change the characteristics of the traffic in the Internet. We only present in this section some pointers to potential solutions that may facilitate the deployment of traffic engineering in the Internet.

The most important aspect of traffic control concerns routing. Traffic engineering should not be limited to the intradomain level: all parts of the network must cooperate to ensure a coherent view of the QoS state is ubiquitous. Hierarchical QoS routing should be the solution for scalability and QoS guarantees. Each ISP uses the BGP view its peers were (generously) inclined to provide after having somewhat policed it. The view of the topology we have today consists more in partial reachability information than in a QoS map of the possible (aggregated) paths to attain every destination. Adding some QoS information to current interdomain and intradomain routing protocols does not suffice to allow ISPs to traffic engineer their incoming and outgoing traffic. It is not clear whether such information will be available someday. However, it should be possible for ISPs to influence routing to enhance the way traffic floods along their internal network and at access points.

We are talking about the ability to influence the paths followed by incoming and outgoing traffic to balance the load over every access link. Because incoming traffic is the most related with congestion, routing must enable ISPs to communicate their will in such a manner that congestion does not appear at the access points. Not only source routing does not allow this, but it could also worsen the situation. Source routing should be limited to gain or provide access to a limited part of the end-to-end path. Hierarchical routing with scalability in mind could be implemented through a distributed LSP establishment mechanism. The task of finding a satisfying path across the network must include every actor: the source network, the transit networks and the destination network. All of them have to agree

[27] In reference to the self-similar nature of the traffic.

on the path so that effective guarantees can be provided to a flow that requests a particular service. We have talked about requirements so far: implementation is a more complex task. There are many elements to achieve before getting these marvelous guarantees: prove their feasibility, show how the Internet could evolve so that incremental changes can lead to it, convince enough people about its use, ... and hope. A good method does not suffice by itself. The context makes the final judgment.

6.3 Future Work

Throughout this dissertation, we evolved from ignorance towards more realism. Of course, we did provide biased views of the problems and we missed some parts of reality. While we got some bad taste of what interdomain traffic hides, roads giving access to QoS and traffic engineering have been uncovered. Some bad news is no reason to discourage us from trying to make further steps towards QoS guarantees.

Several issues need some further work. First, it would be interesting to make the same traffic analysis with a mix of audio, video and other type of traffic. We saw how badly best-effort traffic behaves alone. IPTel and other multimedia applications have very different traffic patterns compared to the one we have today in the Internet. Maybe the addition of longer-lived flows can provide stability. Shaping best-effort traffic may also lessen the problem of traffic volume variability. Second, simulations need to be performed to test the ability of MPLS for traffic engineering and hierarchical routing purposes. These simulations should also implement the already defined service models in order to evaluate their usefulness in a real network. One cannot be sure that end-to-end guarantees can be provided in the Internet. Specifications alone do not ensure proper interoperability under real-life conditions. Third, new tools need to be implemented in order to enable ubiquitous traffic engineering deployment. Providing tools does not help in our network reality. You must in addition take care of interoperability and incremental deployment issues. Finally, there is a tremendous lack in the field of traffic modeling. Self-similarity has a cause. If traffic engineering were really to be used at the interdomain level, we should simulate the behavior of several types of traffic sources and aggregate many sources to evaluate the influence of the type of traffic source on interdomain traffic behavior. One cannot expect static application requirements. New applications with new bandwidth needs and new traffic generation patterns will appear. Traffic engineering should take into account the future needs at least to be able to adapt when they appear. This is why we find important to study the traffic pattern and its relations with source behavior. The TCP/IP stack probably will not be changed just because traffic engineering cannot be properly achieved in the Internet today. We need to know exactly what to expect from a change in applications behavior. Traffic modeling might bring the precious knowledge that would allow us to find problems in traffic behavior before they appear. It could even prevent the huge over-provisioning strategy on which almost everyone relies today.

Appendix 1

Glossary

Address Resolution Protocol (ARP): protocol used to resolve a destination address for a lower layer protocol (MAC) from a known address for another higher layer protocol (IP).

Application Programming Interface (API): set of functions used by an application program to access primitives from lower layer primitives (system or program libraries).

Asynchronous Transfer Mode (ATM): connection-oriented multiplexing and switching method utilizing fixed-length 53 bytes cells. It is asynchronous in the sense that cells carrying use data need not be periodic.

ATM Adaptation Layer (AAL): set of protocols and formats that define support for circuit emulation, packet video and audio, and connection-oriented and connectionless data services. These layers translate higher layer user traffic into a stream of ATM cells and convert the received cells back to the original form at the destination.

Autonomous System (AS): set of routers under a single technical administration, using an interior gateway protocol and common metrics to route packets within the AS, and using an exterior gateway protocol to route packets to other AS's (note that even if multiple IGPs and metrics are used inside the AS, the administration appears to other ASs as having one single coherent routing plan and presents a consistent picture of which networks are reachable through it).

Behavior Aggregate (BA): set of IP packets crossing a link and requiring the same Diffserv behavior

Bridged IP Subnet: IP subnet in which two or more physically disjoint media are made to appear as a single subnet.

Broadcast Subnet: supports an arbitrary number of hosts and routers (in theory) and capable of transmitting a single IP packet to all of these systems.

Cranckback: mechanism for partially releasing a connection establishment attempt, which encounters a failure back to the last node that made the routing decision.

Datagram: basic unit of transmission in a connectionless network service.

Data link layer: layer 2 of the OSI model.

DS3 link: 45 MBPS digital transmission interface (North American standard).

DSCP: specific value of the DSCP portion of the DS field, used to select a PHB.

E1 link: 2 MBPS digital transmission interface (European standard).

E3 link: 34 MBPS digital transmission standard (European standard).

Emulated LAN: logical network comprising both ATM and legacy attached LAN end stations.

End-to-end Path: two hosts that can communicate with one another over an arbitrary number of routers and subnets.

Early Packet Discard (EPD): packet discard (and congestion control) mechanism that drops all cells from an AAL5 PDU, preventing buffer overflow and loss of portions of multiple packets.

FANP: label binding protocol used by CSRs to notify that a flow has been selected for switching.

Forwarding Equivalence Class (FEC): set of packets traveling between a pair of hosts that can be handled equivalently for the purposes of forwarding and thus is suitable for binding to a single label.

File Transfer Protocol (FTP): capability in the TCP/IP protocol suite supporting transfer of files using TCP.

Flow: set of packets having a common characteristic for a specified period of time (generally rather short); This characteristic could be the same quintuple <source address, destination address, protocol, source port, destination port>, same pair <IP source address, IP destination address> or anything that could be considered as an association between two or more hosts.

Forwarding of a packet (or cell): consists in receiving it on an input port, examining some part of it and sending it on the appropriate output port based on the latter examination

Frame merge: label merging, when it is applied to operation over frame based media, so that the potential problem of cell interleave is not an issue.

Frame Relay: WAN networking standard for switching frames between end-users.

Host or end-system: delivers or receives IP packets to or from other systems but does not relay IP packets.

Internet Engineering Task Force (IETF): organization responsible for standards and specification development for TCP/IP networking.

Interface: boundary between two adjacent protocol layers or physical connection between devices.

Internetwork: concatenation of networks often with various different media and lower level encapsulations, to form an integrated larger network supporting communication between any of the hosts on any of the component network.

Interoperability: ability of heterogeneous devices and protocols to operate and communicate using a standard set of rules and protocols.

IP: connectionless datagram-oriented network layer protocol containing addressing and control information in the packet header that allows nodes to forward the packet toward the destination.

IP address resolution: quasi-static mapping between IP address on the local IP subnet and media address on the local subnet.

IP forwarding: process of receiving a packet and using a very low overhead decision process determining how to handle the packet (it is not said whether the header is changed or not by the decision process).

IP routing: exchange of information that takes place in order to have available the information necessary to make a correct IP forwarding decision.

IP subnet: collection of hosts that are able to transmit packets directly to other hosts of same subnet, IP distance between two hosts in the same subnet equals zero.

Label: short fixed length physically contiguous identifier, which is used to identify a FEC, usually of local significance.

Label merging: replacement of multiple incoming labels for a particular FEC with a single outgoing label.

Label stack: ordered set of MPLS labels enabling a labeled packet to traverse a hierarchy of LSPs.

Label swap: basic forwarding operation consisting of looking up an incoming label to determine the outgoing label, encapsulation, port, and other data handling information.

Label swapping: forwarding paradigm allowing streamlined forwarding of data by using labels to identify classes of data packets, which are treated indistinguishably when forwarding.

Label switching: generic term used to describe all approaches to forwarding IP (or other network layer) packets using a label swapping forwarding algorithm under the control of network layer routing algorithms; it uses exact match and rewrites the label on forwarding.

Label switched hop: hop between two MPLS nodes, on which forwarding is done using labels.

Label Switched Path (LSP): path through one or more LSRs at one level of the hierarchy followed by a packet in a particular FEC.

Label Switched Router (LSR): MPLS node that is capable of forwarding native layer 3 packets.

LAN Emulation (LANE): ATM Forum specification enabling ATM devices to seamlessly interwork with legacy LAN devices using either the Ethernet or Token Ring protocols.

Latency: time between initiating a request for a particular service and the beginning of the actual servicing.

Link: communication facility or medium over which (hosts) nodes can communicate at the link layer.

Local Area Network (LAN): MAC-level data and computer communications network confined to short geographic distances.

Logical IP Subnet (LIS): IP subnetwork that is physically spread over one or more physical ATM subnetworks.

Logical Link Control (LLC): sublayer that interfaces with the MAC sublayer of the data link layer in LAN standards.

Loop detection: method of dealing with loops in which loops are allowed to be set up, and data may be transmitted over the loop, but the loop is later detected.

Loop prevention: method of dealing with loops in which data is never transmitted over a loop.

Marking: process of discarding packets based on specified rules; policing.

Medium Access Control (MAC): protocol defined by the IEEE that controls workstation access to a shared transmission medium.

Merge point: node at which label merging is done.

Metering: process of measuring the temporal properties of a traffic stream selected by a classifier. The instantaneous state of this process may be used to affect the operation of a marker, shaper or dropper, and/or may be used for accounting and measurement purposes.

Metropolitan Area Network (MAN): network that operates over metropolitan area distances and number of subscribers.

Multicast capable subnet: supports a facility to send a packet to a subset of the destinations of the subnet.

MPLS domain: contiguous set of nodes that operate MPLS routing and forwarding and which are also in one routing or administrative domain.

MPLS edge node: MPLS node that connects an MPLS domain with a node that is outside of the domain, either because it does not run MPLS, and/or because it is in a different domain.

MPLS egress node: MPLS edge node in its role in handling traffic as it leaves an MPLS domain.

MPLS ingress node: MPLS edge node in its role in handling traffic as it enters an MPLS domain.

MPLS label: label that is carried in a packet header, and which represents the packet's FEC.

MPLS node: node that is running MPLS. An MPLS node will be aware of MPLS control protocols, will operate one or more layer 3 routing protocols, and will be capable of forwarding packets based on labels. An MPLS node may optionally be also capable of forwarding native layer 3 packets.

Multi Protocol Label Switching (MPLS): IETF working group and the effort associated with the working group.

Multi-Protocol Over ATM (MPOA): ATM Forum-defined means to route ATM traffic between virtual emulated LANs, bypassing traditional routers (uses NHRP).

Network: system of autonomous devices connected via physical media that provide a means for communications.

Next Hop Resolution Protocol: IETF-defined protocol for routers to learn network-layer addresses over NBMA networks like ATM.

Non-Broadcast Multiple Access (NBMA) subnet: subnet that supports an arbitrary number of hosts and routers but does not natively support a convenient multi-destination connectionless transmission facility.

OSI: architectural model developed by the International Standards Organization for the design of open systems networks.

Open Shortest Path First (OSPF): routing protocol defined for IP that uses Dijkstra algorithm to optimally determine the shortest path.

Partial Packet Discard (PPD): packet discard (and congestion control) mechanism used in ATM switches consisting in discarding all remaining cells of an AAL5 PDU except the last one after a device drops one or more cells of the same AAL5 PDU.

Per-Hop Behavior (PHB): externally observable forwarding behavior applied at a DS-compliant node to a DS behavior aggregate.

PHB group: set of PHBs that can only be meaningfully specified and implemented simultaneously, due to a common constraint applying to all PHBs in the set such as queue servicing or queue management policy. A PHB group provides a service building block that allows a set of related forwarding behaviors to be specified together.

Policing: process of discarding packets within a traffic stream in accordance with the state of a corresponding meter enforcing a traffic profile.

Port: physical interface.

Protocol: formal set of conventions and rules governing the formatting and sequencing of message exchange between two communicating systems.

QoS: ability to allocate network resources (bandwidth, buffer allocation,...) for communication between two stations.

Router (or gateway or intermediate system): host property and relays IP packets.

Real Time Protocol (RTP): protocol of the TCP/IP suite that conveys sequence numbers and time-stamps between packet video and audio applications.

Resource reSerVation Protocol (RSVP): protocol developed by the IETF to support different classes of service for IP flows.

Service Access Point (SAP): point where services of a lower layer are available to the next higher layer.

Service provisioning policy: policy which defines how traffic conditioners are configured on DS boundary nodes and how traffic streams are mapped to DS behavior aggregates to achieve a range of services.

Service Level Agreement (SLA): service contract between a customer and a service provider that specifies the forwarding service a customer should receive. A customer may be a user organization (source domain) or another DS domain (upstream domain). A SLA may include traffic conditioning rules which constitute a TCA in whole or in part.

Shaping: process of delaying packets within a traffic stream to cause it to conform to some defined traffic profile.

Source routing: scheme in which the originator determines the end-to-end route.

Throughput: total useful information processed or communicated during a specified period.

Traffic conditioner: entity which performs traffic conditioning functions and which may contain meters, markers, droppers and shapers. Traffic conditioners are typically deployed in DS boundary

nodes only. A traffic conditioner may re-mark a traffic stream or may discard or shape packets to alter the temporal characteristics of the stream and bring it into compliance with a traffic profile.

Traffic contract: agreement between the user and the network regarding the expected QoS provided by the network subject to user compliance with the predetermined traffic parameters.

Traffic Conditioning Agreement (TCA): agreement specifying classifier rules and any corresponding traffic profiles and metering, marking, discarding, and/or shaping rules which are to apply to the traffic streams selected by the classifier.

Traffic profile: description of the temporal properties of a traffic stream such as rate and burst size.

Transmission Control Protocol (TCP): layer 4 transport protocol that reliably delivers packets to higher layer protocols. It performs sequencing and reliable delivery via retransmission.

User Datagram Protocol (UDP): connectionless datagram-oriented transport-layer protocol belonging to the TCP/IP suite.

VCID: identifier that uniquely identifies the datalink connection between two adjacent CSRs (contains the ESI of the source node and an unique identifier within a source node).

VC merge: label merging where the MPLS label is carried in the ATM VCI field (or combined VPI/VCI field), to allow multiple VCs to merge into one single VC.

Virtual Circuit (VC): circuit used by a connection-oriented layer 2 technology such as ATM or Frame Relay, requiring the maintenance of state information in layer 2 switches.

VP merge: label merging where the MPLS label is carried in the ATM VPI field, to allow multiple VPs to be merged into one single VP. In this case, two cells would have the same VCI value only if they originated from the same node. This allows cells from different sources to be distinguished via the VCI.

Wide Area Network (WAN): network that operates over a large geographic region.

Appendix 2

Acronyms

AAL5 = ATM Adaptation Layer 5

AF = Assured Forwarding

API = Application Programming Interface

ARIS = Aggregate Route-Based IP Switching

ARP = Address Resolution Protocol

ARPA = Advanced Research Projects Agency

ARPANET = Advanced Research Projects Agency Network

ATM = Asynchronous Transfer Mode

BA = Behavior Aggregate

BGP = Border Gateway Protocol

CE = Customer Edge

CR-LDP = Constrained-based routing Label Distribution Protocol

CRLSP = Constrained-based routing Label Switched Path

CSR = Cell Switching Router

CIDR = Classless InterDomain Routing

DoD = Department of Defense

DS = Differentiated Services

DSCP = Differentiated Services codepoint

DNHR = Dynamic Non-Hierarchical Routing

EBGP = Exterior BGP

EF = Expedited Forwarding

EGP = Exterior Gateway Protocol

EPD = Early Packet Discard

ESI = End-System Identifier

FANP = Flow Attribute Notification Protocol

FEC = Forwarding Equivalence Class

FF = Fixed-Filter

FIFO = First-In First-Out

FTP = File Transfer Protocol

IBGP = Interior BGP

IDRP = Inter-Domain Routing Protocol

IETF = Internet Engineering Task Force

IGP = Interior Gateway Protocol

IP = Internet Protocol

IS = Integrated Services

IS-IS = Intermediate System to Intermediate System

ISP = Internet Service Provider

ISR = IP Switch Router

ITU = International Telecommunications Union

LAN = Local Area Network

LANE = LAN Emulation

LDP = Label Distribution Protocol

LEC = LAN Emulation Client

LES = LAN Emulation Server

LIS = Logical IP subnet

LLC = Logical Link Control

LSR = Label Switching Router

LSP = Label Switched Path

MAC = Medium Access Control

MBPS = Mega Bits Per Second

MPOA = Multi Protocol Over ATM

MPLS = Multi Protocol Label Switching

NARP = NBMA Address Resolution Protocol

NBMA = Non-Broadcast Multiple Access network

NHLFE = Next Hop Label Forwarding Entry

NHRP = Next Hop Resolution Protocol

OSI = Open System Interconnection

OSPF = Open Shortest Path First

PE = Provider Edge

PHB = Per-Hop Behavior

PIM = Protocol Independent Multicast

PPD = Partial Packet Discard

QoS = Quality of Service

RSVP = Resource reSerVation Protocol

RTP = Real Time Protocol

SE = Shared-Explicit

SLA = Service Level Agreement

SMDS = Switched Multimegabit Data Service

SNAP = Sub-Network Access Point

SP = Service Provider

TCA = Traffic Conditioning Agreement

TCP = Transmission Control Protocol

TDP = Tag Distribution Protocol

TFIB = Tag Forwarding Information Base

ToS = Type of Service

TSR = Tag Switch Router

VC = Virtual Circuit

VCI = Virtual Circuit Identifier

VCID = Virtual Connection IDentifier

VPI = Virtual Path Identifier

WF = Wildcard-Filter

WFQ = Weighted Fair Queuing

Appendix 3

Bibliography

[AF99] J. Heinanen, F. Baker, W. Weiss and J. Wroclawski. *Assured Forwarding PHB Group.* Internet RFC 2597, June 1999.

[BD96] P. Brockwell and R. Davis. *Introduction to Time Series and Forecasting.* Springer-Verlag, New York, 1996.

[Ber94] J. Beran. *Statistics for Long-Memory Processes.* Monographs on Statistics and Applied Probability, Chapman & Hall, 1994.

[BGP495] Y. Rekhter and T. Li. *A Border Gateway Protocol 4 (BGP-4).* Internet RFC 1771, March 1995.

[BGPRR] T. Bates, R. Chandra and E. Chen. *BGP Route Reflection: An alternative to full mesh IBGP.* Internet-Draft, work in progress.

[Bra89] R. Braden. *Requirements for Internet hosts – communication layers.* Internet RFC 1122, October 1989.

[CAIDA] CAIDA. *Cflowd.* Available from http://www.caida.org/Tools/Cflowd/, 1998.

[Che99] S. Chen. *Routing Support for Providing Guaranteed End-to-End Quality-of-Service.* Ph.D. Thesis, University of Illinois at Urbana-Champaign, 1999.

[Cis99] Cisco. *NetFlow services and applications.* White paper, available from http://www.cisco.com/warp/public/732/netflow, 1999.

[CL97] J. Wroclawski. *Specification of the Controlled-Load Network Element Service.* Internet RFC 2211, September 1997.

[CSR99] G. Chiruvolu, R. Sankar and N. Ranganathan. *VBR video traffic management using a predictor-based architecture.* ACM Computer Communications Review, vol. 23, 2000, pp. 62-70.

[DDR98] B. Davie, P. Doolan, and Y. Rekhter. *Switching in IP Networks, IP Switching, Tag Switching, and Related Technologies.* San Francisco : Morgan Kaufmann, 1998.

[DKS89] A. Demers, S. Keshav and S. Shenker. *Analysis and Simulation of a Fair Queuing Algorithm.* In Proceeding of ACM Sigcomm '89, pp. 3-12.

[DS98] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss. *An Architecture for Differentiated Services.* Internet RFC 2475, December 1998.

[DS99] D. Mc Dysan and Darren Spohn. *ATM Theory and Applications.* Signature Edition, McGraw-Hill, 1999.

[DSR99] Y. Bernet, A. Smith and S. Blake. *A Conceptual Model for Diffserv Routers.* Internet-Draft, work in progress.

[DGG+99] N. Duffield, P. Goyal, A. Greenberg, P. Mishkra, K. Ramakrishnan and J. van der Merwe. *A Flexible Model for Resource Management in Virtual Private Networks.* In Proceedings of ACM Sigcomm '99, pp. 95-108.

[EF99] V. Jacobson, K. Nichols and K. Poduri. *An Expedited Forwarding PHB.* Internet RFC 2598, June 1999.

[FLY+93] V. Fuller, T. Li, J. Yu and K. Varadhan. *Classless Inter-Domain Routing (CIDR) : an Address Assignment and Aggregation Strategy.* Internet RFC 1519, September 1993.

[FRC98] A. Fedmann, J. Rexford and R. Caceres. *Efficient Policies for Carrying Web Traffic Over Flow-Switched Networks.* IEEE/ACM Transactions on Networking, Vol. 6, No 6, pp. 673-685, December 1998.

[FV97] N. Feldman and A. Viswanathan. *ARIS Specification.* Internet-Draft, work in progress.

[GGM+97] R. Garroppo, S. Giordano, S. Miduri, M. Pagano and F. Russo. *Statistical Multiplexing of Self-Similar VBR videoconferencing traffic.* In proceedings of IEEE GLOBECOM '97, Phoenix, November 1997.

[GR96] R. Govindan and A. Reddy. *An Analysis of Internet Inter-Domain Topology and Route Stability.* Technical Report, University of Southern California, Information Sciences Institute, 1996.

[GS97] S. Shenker, C. Partridge and R. Guerin. *Specification of Guaranteed Quality of Service.* Internet RFC 2212, September 1997.

[Hei93] J. Heinanen. *Multiprotocol Encapsulation over ATM Adaptation Layer 5.* Internet RFC 1483, July 1993.

[HSS87] B. Hurley, C. Seidl and W. Sewel. *A survey of Dynamic Routing Methods for Circuit-Switched Traffic.* IEEE Communication Magazine, September 1987.

[Hum] H. Hummel. *Orchestrally Conducted Traffic (OCT).* Internet-Draft, work in progress.

[IETF] IETF web site, available from http://www.ietf.org.

[IPV6] S. Deering and R. Hinden. *Internet Protocol, Version 6 (IPv6) Specification.* Internet RFC 2460, December 1998.

[IRA97] B. Halabi. *Internet Routing Architectures.* Cisco Press, 1997.

[IS94] R. Braden, D. Clark and S. Shenker. *Integrated Services in the Internet : an Overview.* Internet RFC 1633, June 1994.

[ISDS] Y. Bernet, R. Yavatkar, P. Ford, F. Baker, L.Zhang, M. Speer, R. Braden, B. Davie, J. Wroclawski and E. Felstaine. *A Framework For Integrated Services Operation Over Diffserv Networks.* Internet-Draft, work in progress.

[ISISTE] H. Smith and T. Li. *IS-IS extensions for Traffic Engineering.* Internet-Draft, work in progress.

[Kil99] K. Kilkki. *Differentiated Services For The Internet.* Macmillan Technical Publishing, 1999.

[Kle61] L. Kleinrock. *Information Flow in Large Communication Nets.* RLE Quarterly Progress Report, July 1961.

[KNE97] Y. Katsube, K. Nagami and H. Esaki. *Toshiba's Router Architecture Extensions for ATM : Overview.* Internet RFC 2098, February 1997.

[KS99] E. Knightly and N. Shroff. *Admission Control for Statistical QoS : Theory and Practice.* IEEE Network, March/April 1999.

[Kus99] T. Kushida. *An empirical study of the characteristics of Internet traffic.* ACM Computer Communications, vol. 22, 1999, pp. 1607-1618.

[LANE95] ATM Forum. *LAN Emulation over ATM 1.0.* af-lane-0021.000, January 1995.

[Lau94] M. Laubach. *Classical IP and ARP over ATM.* Internet RFC 1577, January 1994.

[LKP+98] J. Luciani, D. Katz, D. Piscitello, B. Cole and N. Doraswamy. *NBMA Next Hop Resolution Protocol (NHRP).* Internet RFC 2332, July 1998.

[LTW+94] W. Leland, M. Taqqu, W. Willinger and D. Wison. *On the Self-Similar Nature of Ethernet Traffic (Extended Version).* IEEE/ACM Transactions on Networking, February 1994.

[Ma98] Q. Ma. *Quality-of-Service Routing in Integrated Services Networks.* Ph. D. Thesis, School of Computer Science, Carnegie Melon University, January 1998.

[MIDS] http://www.mids.org/growth/internet/.

[MPLSAR] E. Rosen, A. Viswanathan and R. Callon. *Multiprotocol Label Switching Architecture.* Internet-Draft, work in progress.

[MPLSCH] MPLS charter, available from http://www.ietf.org/html.charters/mpls-charter.html.

[MPLSCRLDP] B. Jamoussi. *Constraint-Based LSP Setup using LDP.* Internet-Draft, work in progress.

[MPLSCRAS] J. Ash, M. Girish, E. Gray, B. Jamoussi and G. Wright. *Applicability Statement for CR-LDP.* Internet-Draft, work in progress.

[MPLSFR] R. Callon, N. Feldman, A. Fredette, G. Swallow and A. Viswanathan. *A Framework for Multiprotocol Label Switching.* Internet-Draft, work in progress.

[MPLSLDP] L. Andersson, P. Doolan, N. Feldman, A. Fredette and B. Thomas. *LDP Specification.* Internet-Draft, work in progress.

[MPLSSHIM] E. Rosen, Y. Rekhter, D. Tappan, D. Farinacci, G Fedorkow, T. Li and A. Conta. *MPLS Label Stack Encoding.* Internet-Draft, work in progress.

[MPLSRSVP] B. Davie, Y. Rekhter, E. Rosen, A. Viswanathan, V. Srinivasan and S. Blake. *Use of Label Switching with RSVP.* Internet-Draft, work in progress.

[MPLSRSVPE] B. Davie, Y. Rekhter, E. Rosen, A. Viswanathan and V. Srinivasan. *Extensions to RSVP for LSP Tunnels.* Internet-Draft, work in progress.

[MPLSTE] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell and J. McManus. *Requirements for Traffic Engineering Over MPLS.* Internet RFC 2702, September 1999.

[MPLSTEQoS] P. Bhaniramka, W. Sun and R. Jain. *Quality of Service using Traffic Engineering over MPLS: An Analysis.* Internet-Draft, work in progress.

[MPOA97] ATM Forum. *MultiProtocol over ATM (MPOA), Version 1.0.* af-mpoa-0087.000, July 1997.

[NEH+96[1]] P. Newman, W. Edwards, R. Hinden, E. Hoffman, F. Ching Liaw, T. Lyon and G. Minshall. *Ipsilon Flow Management Protocol Specification for IPv4 Version 1.0.* Internet RFC 1953, May 1996.

[NEH+96[2]] P. Newman, W. Edwards, R. Hinden, E. Hoffman, F. Ching Liaw, T. Lyon and G. Minshall. *Ipsilon's General Switch Management Protocol Specification Version 1.1.* Internet RFC 1987, August 1996.

[NEK+99] K. Nagami, H. Esaki, Y. Katsube and O. Nakamura. *Flow Aggregated, Traffic Driven Label Mapping in Label-Switching Networks.* IEEE journal on selected areas in Communications, Vol. 17, NO. 6, June 1999.

[Nimrod] I. Castineyra, N. Chiappa and M. Steenstrup. *The Nimrod routing Architecture.* Internet RFC 1992, August 1996.

[NKS+97] K. Nagami, Y. Katsube, Y. Shobatake, A. Mogi, S. Matsuzawa, T. Jinmei and H. Esaki. *Toshiba's Flow Attribute Notification Protocol (FANP) Specification.* Internet RFC 2129, April 1997.

[NLM96] P. Newman, T. Lyon and G. Minshall. *Flow Labeled IP : A Connectionless Approach to ATM.* In Proceedings of the IEEE Infocom, March 1996.

[NML98] P. Newman, G. Minshall and T. Lyon. *Ip Switching - ATM under IP.* IEEE/ACM Transactions on Networking, 6(2):117-129, April 1998.

[OSPFTE] D. Yeung. *OSPF Extensions for Traffic Engineering.* Internet-Draft, work in progress.

[Par92] A. Parekh. *A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks.* Technical Report LIDS-TR-2089, Laboratory for Information and Decision Systems, MIT, 1992.

[PASTE] T. Li and Y. Rekhter. *A Provider Architecture for Differentiated Services and Traffic Engineering (PASTE).* Internet-Draft, work in progress.

[PMH+95] M. Perez, F.A. Mankin, E. Hoffman, G. Grossman, and A. Malis. *ATM signaling support for IP over ATM.* Internet RFC 1755, February 1995.

[PERL97] R. Schwartz and T. Christiansen. *Learning Perl.* O'Reilly & Associates, 1997.

[PERLA97] S. Srinivasan. *Advanced Perl Programming.* O'Reilly & Associates, 1997.

[PERL99] J. Orwant, J. Hietaniemi and J. Macdonald. *Mastering Algorithms with Perl.* O'Reilly & Associates, August 1999.

[PF95] V. Paxson and S. Floyd. *Wide-Area Traffic : The Failure of Poisson Modeling.* IEEE/ACM Transactions on Networking, June 1995.

[QoSOSPF] G. Apostopoulos, R. Guerin, S. Kamat, A. Orda, T. Przygienda and D. Williams. *QoS Routing Mechanisms and OSPF Extensions.* Internet-Draft, work in progress.

[RDK+97] Y. Rekhter, B. Davie, D. Katz, E. Rosen and G. Swallow. *Cisco Systems' Tag Switching Architecture overview.* Internet RFC 2105, February 1997.

[RG95] Y. Rekhter and P. Gross. *Application of the Border Gateway Protocol in the Internet.* Internet RFC 1772, March 1995.

[RSVP97] R. Braden, L. Zhang, S. Berson, S. Herzog and S. Jamin. *Resource ReSerVation Protocol (RSVP) Version 1 Functional Specification.* Internet RFC 2205, September 1997.

[Sta98] W. Stallings. *High-Speed Networks : TCP/IP and ATM Design Principles.* Prentice-Hall, 1998.

[Ste94] W. Stevens. *TCP/IP illustrated : the protocols.* Addison-Wesley, 1994

[Tan88] A. Tannenbaum. *Computer Networks.* 3rd Edition, Prentice-Hall, 1988.

[TE00] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja and X. Xiao. *A Framework for Internet Traffic Engineering.* Internet-Draft, work in progress.

[Telstra] http://www.telstra.net/ops/bgptable.html.

[ToS92] P. Almquist. *Type of Service in the Internet Protocol Suite.* Internet RFC 1349, July 1992.

[TMW97] K. Thompson, G. Miller and R. Wilder. *Wide-Area Traffic Patterns and Characteristics.* IEEE Network, November/December 1997.

[UB00] S. Uhlig and O. Bonaventure. *On the cost of using MPLS for interdomain traffic.* To appear in the proceedings of the Qofis2000 conference, Berlin, September 2000.

[VPNBGP] E. Rosen and Y. Rekhter. *BGP/MPLS VPNs.* Internet RFC 2547, March 1999.

[VPNIP] B. Gleeson, J. Heinanen, G. Armitage and A. Malis. *A Framework for IP Based Virtual Private Networks.* Internet-Draft, work in progress.

[VPNMPLS] K. Muthukrishnan and A. Malis. *Core MPLS IP VPN Architecture.* Internet-Draft, work in progress.

[VPNRT] H. Berkowitz. *Requirements Taxonomy for Virtual Private Networks.* Internet-Draft, work in progress.

[WE94] L. Wei and D. Estrin. *The Trade-offs of Multicast Trees and Algorithms.* In the Proceedings of the 1994 Conference of Computer Communications and Networks, 1994.

[WE99] I. Widjaja and I. Elwalid. *Performance Issues in VC-Merge Capable Switches for Multiprotocol Label Switching.* IEEE journal on selected areas in Communications, Vol. 17, NO. 6, June 1999.

[WP98] W. Willinger and V. Paxson. *Where Mathematics meets the Internet.* Notices of the American Mathematical Society, 45(8), pp. 961-970, September 1998.

[ZE97] D. Zappala and D. Estrin. *Alternate Path Routing and Pinning for Interdomain Multicast Routing.* Technical Report, University of Southern California, Computer Science, 1997.

Appendix 4

Label Switching Implementations

A4.1 Toshiba's "CSR/FANT"

The first label switching implementation publicly announced was the "Cell Switching Router" (or CSR) from Toshiba (see [KNE97]). This is one implementation of running IP control protocol over ATM switches. Flow Attribute Notification Protocol (FANP) is the label binding protocol used by the CSR approach. Everything began with one significant drawback of the "Classical IP and ARP over ATM" model [Lau94]: the obligation for two hosts within the same ATM network but on different LISs to communicate through a router even if a direct VC could have been established between them. At the time of the development of the "Classical IP over ATM" model, routers' performance was far behind those of ATM switches in terms of throughput, latency and QoS. The need for a router implied that inter-LIS traffic (the cell stream) had to be reassembled by every router on the path. The router made its forwarding decision on the reassembled packets and then segmented the cell stream for the next router or the destination host. The overhead generated by this traffic was a great motivation for the CSR proposal. The goal of the CSR solution was to allow ATM switches to switch IP traffic without being obliged to assemble and disassemble inter-LIS traffic.

A CSR can be viewed like an emulated IP router on top of a classical ATM switch. From a control point of view, a CSR looks like a router since it has been designed to interconnect LISs. From a forwarding point of view, it is like an ordinary ATM switch. The CSR specification defines three types of VC's: "default", "dedicated" and "cut-through". All CSR's VCs are unidirectional except the default ones that are bi-directional. The default VC is used in the initial state, for communication between a pair of CSRs: all traffic between them flows over this VC (routing protocols, data traffic...). A dedicated VC carries an individual flow identified by a *<source address,destination address, protocol,source port,destination port>* quintuple. A cut-through VC is the result of the merging of two dedicated VCs at a CSR. In the initial state, all traffic between a pair of adjacent CSRs passes on the default VC. Adjacent means that there may be any number of classical ATM switches between them without any side effect; the only constraint is that there must exist a direct VC between these two CSRs. At every CSR, incoming traffic from the default VC is monitored and flows are selected in some way from it with the intention of providing cut-through VCs for them. The cut-through trigger in CSRs is based on the examination of TCP and UDP port numbers that identify the application that is sending traffic. The traffic received by a CSR on the default VC is reassembled and forwarded to the next hop CSR the way classical routers do it ("normal" IP routing). If a packet belongs to a flow that is selected for a cut-through VC, a dedicated VC is selected by the CSR for that flow. The association between flows and VCs are communicated to the next hop CSR through FANP. When a CSR receives a particular flow on a VC and transmit the same flow on another VC, it can set up a cut-through VC to forward cells from the flow at the ATM level (true switching this time, no assembly and disassembly between cells and packets).

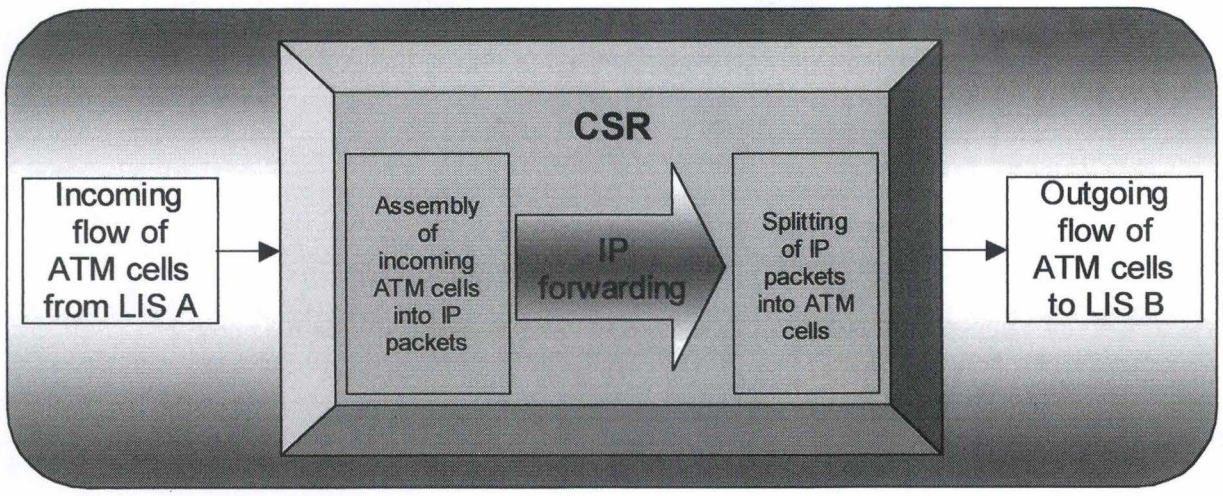

Figure A4.1: CSR operating as router

The Flow Attribute Notification Protocol (see [NKS+97]) is the label distribution protocol that runs on every CSR (between adjacent CSRs). Its messages are directly transported in IP (except for the OFFER message that is sent at the link layer[28]). FANP supposes that the native ATM layer labels (VPI/VCI) cannot be used as CSR labels because adjacent CSRs can be interconnected by any number of classical ATM switches (the intermediate ATM switches rewrite the VPI/VCI fields at each hop). FANP operates in two phases: the VCID negotiation phase and the binding between a VCID and a FLOWID phase. In the first phase, a PROPOSE message containing a VCID is sent on a particular VC. This message means for the receiver CSR that it has to associate the VCID of the message with the particular VC (the VCID is the new logical identifier for both CSRs for the VC on which it was sent). The PROPOSE message is then acknowledged by a PROPOSE ACK message sent on the default VC.

In the second phase, an OFFER message containing a pair *<VCID,FLOWID>* indicates the flow that will be sent through this VC (the FLOWID identifies an IP flow). The only flow defined by now is of the form *<IP source address,IP source destination>*. That means that once a dedicated VC established between a pair of IP hosts, all traffic between them will transit through this VC.

This is not a very serious drawback since new FLOWID types could be defined that would give different granularity levels to the flows that transit through a dedicated VC. If we look at the nature of the differences between the two phases, we note that the first one is "hard state" which means that the VCID is permanently installed and is not refreshed. On the other hand, the <VCID, FLOWID> association is established in "soft state" which means that binding between VCID and FLOWID must be refreshed periodically. Once a flow becomes inactive, the association is broken and a new OFFER message must be sent to allow traffic to flow on the VC. Note that only the PROPOSE message is sent on a dedicated VC, all other messages are sent on the default VC which allows to transport only data traffic on dedicated and cut-through VCs (except the PROPOSE message).

The READY message must be sent at least once per refresh interval (defined in OFFER message). Otherwise, the association between VCID and FLOWID is broken. Note that the association VCID/FLOWID can be explicitly broken by the REMOVE message (and REMOVE ACK).

[28]FANP was intended to work on any connection-oriented data link layer but the specification and current implementations only allow ATM by now.

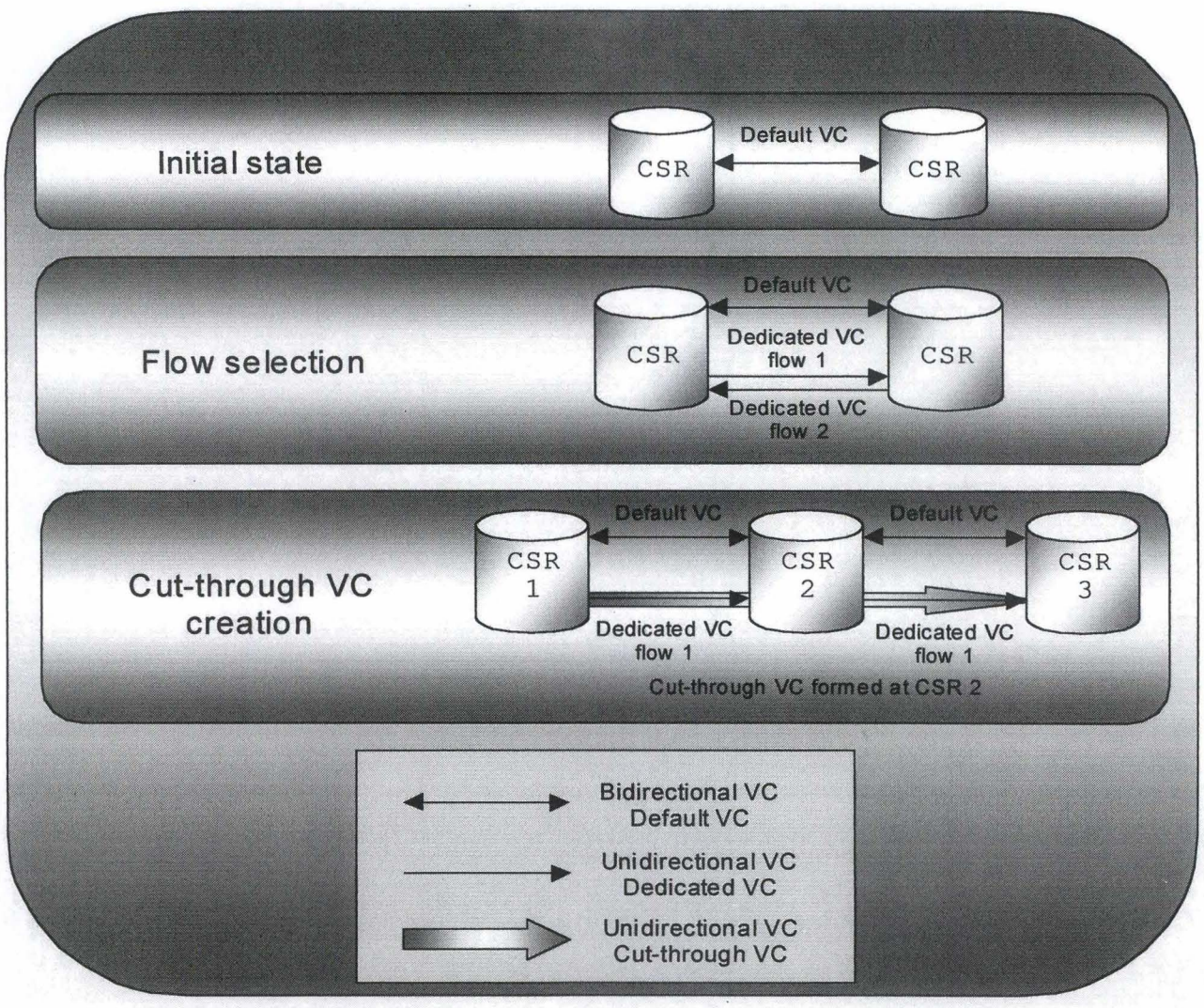

Figure A4.2: VC types in a CSR

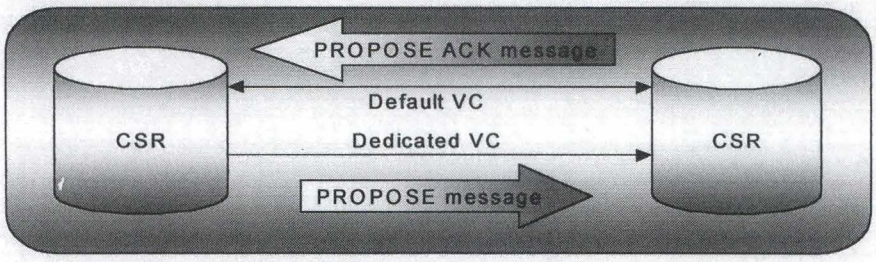

Figure A4.3: Phase 1, VCID negotiation

An advantage of the CSR/FANP approach is that it was developed to work transparently in IP networks. It is also the sole approach that enables label switching devices to communicate over classical ATM VCs. According to the concepts defined in section 2.5.6, CSR/FANP is a data-driven, upstream and local binding approach since transit traffic determines the creation of dedicated VCs (TCP and UDP ports), VCID and FLOWID values are defined by the upstream CSRs and a CSR does not need to know remote binding information to advertise local binding (since VCID/FLOWID is a local and independent choice). Finally, forwarding entries creation is independent because a CSR does not need to know remote binding information to create the forwarding entries.

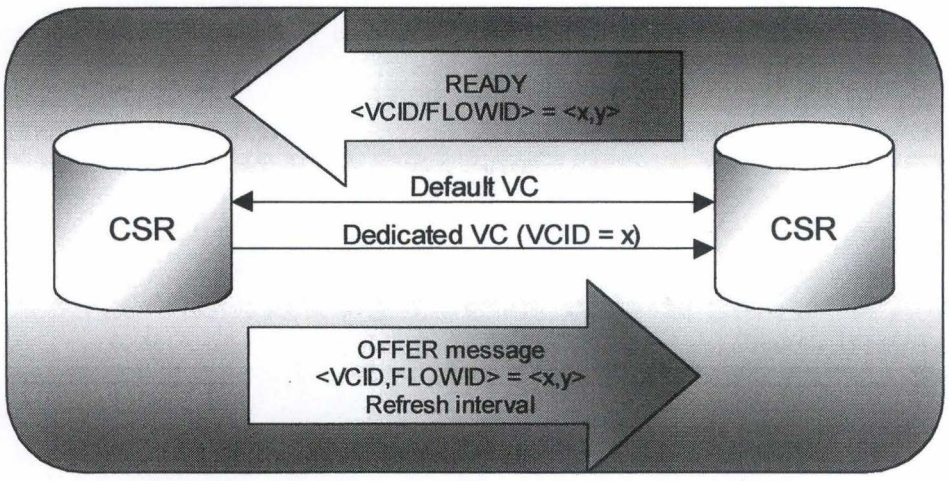

Figure A4.4: Phase 2, VC/flow binding

A4.2 Ipsilon's "IP Switching"

Ipsilon approach to label switching (see [NLM96]) was invented soon after CSR and the first that delivered real products on the marked. It has also been the catalyst for Cisco's Tag Switching approach and the MPLS working group in the IETF. This approach is conceptually quite similar to the CSR one in the sense that it uses data-driven label binding and the granularity of label creation does not differ very much. The principal difference with the previous approach is the existence of a switch management protocol, known as GSMP (General Switch Management Protocol) that allows an ATM switch to be controlled by an "IP switch controller". At an architectural point of view, IP Switching is somewhat different from the other approaches since it has separated the control and forwarding components. IFMP (Ipsilon Flow Management Protocol) cares about label binding between IP Switches while GSMP is a master/slave protocol with the ATM switch as the slave and the master running on an IP Switch controller (general purpose computing engine with an ATM interface to communicate with the switch). The GSMP part of IP Switching is a good solution since it is less risky to buy an ATM switch and control it by software than buying a completely new product (new switch and new software): this philosophy permitted to gain the attention of the market. The device that implements IP Switching is called an IP Switch.

A remarkable characteristic of all the approaches presented here is the complete removal of ATM control plane needed in classical ATM networks. Remark that some approaches allow the coexistence of IP switching and classical ATM on the same LSRs ("ships in the night" operation mode). This choice takes origin into the complexity due to the interaction between the simple IP and the complex ATM signaling protocols that waste networking protocols implementers' life (and time). IP Switching integrates IP routing and ATM switches in a quite simple way as illustrated in figure A4.5. The benefit from the removal of the ATM control plane removal is that IP Switches can communicate directly with other peer IP Switches.

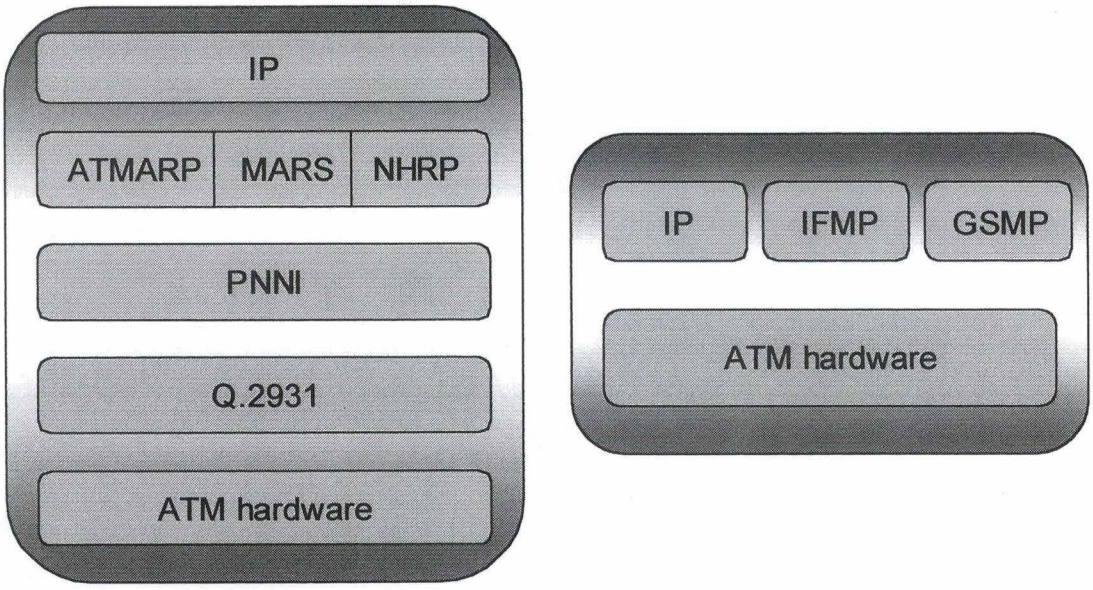

Figure A4.5: Standard "IP over ATM" vs. IP Switching models

Figure A4.5 will be our guide during the explanation of the functioning of IP Switching. The default VC is used to get control traffic (routing protocols and IFMP messages) from adjacent IP Switches through a well-known VPI/VCI VC without being obliged to resort to signaling for a VC. The default VC does not rely on any ATM signaling procedure since it only connects the two adjacent IP Switches controllers through their attached ATM switches. Traffic that flows through the default VC is encapsulated according to [Hei93] using LLC/SNAP, sent to the switch controller and reassembled. Data that does not have label associated with it is forwarded on this VC by the switch controller in software.

The role of IFMP (see [NEH+96[1]]) is to allow flow to label binding information to be communicated between two IP Switches (connected by a point to point link). It uses downstream label binding in theory but in practice, the upstream node is free to use or not label's values sent. The matter of when and how to allocate labels is a local one as long as this decision is taken in a coherent manner between adjacent IP Switches within a particular domain (coherence is defined with reference to the IFMP operations later defined). IFMP is a soft-state protocol (like flow/VCID binding in CSR approach) that needs periodic refreshes. Flow binding existence is thus limited in time and must be refreshed by the downstream IP Switch. IFMP messages also contain synchronization information (a sequence number) in case of loss of some IFMP message. Since IFMP uses best-effort message delivery over IP, a synchronization method has been built within IFMP: if an IFMP message gets lost (not by itself of course), flow state synchronization will be lost until the message is retransmitted. IFMP is made up of an adjacency protocol and the redirect protocol. IFMP's adjacency protocol, as its name implies, enables cooperating switches to exchange an initial set of information so that they acquire enough shared state to begin label advertisement. The ADJACENCY message enables the switches at the end of the link to learn each other's identity[29]. There is no need to adjacent IP Switches to explicitly communicate their respective address since it can be deduced from the IP encapsulation of the ADJACENCY message (source address). Adjacency protocol messages must be resent periodically due to the soft-state nature of IFMP. Synchronization and link number exchange of information between two IP Switches appears in the adjacency protocol for message loss handling and link identification. If synchronization is lost due to some link error, the protocol enables the link to be reset.

[29] The IP limited broadcast address 255.255.255.255 is listened to by all hosts on a network so that the IP Switches at both ends of the link can send messages to each other before they know the other's IP address.

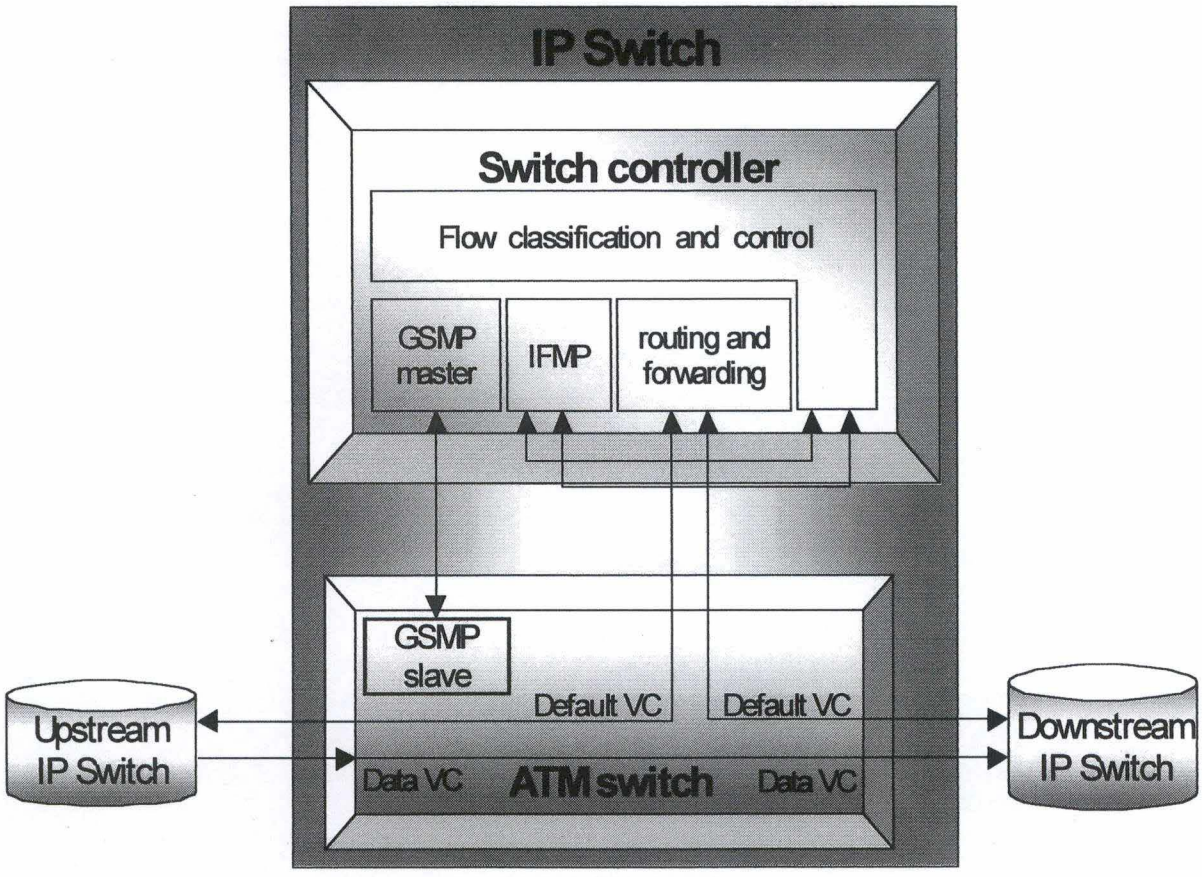

Figure A4.6: Simplified IP Switch architecture

IFMP's redirection protocol uses five message types. All of them are encapsulated in IP datagrams that are sent to the unicast address of the peer system. Recall that the unicast address is learned via the adjacency protocol.

The Op code field serves to indicate the message type contained in the message body. More than one message can be present in the message body but all of them must be of the same type (same Op code value). The previously mentioned instance and sequence number fields provide link identification and synchronization information. The message body contains one of the five messages of the protocol: REDIRECT, RECLAIM, RECLAIM ACK, LABEL RANGE and ERROR.

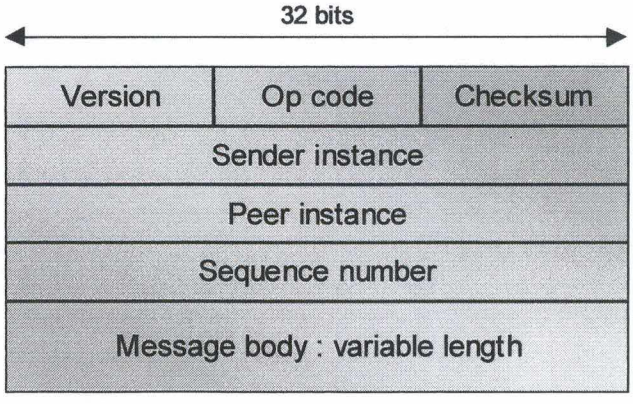

Figure A4.7: IFMP REDIRECT protocol message format

The first one, the REDIRECT message, is the most important since it is used to bind a label to a flow. We will only give an overview of the other messages: RECLAIM enables a label's value to be unbound for further use, RECLAIM ACK is an acknowledgment for a RECLAIM message, LABEL RANGE enables the acceptable range of label's value to be communicated for a switch to its neighbors and finally ERROR message deals with error conditions. We shall go a little deeper within the REDIRECT message format because it has interesting implications on the flow concept. There are two types of flow in the IFMP protocol: type 1 is of the form *<source address,source port,destination address,destination port>* while type 2 is of the form *<source address,destination address>*. The previous definitions of the flow concept determine the label binding made by IFMP. Nevertheless, the IP Switching architecture does not restricts flow definition to the two previously defined. The real IFMP flow definition is somewhat more restrictive since a flow is a set of packets that have the same value in all the IP header fields (except TOS and Protocol ID for type 2 flows that could eventually change between two communicating IP hosts). The flow identifier present in the message body corresponds thus to one of the two formats defined in figure A4.8. Note that all IPv4 header fields can be found in the REDIRECT message body for reasons that will be explained later. Once the REDIRECT message is exchanged between adjacent IP Switches, the complete body message is no longer useful (since the only information it provides is that the message encapsulated comes from the IP protocol) and encapsulation on a redirected VC is used. When a flow is moved from the default VC to a redirected one, the LLC/SNAP header is removed while the IP header (TCP and UDP headers would also be affected by type 1 flows) is transformed into an IFMP flow type header. Let us see how it is done. First, an IP Switch selects a particular flow for redirection by inspecting the IP header of its packets. Then, the IP Switch sends a REDIRECT message containing a flow identifier and a VPI/VCI bound to it to the upstream IP Switch. The upstream IP Switch sends the flow on the VC specified in the REDIRECT message using the appropriate encapsulation (depending on the flow type).

Now, given the fact that a switch keeps a copy of every REDIRECT message it has sent, it can reconstruct the IP header of the flow packets since it receives the flow identifier through the VPI/VCI VC where flow encapsulation is used.

The IP Switch receiver of the redirected flow can then reconstruct the IP message header. A situation in which the redirected VC traverses several IP Switch is similar. The egress IP Switch will be able to reconstruct the IP header since it has the REDIRECT message information (recall that all IP Switches keep a copy of every REDIRECT message sent) and it receives the flow information. The reader interested in the IP header reconstruction process can compare the IP header, the REDIRECT message (figure A4.8) and the flow encapsulation. One important thing is the TTL field. Remember from the previous section that the CSR/FANP protocols did not allow the TTL field to be decremented at each hop. This is not a serious problem when routing coherence is maintained on a permanent basis (except when hop count is needed for application like "traceroute"). However, this is not always the case. IFMP implements TTL decrementing another way: TTL field alone cannot be decremented along the redirected VC and reinserted at the last IP Switch because IP checksum field would then be corrupted. The method used first subtracts the initial TTL value from the initial checksum, then a new TTL that is decremented at each IP Switch is inserted in the flow identifier and finally, the last IP Switch on the redirected VC adds the TTL it receives in the flow identifier to the checksum it has.

Note that TTL decrementation does not arise in the classical way. The TTL present in the flow identifier is not decremented dynamically by each IP Switch (an identifier does not change). At each redirection hop (we mean at each intermediate IP Switch along the redirected path), the TTL value of the flow identifier sent in the REDIRECT message corresponds to the initial TTL value decremented by one. The egress IP Switch can then add the TTL received in the Flow identifier to the checksum without problem. The checksum is guaranteed to be correct, in compliance with the IP header regenerated by the egress IP Switch.

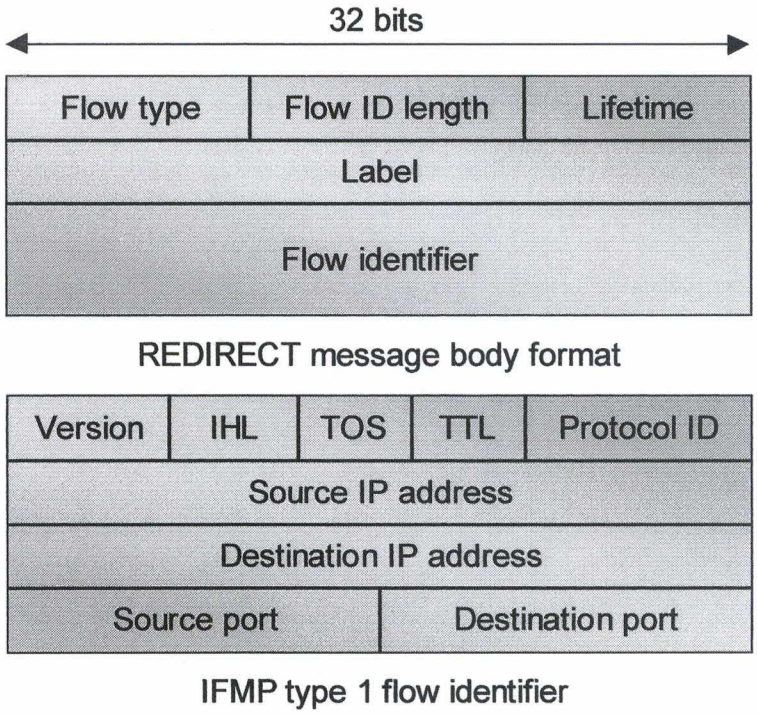

Figure A4.8: IFMP REDIRECT message body format

Let us turn into GSMP (see [NEH+96[2]]) now. The two components of GSMP (master and slave) are connected via an ATM link. The master portion establishes and releases VC connections across the ATM switch, adds and deletes leaves to point to multipoint connections, performs port management (up, down, reset and loopback) and requests data (configuration, statistics). The slave, for its part, can inform its master portion if something interesting happened on the switch. The two GSMP portions communicate through encapsulation over AAL5. Note that GSMP does not have to know anything about IP, it only allows control of an ATM switch independently from the architecture of the switch controller: GSMP can be viewed like an optimization of IP Switching since a custom switch control software could be written for each different switch (even if not an efficient approach). GSMP does not represent an important part of the IP Switching architecture at a technological point of view because a protocol between an external controller and a switch seems to be of little use compared to inter-switch communication provided by IFMP. Even if we relativize GSMP importance in IP Switching, it should be insisted on the fact that GSMP represents a very useful technique to control ATM switches with an exterior software component.

A4.3 Cisco's "Tag Switching"

Like all the other approaches, Tag Switching (see [RDK+97]) has its own terminology: labels are referred as tags, LSRs as Tag Switching Routers (TSRs) and LDP as Tag Distribution Protocol (TDP). A Tag Switching network is made of Tag Edge Routers (edge LSRs role) and Tag Switching Routers (LSRs role). Tag Switching design goals focuses on adding functionality such as explicit routes for traffic engineering and improving scalability. Tag Switching does not restrict itself to any particular link layer technology and can be implemented on a variety of devices, not only ATM switches. No special hardware is required to implement Tag Switching on routers or switches.

In the previous IP switching approaches, the sole functionality provided[30] is destination-based routing. In the current approach (in ARIS as well), it is one of the many functions. A TSR participates in unicast routing protocols just like ordinary routers to construct its mapping between FECs and their next hops. However, the TSR does not use this mapping for the actual packet forwarding. The Tag Switching control component uses this mapping to construct its Tag Forwarding Information Base (TFIB), which serves for the actual packet forwarding. To construct the entries in the forwarding table, the TSR needs three sources of information: the local binding between FECs and tags, a mapping between the FEC and its next hop, and finally the remote binding between FECs and tags (received from the next hop). The construction of forwarding entries (local binding for a particular FEC) works as follows:

1. The TSR selects a tag in its pool of free tags;
2. The selected tag is used as an index in the TFIB (determines the particular TFIB entry that has to be updated);
3. The incoming tag in the TFIB entry is set to the selected free tag from step 1;
4. The next hop for the TFIB entry found in step 3 is set to the address of the next hop associated with the FEC;
5. The outgoing interface is set to the interface that should be used to reach the next hop.

Once the local binding has been done, the TSR may advertise its set of *<address prefix,tag>* pairs where *address prefix* identifies a FEC and *tag* defines the tag value used by the local TSR. The TSR is waiting for remote binding information. When it receives remote binding information, if it relates to a FEC that has already received local binding and if the interface by which it received the remote binding information corresponds to the one it uses as outgoing one for the same FEC, then the outgoing interface of the TFIB entry corresponding to that FEC is set to the remote label value (see figure A4.9 to help visualize this "somewhat" complicated explanation).

If no local binding information exists about the remote FEC, the TSR may either discard the remote binding information (in which case it must be able to ask the other TSR to send the binding information one more time) or keep it for later use. If the TSR receives remote binding information on an interface that does not corresponds to the next hop for the FEC, the former choice applies. The issue concerning the mechanisms for distribution of tag binding information depends on the routing protocols used to construct the FEC to next hop mapping. Depending on whether the mapping is constructed via link-state or distance vector routing protocols, the tag binding information will be provided through a Tag Distribution Protocol or piggybacked on the routing protocol. When the construction of FEC to next hop mapping relies on a link-state routing protocol (e.g., OSPF), routing information is flooded unmodified among a set of routers that participates in the routing protocol. On the other hand, tag binding information need only be distributed between adjacent TSRs. This makes piggybacking on a link-state routing protocol not suitable. Consequently, the use of a TDP should be considered. In the distance vector case, the extensibility of the routing protocol determines the use of a TDP protocol. With RIP or RIP-II, a TDP will be used since modification in a backward-compatible fashion seems tricky, if not impossible. By backward-compatible, we mean that changing the protocols will not affect the behavior of older versions of the protocol (modifications will not cause the older protocol versions to misbehave or crash). On the other hand, BGP allows piggybacking in a backward-compatible manner thanks to its optional attributes. The knowledge of TSR peering relationship necessary for tag binding information distribution takes place through routing protocols running on the TSR. Every routing peer is included in the set. In general, all TSRs that share a common subnetwork with one of the interfaces of the local TSR belong to this group. Given these functional considerations

[30] Any IP switching approach must provide this fundamental capability.

and according to the taxonomy defined in section 2.5, Tag Switching can be considered as a control-driven, downstream label binding and independent label binding creation approach.

We said at the beginning of this section that scalability was an important design goal. To see why this is true, we have first to consider today's Internet routing architecture. The routing architecture used in the Internet today may be viewed as a collection of routing domains, each implementing intradomain routing protocols. At the same time, border[31] routers of every domain interdomain routing protocols. This (partial) partitioning of routing information reduces the volume of routing information routers need to maintain, improving the scalability of the network. Nevertheless, this incomplete partitioning does not resolve the problem of transit networks that have to maintain interdomain routing information. Even interior routers have to know about the interdomain routes since transit traffic must be forwarded across the domain. We see now why explicit labels are not suitable in large networks: the semantics associated with global addresses implies that true scalability cannot be achieved easily with classical IP routing. Trying to implement hierarchical routing in the Internet will not be possible unless label switching is used to forward packets across intermediate domains. There will always exist a limit to scalability of network layer routing when using purely explicit routing decisions. Implicit-based routing decisions allow forwarding packets within transit domains without requiring the interior routing devices to keep any exterior domain reachability knowledge. It should be pointed out the fact that there exists a strong correlation between scalability and hierarchical routing decisions. The problem with explicit routing is that intradomain routers should not have to care about exterior routing information, only about how to reach exterior prefixes. The only thing intra-domain routers should have to do with transit traffic is forward it without knowing its interdomain origin and its interdomain destination since this information is senseless (recall that we are talking about intradomain routers). Hierarchical routing must be strongly related with hierarchical routing knowledge, which in turn translates into implicit routing decisions between distinct levels of the routing hierarchy. Scalability will not be achieved without implementing different routing levels. Hiding part of the routing information to the other routing levels of the network is the only means to attain scalability. When a change happens to a transit route, the interior nodes should not explicitly know about it. The only event they should experience is the modification of the label binding corresponding to that FEC (removal for some interior nodes and eventual creation for others).

Tag Switching implements this kind of implicit routing through the creation of labels corresponding to transit routes established on edge TSRs initiative. Recall that edge TSRs participates in inter-domain routing. Because of its control-driven label binding method, once the interdomain routing protocol creates an entry for exterior domain reachability information, a tag binding information flows across the local domain TSRs from the egress point to the ingress point of the transit route (since label binding flows from downstream). Tag Switching supports hierarchical routing through a stack of tags. The width of the stack carried by each packet reflects the routing levels the packet will traverse. For the simple example of a single transit domain in a Tag Switching network, suppose that the origin host is located in domain A, the transit network is domain B and destination host is situated in domain C. The first TSR in domain A that receives the packets from the origin host pushes a tag (creates a stack of width 1) on the packets, the tag pushed corresponds to the FEC which destination is the border TSR that permits to attain domain C. This implies that exterior reachability information is known by every intra-domain TSR. This information is provided through creation of label binding between tags corresponding to border TSRs and address prefixes of domains reachable via a particular border TSRs. The border TSR of domain A swaps the current tag and forwards the packet to the border TSR of domain B, the next hop for domain C according to the interdomain reachability information. The border TSR of domain B receives the packet, pushes a new tag associated with domain C prefix on the stack, allowing the packet to transit across domain B without any intradomain B TSR explicitly knowing the packet's destination. All interdomain TSRs on the path swap the tag. At the egress of domain B, the border TSR pops the tag off the stack and forwards the packet to the border TSR located in domain C based on the remaining tag. The border TSR of domain C pops the tag and pushes a tag that will

[31] A border router is a router connected to both routing domains (interior and exterior).

permit to reach the destination host (since it participates in intradomain routing). If another level of routing domains exists in domain B, the tag stack would have one more tag width when traversing one of the domains contained in domain B.

We said that destination-based routing was just one of the many functions of Tag Switching: the main others are multicast and explicit routes support. Multicast routing relies on the concept of multicast distribution tree. Multicast routing procedures have as an objective the construction of spanning trees (the leafs being the receivers and the root the sender) while multicast forwarding cares about forwarding multicast packets along these trees. Tag Switching supports multicast forwarding function via the association between a tag and a multicast tree. When a tag switch creates a multicast forwarding entry and the corresponding list of outgoing interfaces, it also creates one local tag per interface. The binding between the previous local tag (associated with each interface) and the tree is advertised on each outgoing interface. When a tag switch receives a binding between a multicast tree and a tag from another tag switch, if the other switch is the upstream neighbor (in the multicast tree context) then the tag is placed by the local tag switch in the incoming part of the multicast forwarding entry. Note that for multiple access networks, the tag allocation scheme for multicast must be coordinated among the switches. The next functionality allowed by Tag Switching is the support for explicit-routes (routes that do not correspond to the destination-based routing paths). This allows implementing load balancing among multiple links without being obliged to use Frame Relay or ATM to do it.

Tag Switching and ATM are quite similar. This allows running Tag Switching on unmodified ATM hardware by replacing the ATM Control Plane with the Tag Switching Control Component. The tag information may be carried in the VCI/VPI field. This allows having two levels of tagging. In practice however, the limited VPI field space constraints the size of networks. The control information might be provided by Network Layer routing protocols like OSPF, IS-IS, BGP, etc. There is one issue when supporting destination-based routing with Tag Switching on an ATM switch: several tags associated with a route (or a group of routes) should be maintained to avoid interleaving of packets that arrive from different upstream tag switches but are sent concurrently to the same next hop. Tag Switching over an ATM switch might constitute a good solution for the ATM switches/routers integration since an ATM switch capable of tag switching would appear as a "normal" router to adjacent routers. Furthermore, tag switching over an ATM switch removes the need for the ATM control plane. We might imagine a partitioning of VCI/VPI space and resources to allow the ATM switch to work as a true ATM switch as well, i.e. "ships in the night" operation mode.

Note that in state 1 of figure A4.9, the output port is defined (value = y) because FEC to next hop mapping is done on a local basis.

As mentioned previously, piggybacking on top of routing protocols should be the favorite way to distribute label binding information. Nevertheless, this option does not always match with routing protocols realities. Tag Switching provides thus its own protocol to distribute label binding information: the Tag Distribution Protocol (TDP). Routing information exchange between TSRs is coupled with a TDP session in order to exchange tag binding information for the routes that have been constructed from routing information. TDP uses the technique of incremental updates. TSRs advertise changes to the tag binding information. To meet the reliability needs of incremental updates, TDP uses TCP as transport protocol (from the experience gained with BGP).

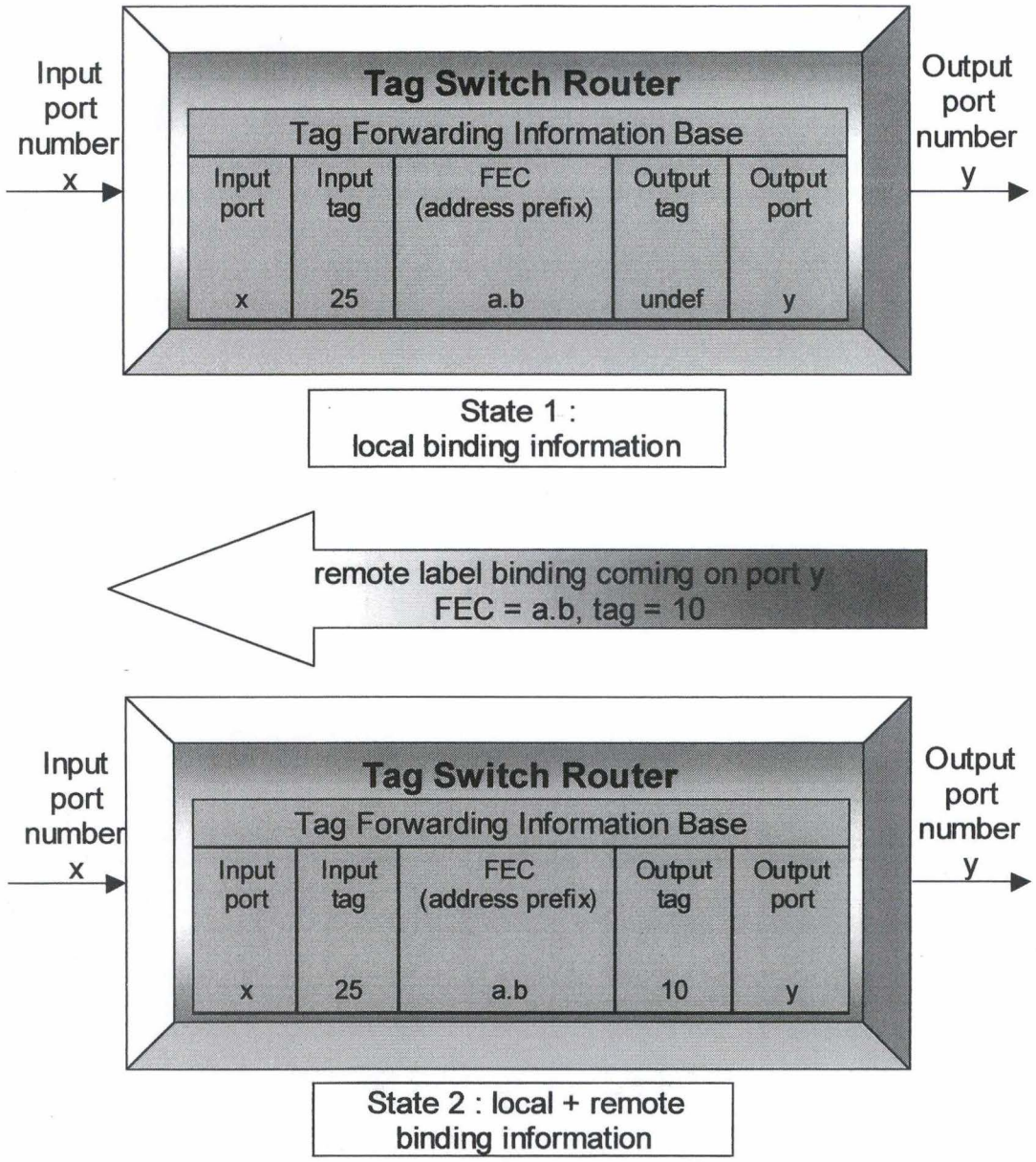

Figure A4.9: TFIB entry creation example

A4.4 IBM's "Aggregate Route-Based IP Switching"

ARIS (see [FV97]) was developed in parallel with Tag Switching. This is the reason why these approaches have so many points in common. The ARIS name, Aggregate Route-based IP Switching, should have something to do with its origin: ARIS binds labels to aggregate routes rather than flows.

ARIS uses a control-driven approach just like Tag Switching. Label bindings are thus set up in response to control traffic. Label switched paths for destination-based routing are established according to the downstream allocation and the ordered creation models. LSRs are referred to Integrated Switch Routers in ARIS terminology. ARIS assumes that ISRs are running conventional routing protocols and ARIS operation depends on the type of routing protocol used. The ISRs need to know the type of egress ID (an egress ID identifies an egress ISR and must be unique within a label switching region) used according to the type of routing protocol.

Suppose that we take the case of a link-state protocol and that all ISRs in the region we consider are located in the same routing area. In that case, all ISRs have a complete map of the topology. Therefore, every ISR can determine the egress ISR for every entry and not only the next hop. The establishment of a label switched path begins with the advertisement by an egress ISR of a binding between a label and its own egress ID to its upstream ISR neighbors[32] (this is why egress IDs must be unique within a given network). The choice of egress IDs occurs via ISRs agreement. Each ISR checks upon receiving of an advertisement from a neighboring ISR whether the message came from the expected next hop for that egress ID (by checking the routing table entry for that egress ISR). If the message came from an inappropriate next hop, it is simply discarded. If it came from the right next hop, the ISR records the binding between the egress ID and the advertised label. The ISR now knows that this label may be used to forward packets which route pass through that egress ID. The local ISR then generates a local binding between the egress ID and a locally chosen label. This label will be used as incoming label for the routes associated with that egress ID. This local label will be advertised to every ISR neighbor. ISRs will eventually discard the message according to the fact that it is received on the right interface for the advertised routes. Note that the label binding creation process generates a multipoint to point tree rooted at the egress ISR (see figure A4.10). Label distribution happens on a controlled (ordered) manner, starting at the egress and propagating towards the ingress(es).

When a distance-vector routing protocol is used, one can no longer assume that each ISR knows the complete topology of the network. In that case, ISRs only know the next hop for each prefix. The solution for this problem consists in changing the definition of an egress ID. The new egress ID identifies a prefix carried in the distance-vector protocol. An egress ISR will thus bind a label to the egress ID and advertise it if it had a route to that prefix whose next hop lies outside the ISR region. Remark that this method consumes more labels than the link-state one due to the use of one label every time a new prefix becomes reachable (even if several prefixes are reachable via the same egress ISR).

Like the previous approach, ARIS implements a loop prevention scheme as well as a TTL decrementing mechanism. The principle of loop mitigation consists in checking whether an ISR ID appears in the list of ISRs during the establishment of the label switched paths. The TTL problem is resolved by decrementing from the packets the right value at the ingress ISR of the path (ingress ISRs must know the hop count across the ARIS region). When the ingress ISR determines that the TTL will reach the 0 value before the end of the label switched path, the packet is forwarded on a hop-by-hop manner so that ICMP messages will be generated by the ISR that sees TTL=0.

ARIS supports explicit routes through a new type of egress ID. A list of specified IP addresses may be specified to force a strict source route. In this case, the path establishment may be initiated at either the ingress or the egress.

One aspect on which the designers of ARIS have focused is the deployment over ATM. The focus on loop prevention shows that ATM characteristics have driven the design of ARIS. Recall that paths are multipoint-to-point in ARIS. This raises some challenges because ATM cells from different packets but the same label might interleave at the link level. The problem only arises in ARIS and Tag Switching because the other approaches (IP Switching and CSR) do not require label assignment to be based on destination only. The solutions proposed are VC-merge, VP-merge and assignment of multiple labels. In VP-merge, the label is carried in the VPI field only, VCI values identifying the sources. The drawback of the approach relates to the limited space of VPI field and the necessity to assign an identifier for every ingress ISR. Another important drawback comes from the fact that the majority of ATM hardware cannot perform EPD (early packet discard) or PPD (partial packet discard) when switching on the VPI field only. These congestion control features are thus not available. VC-merge consists in mapping many IP routes to the same VC label, requiring reassembly buffers so that cells belonging to different packets and intended for the same destination do not interleave with each other. When neither VC-merge nor VP-merge is available, some mechanism is needed to prevent cell

[32] Which are all ISRs directly connected to it.

interleaving to interfere with the correct reception of frames. The only solution consists in allocating different labels to different sources. In that case, the modified path establishment causes label binding to flow from ingress to egress[33]. The per-*<ingress, egress>* pair label prevents cell interleave.

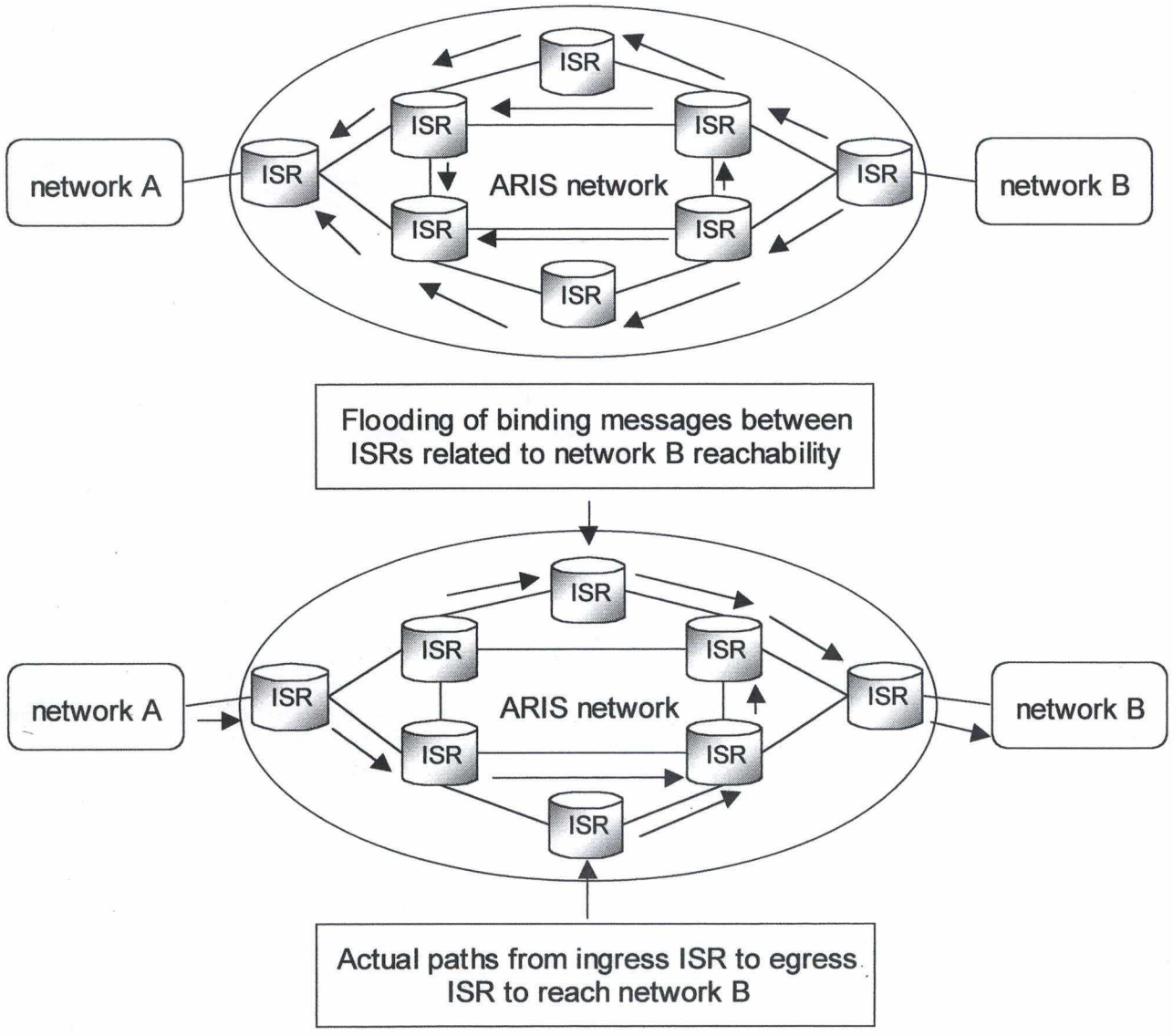

Figure A4.10: ARIS label binding creation

The ARIS protocol contains six types of messages: INIT, KEEPALIVE, ESTABLISH, TRIGGER, TEARDOWN and ACKNOWLEDGE. The INIT message is the first message exchanged by ARIS neighbors to establish adjacency. It must be periodically transmitted over each ARIS link. When a successful INIT message exchange arises, the neighbor state is transitioned to ACTIVE. All other ARIS messages may then be transmitted. The INIT message contains neighbor timeout period and the label ranges supported by the ISRs. The KEEPALIVE message is sent periodically to keep the adjacency up (in the absence of other data flow). The ESTABLISH message serves to establish a binding between an egress ID and a label. It also contains a list of routers that have been traversed between the egress and the current ISR. It is used to check whether the message originates from the expected next hop for the egress ISR and if the path is loop-free. If both conditions are true, the recipient replies with an ACKNOWLEDGE message. The sender of the ESTABLISH message must receive an ACKNOWLEDGE message (positive or negative) unless it will keep re-transmitting the

[33] Upstream label binding is used in this case.

message. ACKNOWLEDGE messages are used because it runs directly over IP. In Tag Switching, TCP removes the need for explicit retransmissions. The TRIGGER message serves the purpose of explicitly requesting a new binding rather than waiting for the next ESTABLISH message. It is used when a routing change causes the next hop to some egress ISR to change. The upstream node sends then the TRIGGER message toward the next hop, which should respond with an ESTABLISH message. The TEARDOWN message is the opposite of an ESTABLISH message, enabling an egress ISR to withdraw bindings previously advertised (in the case an egress ISR is no more the egress for a particular route because of the wake-up of some ISR or some routing reachability change).

A4.5 Evaluation

This section evaluates the four previous approaches according to the taxonomy defined in section 2.5. The implications of the design decisions will also be discussed. Our intent relates to distinguishing characteristics that have an influence on label switching performance and scalability properties. We wish to make apparent global design choices that have direct and important implications on any label switching implementation. The most significant characteristic of any of the four approaches lies within the label binding model: the choice between control-driven and data-driven.

When trying to evaluate performance of the different schemes, the problem that appears is the one of the conditions under which the evaluation is done: "What are the ideal conditions?" The best case depends on so many variables that it may be virtually impossible to present a quantitative evaluation for every traffic pattern experienced by every type of network. The "best case" for forwarding performance is obviously that of the ATM switch. Since most ATM switches can forward traffic at "wire speed", if no interface were congested and no change in label binding was ever experienced, the label binding scheme would provide the same throughput as the hardware does. However, "ideal conditions" cannot be achieved in real life networks because routing changes occur and traffic flows are not infinitely long-lived. This implies that control-driven and data-driven schemes both experience non-ideal traffic conditions.

For data-driven schemes, "ideal conditions" correspond to infinitely long-lived flows that allow the label path setup to be amortized. However, if a packet is not label-switched, it has to be processed by the control processor. Unfortunately, the control processor often is a conventional router implemented in software with some flow detection code and label distribution protocol. Therefore, we need to take care of the burden we place on this device[34]. The packet forwarding capacity required for the control processor depends on the rate at which new flows arrive and the number of packets that are not label-switched within a flow. The underlying problem resides within the fact that if the forwarding processor cannot sustain the packet load, some functions may have to make for it: data packets dropping, flow identification process and routing updates processing lessening.

Packet per second forwarding capacity required ≥ packets per second for control + data packets per second

Data packets per second = (data packets required for flow establishment * number of flow establishments per second) + number of data packets per second that do not trigger shortcut-VC establishment

[34] This device must also run routing protocols.

Some researchers have considered that performance of data-driven schemes under real traffic conditions can be quite good, with a high percentage (at least 70 % according [NEK+99]) of the traffic label-switched. However, Chapter 5 provides some different view of the performance of label switching for interdomain flows. Pure data-driven schemes probably will not sustain interdomain flows variability. Some hybrid scheme may be required to limit the overhead of connection establishment and release.

For control-driven schemes, "ideal conditions" rhyme with no routing change. That means that when the topology is stable (and traffic load is low enough for every LSR), no packet will be processed by the control processor and thus hardware speed will be achieved. Note that this is true with destination-based routing, not in the multicast case. Change in the multicast trees does not always occur due to topology change but to group membership change. Unlike the data-driven scheme, such an ideal situation may happen. It should be pointed out that control-driven schemes might achieve ideal performance even under topology change. This might be true because LSRs may learn alternative routes for a particular prefix from an LSR that was not the next hop before the change but becomes the next hop after. In this case, the change of shortcut-VC may occur almost instantaneously in hardware. Meanwhile, a few packets may be lost or forwarded through classical IP routing. A change in the topology also affects the data-driven schemes because the LSR that becomes a new next hop receives many flows. Therefore, it has to conventionally forward packets from the new flows at first waiting for the establishment of shortcut-VCs for them. In fact, the flow detection code has to first determine whether the data received will be label-switched. This takes some time to analyze and places some burden one the forwarding processor. Control-driven schemes also suffer from a performance drawback in situations where route aggregation occurs. We see here one of the many situations in which a conflict appears between scalability and performance. When route aggregation is made within a label switching domain, routing information is also aggregated. Therefore, the egress LSRs that did aggregate the routing information might be constrained to perform some conventional forwarding for the LSPs that use the aggregated information because the egress LSR is the only one to have non-aggregated routing information. Nevertheless, routing information aggregation does not happen by accident. Network designers often can predict the impacts of such an aggregation and take care of the burden placed on the egress LSRs. Data-driven schemes do not suffer from this type of aggregation since the flow definition generally occur higher in the OSI model layers (although they might use a network-to-network flow definition). For what concerns scalability, control-driven schemes seem to have an edge over data-driven ones because topology changes are due to occur less frequently than flow setup and teardown in data-driven schemes. [NEK+99] shows that both approaches (data-driven and control-driven) require a quite a large number of labels but aggregation strategies such as label mapping policies that are combination of the two schemes may be useful in backbone areas. It proposes a label mapping with the aggregated packet stream toward a specific destination network, triggered by the actual packet arrival belonging to the defined aggregated packet stream. This policy is reported to result in an important decrease in label range requirement and an increase in cut-through ratio (proportion of traffic that is label-switched). Chapter 5 discusses interdomain flow variability and techniques to enhance label switching performance. The issue of scalability is and always will be present because we will not get scalable and high-performance networks without making policy decisions. There will always exist a place where a compromise between performance and adaptability will have to be made (recall what happened with IP routing and the integration of IP and ATM). The size of the network as well as its purpose could give you an idea of the policy that should fit with it. Scalability is an issue for a backbone while not always for a class C LAN.

Another important choice concerns independent versus ordered binding creation. The main difference appears when we look at the FECs selection process. With the independent scheme, every LSR may use its own definition of a flow and correspondence between packets and FECs is free. With the ordered scheme, LSRs are constrained by remote binding that adjacent LSRs advertise so they all must use the same flow definition. Independent schemes need thus some configuration in order to make the FECs to label binding decision somewhat useful: if every LSR uses a different flow definition, LSPs are of no use. Downstream creation presents a drawback compared to upstream since more packets have to be

forwarded the conventional way during the propagation of establishment demands along the shortcut-VC.

Label distribution protocol issues require some remarks. First, transport reliability may be achieved in two ways via either TCP or the LDP. Reliability and in order delivery is guaranteed by TCP, the most widely used reliable transport protocol. Otherwise, the LDP has to deal by itself with reliability and in order delivery. In that case, one must take care when a binding advertisement is followed by a withdrawal of that binding because order is important. When one looks at the problems encountered at tuning TCP in order to improve performance and robustness, we argue in favor of well-tested protocol rather than trying to start from scratch. Second, the issue of piggybacking label distribution on top of existent routing protocols is bound to the synchronization of label binding. A choice has to be made between obtaining synchronization via control traffic and the necessity to modify more protocols to achieve this functionality. Care must also be taken with piggybacking because label binding information must not attain devices that might be disoriented by information they do not understand (by preventing the information to attain them or make the devices ignore it). Third, the hard or soft state depends on the characteristics of the routing protocols used to piggyback label binding information. Some routing protocols use hard state like BGP and some soft state like PIM. When a LDP is used, the underlying transport protocol dictates the eventual need for refreshing of label binding state. If the binding information is assumed stable, there is no need to refresh the binding information and hard state should be used instead. Only changes of binding information will be transmitted in that case. On the other hand, when label binding information or routing information is known to be non-stable, KEEPALIVE-like messages, explicit acknowledgments and soft-state might be more adequate. Remark that data-driven schemes fit naturally well with soft-state due to their intrinsic dynamic condition. On the opposite, control-driven schemes allow for communicating changes of binding information. They tend to facilitate the use of hard-state label binding information. In any case, the cost of the eventual overhead caused by the extra reliability mechanisms (timers, retransmissions, etc.) must be considered.

Now that we have toured around several label switching implementations, did we made steps forward toward IP over ATM integration? The IP over ATM integration motivation was primarily the overcoming of conventional routing shortcomings (pure performance and routing functionality pressure). Meanwhile, conventional routers (routers that forwards IP packets by a full examination of the IP header) have evolved towards multi-gigabit throughputs. The label switching claim of performance improvements while at lower cost than conventional routers may be questioned. The snag arising at trying to evaluate the cost between current routers and LSRs is that real life products cost is driven by a mixture of technical and economical factors. The insight we will give concerns the limited impact of label switching on the cost of the whole routing system due to the cost of the components (switch fabric and buffers), which increase as link speed increases. In fact, given that buffer memory might constitute a significant part of the overall device cost, the benefit of label switching does not appear overly convincing. An argument pointed by Ipsilon is the one of ATM hardware becoming "commodity" items due to the large demand for simple ATM switches. The label switching approaches that do not rely on ATM signaling procedures will drive the increasing demand for ATM switches. That increase in switches demand might in turn drive cost reduction in ATM switches chipsets. Since LSRs may be implemented through low cost ATM switches controlled by a PC (running the control part of label switching), LSRs might very well become low cost devices. We said in Chapter 2 that the important shortcomings of conventional IP routing were the routing functionality. The advantages of label switching relates to explicit route support, better scalability and more stable interior routing (through better separation of interdomain and intradomain routing), IP/ATM integration (through removal of complex ATM signaling) and evolvability (adding new routing functionality becomes easier thanks to separation of control and forwarding functions).

Appendix 5

Tools for Interdomain Traffic Analysis

A5.1 Introduction

The aim of this Appendix is at presenting the tools we used during the traffic traces collection phase. We also present a summary of the code we developed to transform these traces into useful information, information that may be interpreted. Section 2 of this Appendix briefly presents the architecture of the collection environment that served at gathering our data. Section 3 presents a summary of the most interesting code used during the data transformation and analysis part.

Chapter 5 presents our analysis of interdomain traffic traces in some insidious way: it seems that the core of our work has been the analysis and results interpretation phases. This is completely wrong! The main part of the traffic traces analysis task lies within gathering data and writing code to transform traffic traces into data files. The burden of that part of our work has revealed to be unexpectedly important. We could not have thought before entering the actual gathering and coding that it would be so difficult. From the outside, one could think he just needs to get some traffic traces files and use some existing code to get similar results. We got several problems and constraints, which influence was unexpected. Files size, CPU processing time, inconsistencies within results... Doing this kind of job should have actually required some more persons to be carried in a complete and systematic manner. We did it in parts, some result would have required more processing capabilities, some other ones could not have been carried even with the strength of our will.

Because of (and thanks to) our will to use the Linux environment and GNU software, everything related to development and configuration has been easier compared to what might have happened by working on a Windows platform. We doubt it could have been possible to carry such a study on a non-UNIX platform. Linux may be considered as ideal for what concerns parameterization and easy development. Without mentioning the fact that we did not spend a penny at any software license...The costs were only for hardware, power supply, human resources and last but not least time[35]. So at least it should be pointed out that interdomain traffic analysis is not a costly research activity.

A5.2 Software for Traffic Traces Collection

A5.2.1 Export Part

Because we gathered interdomain traffic traces, we relied on the *Netflow* export (see [Cis99]) capability that is present on most Cisco routers. Netflow export permits to collect statistics for every flow traversing a router and to send them to a particular host in the Internet. *Netflow* architecture is composed from routers exporting their traffic statistics using UDP datagrams and a collector workstation collecting these datagrams. Routers exporting *Netflow* data send the UDP datagrams towards a specific IP address and a specific UDP port number. It has the advantage of summarizing the trace but the drawback that traffic within a particular flow is linearized according to the time interval configured on the collecting host. Given that collector hosts have limited processing ability, a time

[35] Coffee should also deserve a mention...

interval in the order of several seconds seem impossible to sustain when the amount of traffic seen by the exporting router becomes important (which is due to be the case for interdomain traffic). Note that using *tcpdump* or some other packet-level sniffer is not likely to be a good idea unless you dispose from several "beasts" having some serious number of CPUs. The granularity provided by *Netflow* probably better fits interdomain traces than packet-level precision. The earlier the aggregation, the less post-processing activities that will have to be performed.

A5.2.2 Collector Part

The collector running on our workstation was Caida's *cflowd* software (see [CAIDA]). The *cflowd* system architecture comprises three elements. The first component, *cflowdmux*, handles raw flow data exported by routers implementing *Netflow* export. The second component, *cflowd*, maintains tabular data for each router and passes it to the third component, *cfdcollect*. *Cfdcollect* retrieves tabular data from *cflowd*.

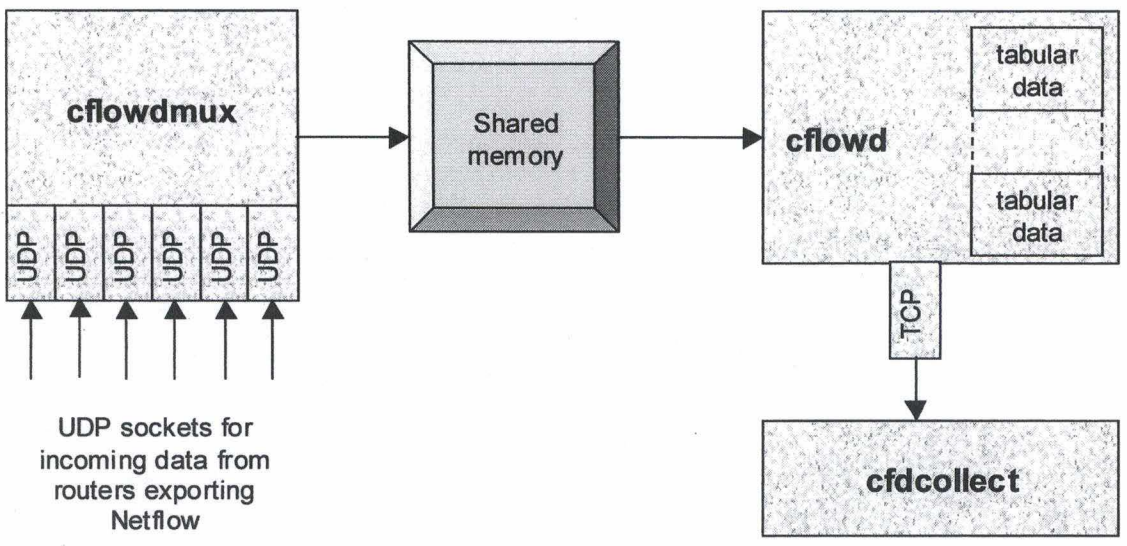

Figure A5.1: The *cflowd* system

Cflowdmux writes *Netflow* export packets onto shared memory for packets arriving on UDP sockets. It does not modify the content of the UDP datagrams data so it is usable as a standalone program. *Cflowd* monitors the shared memory and reads a buffer when it becomes available. A semaphore is used to manage the access of the shared memory because both *cflowdmux* and *cflowd* are due to access it at the same time. *Cflowd* converts the *Netflow* export data into tables for which it is configured. It listens to client connections on a TCP socket, sends the client the data it requested and clears its tables. *Cfdcollect* retrieves and archives tabular from *cflowd* at regular intervals. This generates time series data in ARTS files, by means of the *arts++* utilities. The *arts++* package provides APIs written in C++ for accessing and manipulating ARTS files.

A5.3 Software for Traffic Analysis

We were lucky enough to be able to use existing software for data collection. This was not the case for data transformation. We thus had to start "from scratch". The choice of the programming language was an issue because of the expected size of the data to analyze. We chose Perl (see [PERL97]) due to its outstanding string processing features and its hybrid nature (between C and bash).

Most Perl scripts[36] grossly performed the same operation: take as argument one or more files and transform them to generate one or more new files. The whole traffic analysis may be viewed as an incremental processing with very distinct steps. Figure A5.2 abstracts this file transformation process. The pre-processing phase aiming at getting ASCII files from the ARTS binaries simply consisted in applying some *arts++* executable on a specific ARTS file in binary format (giving traffic statistics for a whole day). The result was an ASCII file which format depended on the particular executable applied. These ARTS files (ASCII) were quite big when they were related with the research ISP (Belnet): around 750 Mbytes. Working on such files was not possible due to processing time. Therefore, the role of the Perl scripts was to perform this processing phase. Due to space limitations, we only present the main part of the code in terms of data processing. Many other scripts have been written, tackling other problems: script generators, self-similarity checkers, bandwidth reservation techniques evaluators, BGP topology analyzers... Explaining all scripts would have required a separate manual but this is out of the scope of this dissertation.

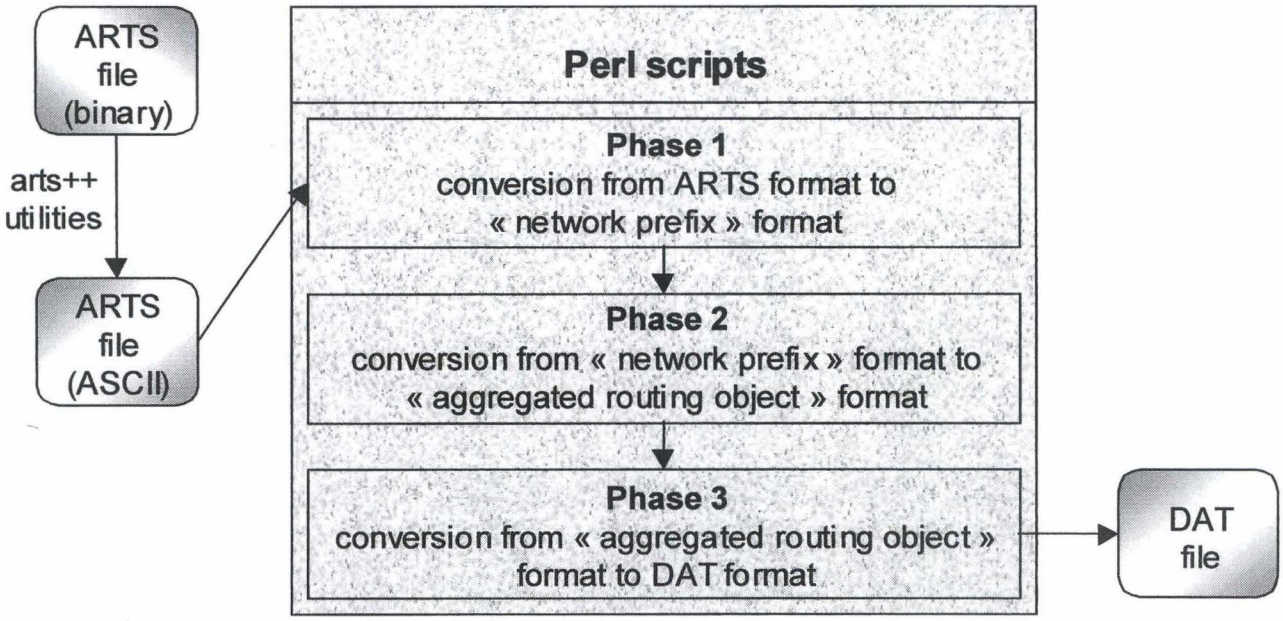

Figure A5.2: Traffic traces transformation process

A5.3.1 Phase 1

Because ASCII ARTS files were "somewhat" too big, some compressed format was necessary. The *arts++* executable we used was *artsnets* due to our need to get the less aggregated BGP object. End-to-end IP flows were out of the scope of our study so the coarser granularity had to be the network prefix (in its BGP sense). A network prefix in the BGP sense is a *<network prefix,masklength>* pair corresponding to a best-matching entry in the BGP table. The advantage of *arts++* is that routing can be simulated thanks to a BGP4+ object giving for each 32-bit IP source address the corresponding entry in the BGP table. This was only true for outgoing traffic, not for incoming. The interesting part of an incoming flow is the source address, which is an external address that is not routed. We thus had full 32-bit addresses for the source part of incoming flows. This was due to be a drawback since we had to route these addresses while *artsnets* on outgoing traffic gave routed prefixes. In fact, it appeared that something was wrong with outgoing prefixes[37] because we got two times more prefixes over the day than for incoming traffic while incoming traffic was far more important than outgoing. We did not

[36] A Perl program is a script because Perl is an interpreted language.
[37] The one routed through *arts++*.

found the reason of this odd behavior of *arts++*. The format of the output of an ARTS file processed by artsnets is shown on Figure A5.3. It gives for every *cfdcollect* time interval (one minute in our case) as many lines as there are <*source network prefix,destination network prefix*> pairs for which traffic was seen by the router exporting *Netflow*. Studying interdomain traffic is only affected by exterior prefix behavior, not interior prefixes. A problem arising from traffic traces collection concerns anonymity. We had to anonymize all addresses interior to the studied ISP because we were not supposed to know anything about a particular internal IP address. We thus transformed every internal address by giving them a value of "10.0.0.0/0". This ensured that nobody would be ever able to determine the internal address associated with a particular flow. It is therefore impossible to study the behavior of a particular host with our "post-*artsnets*" files.

```
router:  193.190.197.253
ifIndex: 26
period:  12/15/1996 00:59:04 - 12/15/1996 01:00:01 CET
         Src Network         Dst Network          Pkts          Bytes
  ------------------  ------------------  ------------  ------------
     205.189.33.0/24      224.2.202.0/24         19656      29462799
      130.149.0.0/16      224.2.231.0/25         15995      19863130
     209.246.14.0/24     204.212.44.0/22         15805      16652197
     212.27.32.91/20    192.172.226.0/25          6040       9053462
```

Figure A5.3: ASCII ARTS file format

Our "post-phase 1" data format had two lines per *cfdcollect* time interval. Each first line gave all external prefixes that did experience traffic during the considered time interval while the second showed the traffic in bytes associated with the prefixes of the first line following the same order. All lines were prefixed with an integer timestamp incremented by one at every time interval (often from 0 to 1439 for one day of traffic). Figure A5.4 gives an example of the "post-phase 1" file format.

```
756 216.237.128.0/18 216.35.0.0/19 147.126.0.0/18 207.226.240.0/20 ...
756 56816453 91473999 106645331 119252346 ...
```

Figure A5.4: "Post-phase 1" file format

Note that *artsnets* also sorts the network prefix pairs by decreasing number of bytes exchanged during the time interval. Our format also had this sorting for each couple of lines for a given time interval. This file format did limit the file size from several hundreds (more than 500 Mbytes) of Mbytes to 150 Mbytes.

A5.3.2 Phase 2

Even if our "post-phase 1" file format was not optimal in terms of file size, its very general structure enabled to write most of our scripts without having to take care of the type of object that actually was used in each "first line" for a given timestamp. Hence, having the BGP routing table information permitted to compute the traffic associated with ASs objects or inter-AS links objects (see Appendix 6 for an introduction to BGP routing). Phase 2 scripts consequently took as input one "post-phase 1" file and the BGP routing table with which the traffic had been routed. Since a BGP entry contains the <*network prefix,masklength*> pair and the AS vector of all traversed ASs to reach the network prefix, replacing the network prefix from the "post-phase 1" file by an AS number or several inter-AS links was conceptually simple. All we had to do is finding the entry corresponding to the <*network prefix,masklength*> pair in the BGP table and replace it by the last AS number of the AS vector or to attribute to every inter-AS link of the AS vector the traffic for the entry. The problematic aspect is that each couple of lines from the "post-phase 1" file required as many BGP table lookups as there were distinct network prefixes in the line. Given that the research ISP counted a mean number of several

thousands prefixes for each line, for 70000 entries of the BGP table and 1440 time intervals in our case, we were in the order of several billions of comparisons and lookup operations. To give an idea, it had to take about two weeks of computations just for one day of traffic! Fortunately, the "Memoize" module (see [PERL99]) allowed us to shorten this calculation period to a little less than two days. This module can be used to "cache" the correspondence between input and output values of any function so that lookup operations into the BGP table were transformed into a simple function call[38].

Some explanation need to be given about these aggregated routing objects. Figure A5.5 shows the format of a BGP table entry. We will base our explanation on this figure.

P	Pref	Time	Destination	Next Hop	If	Path
B	0	14 : 03 : 32	4.17.115.0/24	193.190.60.133	atm0	2611 9010 701 13832

Figure A5.5: BGP table entry format

The network prefix used to perform the best-matching algorithm is the "Destination" column. It is a *<network prefix,masklength>* pair. The AS corresponding to the destination should be the last AS number of the "Path" column. We said "should" instead of "must" because sometimes BGP routes are incomplete and the "Path" information is henceforth inaccurate. An inter-AS link is any pair of ASs that are consecutive in a BGP path (and assuming the path is complete). To visualize the difference between these routing concepts, we suggest the reader to have a close look at Appendix 6.

While the code that replaces a routing object from a "post-phase 1" file remains quite simple, the trickiest part lies within the optimizations needed to make accesses to the BGP table fast and efficient. For that purpose, we had to transform all IP address related information into "binary mode" (bit string) information so that comparisons between two IP prefixes were fast. Figure A5.5 shows that part of the code doing a simple BGP routing on some "post-phase 1" format file.

```
#!/usr/bin/perl

# Turns on caching for all subroutines to speed up
# operations at the expense of some RAM space...
use Memoize;
memoize 'bgp2bitstring';
memoize 'bestmatch;
memoize 'dec2bin';

##################################
# Variables declaration section #
##################################

# BGP entries hash
%BGPHASH;

# Temporary lists
@MYLIST;
@PREFIXLIST;
@TRAFFICLIST;
$COUNT = 0;

#####################
# Code begins here #
#####################
#########################################################################
# First part : we take every entry of the BGP table and put it          #
# in a hash table so that we have an index on the routing table.        #
# This method is the fastest in Perl, hashes are the fastest access     #
# data structure the language owns...(read [PERL99] to get convinced).#
#########################################################################
```

[38] At the expense of some additional RAM use of course...

```
$BGP_ROUTING_TABLE = $ARGV[0];
open (BGPFILE,"<$BGP_ROUTING_TABLE");
$CURRENTLINE = <BGPFILE>;
$CURRENTLINE = <BGPFILE>;
$CURRENTLINE = <BGPFILE>;
$CURRENTLINE = <BGPFILE>;
$CURRENTLINE = <BGPFILE>;
# Loop over the BGP routing table file, one line at a time
while (defined($CURRENTLINE=<BGPFILE>)){
chop($CURRENTLINE);
$CURRENTLINE = substr($CURRENTLINE,3,256);
@MYLIST = split(/[\s]+/,$CURRENTLINE);
  if ($MYLIST[0] ne ""){
     $MYLIST[0]=~/([0-9]+\.[0-9]+\.[0-9]+\.[0-9]+)[\/]([0-9]+)/;
     $BGP32BITSTRING = bgp2bitstring($1,$2);
     $TEMP = $BGP32BITSTRING;
     # Adding new entry in BGPHASH
     $BGPHASH{$BGP32BITSTRING}= $MYLIST[0];
     };
  # End if
}; # End while
# End of loop over "post-phase 1" file
close(BGPFILE);

#########################################################################
# Second part: taking each line of "post-phase 1" file, doing our     #
# best-matching entry  algorithm in binary mode and placing the line #
# in which network prefix has been routed into OUTPUT_FILE.           #
#########################################################################

$POST_PHASE_1_FILE = $ARGV[1];
$OUTPUT_FILE = $ARGV[2];
open (IN,"<$POST_PHASE_1_FILE");
open (OUT,">$OUTPUT_FILE");
while (defined ($CURRENTLINE = <IN>)){
chop($CURRENTLINE);
@MYLIST = split (/[\s]+/,$CURRENTLINE);
print OUT "$MYLIST[0]";
if (@MYLIST > 1){
foreach $COUNT (1..@MYLIST-1){
$MYLIST[$COUNT]=~/([0-9]+\.[0-9]+\.[0-9]+\.[0-9]+)[\/]([0-9]+)/;
# Transforms decimal network prefix into "binary mode" prefix
$PREFIX = bgp2bitstring($1,$2);
# Finding best-matching entry in BGP table
$ENTRY = bestmatch($PREFIX);
# Converting "binary mode" best-matching entry into decimal prefix
$ROUTEDPREFIX = $BGPHASH{$ENTRY};
print OUT " $ROUTEDPREFIX";
};
print OUT "\n";
}
else {print OUT "\n";};
$CURRENTLINE = <IN>;
print OUT "$CURRENTLINE";
};
close(IN);
close(OUT);

#####################
# subroutines part #
#####################

#########################################################################
# converts a < bgpprefix,bgpmasklength > pair into a 32 bit bitstring   #
#########################################################################

sub bgp2bitstring {
    my $prefix = $_[0];
    my $mask = $_[1];
    $bitstring;# the result
    my @LIST = split(/\./,$prefix);
```

```
if (defined($LIST[1])){if (defined($LIST[2])){if (defined($LIST[3])){# mask is >
24 bits long
$prefix =
substr(dec2bin($LIST[0]),24,8).substr(dec2bin($LIST[1]),24,8).substr(dec2bin($LIST
[2]),24,8);}
else {# mask is > 16 bits long
$prefix =
substr(dec2bin($LIST[0]),24,8).substr(dec2bin($LIST[1]),24,8).substr(dec2bin($LIST
[2]),24,8);}}
else {# mask is > 8 bits long
$prefix = substr(dec2bin($LIST[0]),24,8).substr(dec2bin($LIST[1]),24,8);}}
else {# mask is <= 8 bits long
$prefix = substr(dec2bin($LIST[0]),24,8);};
# mask is int so we only need to take the first $mask bits of prefix to be done
$RESULT = "";
$RESULT = substr($prefix,0,$mask);
return $RESULT;
}# end bgp2bitstring

##############################################################################
# Finds the longest matching entry in BGP hash table for an 32-bit IP        #
# address and returns the "binary mode" entry.                               #
##############################################################################

sub bestmatch {
    my $ip = $_[0];
    $bestlength = 0;
    $bestmatch = "";
foreach $PREFIX (keys %BGPHASH){
$TEMP = 0;
foreach $COUNTER (0..31){
if (defined(substr($PREFIX,$COUNTER,1))){
if ((substr($ip,$COUNTER,1)) eq (substr($PREFIX,$COUNTER,1))){
$TEMP = $TEMP +1;}
else {last;}
}
else {last;}
};# end foreach $COUNTER
if ($TEMP > $bestlength){$bestlength = $TEMP;
                         $bestmatch = $PREFIX;
};
};# end foreach $PREFIX
if ($bestlength > 0) {return $bestmatch;}
else {return "";}
}# end bestmatch

##########################################
# Converts an integer to a 32-bit string #
##########################################

sub dec2bin {
    return unpack("B32",pack("N", shift));
}# end dec2bin
```

Figure A5.6: BGP table lookup code

The code presented in Figure A5.6 does only BGP prefixes routing. If network prefixes need be replaced by ASs numbers or inter-AS links, the only thing one has to do is define a new hash to bind a BGP entry with its AS number or the AS path vector and replace the routed prefix by the desired BGP object. We did also write code to transform the BGP path vectors of the BGP table into a sink-tree topology. Such a scheme would enable to work on the inter-AS graph rooted at the local AS, for example to evaluate the differences in variability that occur when balancing the traffic among several upstream AS paths but this is out of the scope of this dissertation.

A5.3.3 Phase 3

This phase was the one for which the biggest number of scripts exists. What we call DAT format may be viewed as a two-column matrix. The first column gives the timestamp while the second (and the others) the value of the considered "concept(s)" corresponding to the timestamp. The "concept" could represent the total incoming traffic, the traffic of a particular prefix, the number of prefixes or ASs considered as active during the timestamp... The task performed essentially relates to summarization and extracting subsets of the "post-phase 2" files. These DAT files were used to plot almost all figures presented in Chapter 5.

Appendix 6

Interdomain Routing in the Internet

A6.1 Introduction

The Internet is a set of independent routing domains (or ASs). Each AS is under a specific administrative authority. ASs run Interior Gateway Protocols (IGPs) such as RIP, OSPF, and IS-IS within their boundaries and interconnect through an Exterior Gateway Protocol (EGP) called the Border Gateway Protocol. EGPs were introduced because IGPs do not scale in networks larger than the size of the enterprise. IGPs were not designed to cope with global Internet routing. They do not have the features to segregate enterprises into different administrations that are technically and politically independent.

The purpose of this Appendix is to provide the reader an introduction about the way routing happens to function today at the interdomain level. For the reader further interested in interdomain routing, reference [IRA97] presents a realistic view of the way routing happens to work in the Internet today.

A6.2 Dividing the Internet

EGP routing protocols were created to limit the expansion of routing tables and to provide a more hierarchical structure to the Internet by partitioning its routing domains into separate administrations, i.e. Autonomous Systems that all have their own routing policies. The reason for which interdomain routing protocols are different from intradomain ones resides in the assumption that routing domains do not want to be combined into one network basket. Routing domains are independently funded so that resource consumption within one routing domain does not follow the same billing and priority policies as one of another routing domain. In addition, routing domains do not trust one another. They do not trust the routing algorithms running on the other domains because bugs into these routing protocols should not affect the former domain routing and because administrative rules might have to restrict routes advertised by other routing domains.

The currently used EGP is BGP4, the de facto standard for Internet routing. BGP is an exterior protocol that provides a controlled and as well as loop-free topology.

A6.2.1 Routing Concepts

Static routing means that information is manually configured into routers. The static routes remain in the routing table even if the destinations were down, and traffic would be send toward the destinations. Default routing means that traffic to destinations that are unknown to the router will be sent to a default interface. This kind of routing is the easiest for a routing domain connected to only one single exit point. Dynamic routing means that routes are learned via a routing protocol. The network reachability information is thus dependent on the existence and state of the destination network. If a destination goes down, the route will disappear from the routing table and traffic will not be sent toward that destination. Do not think that dynamic routing is always more efficient: static and default routing also have advantages. Static routing can bypass transient states that appear with dynamic

routing and limits the risks for bad routing information injected by others routing domains. Default routing for its part significantly reduces the routing table size.

A6.2.2 Autonomous Systems

An Autonomous System (AS) is a set of routers having a single routing policy and running under a single technical administration. The AS is viewed as a single entity to the outside world. Each AS has an identifying number, which is assigned by an Internet Registry or a provider. Routing information is exchanged between ASs via an exterior gateway protocol.

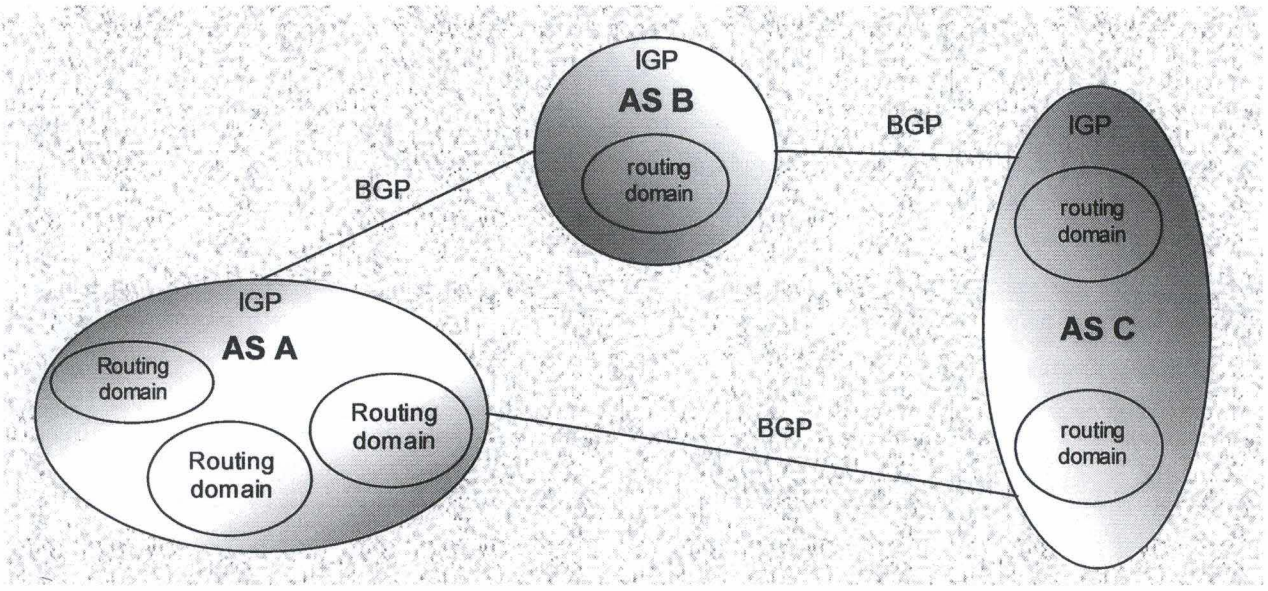

Figure A6.1: Example of AS relationship

The partitioning of the Internet into administrative domains means capability to manage one large network. The splitting into ASs allows each AS to run its own set of IGPs independently one from the others.

ASs can be classified in three types: "stub", "multi-homed non-transit" and "multi-homed transit". A *stub* AS reaches networks outside its domain via a single exit point. A *stub* AS does not need to learn routes from its provider because all non-local traffic passes through the single exit point. A *multi-homed* AS has more than one exit point to the outside world. *Transit* traffic is any traffic that has both source and destination outside the AS. A *non-transit* AS does not allow transit traffic to go through it. A *non-transit* AS would only advertise its own routes and not the routes it learned from other ASs. This ensures that traffic for any destination that does not belong to the AS would not be directed to the AS. A *multi-homed transit* AS has more than one connection to the outside world and can be used for transit traffic by other ASs.

Note that even if BGP is an exterior gateway protocol, it may be used inside an AS to exchange BGP updates. BGP connections inside an AS are called Internal BGP (IBGP) while BGP connections between ASs are called External BGP (EBGP). Routers that are running IBGP are called *transit routers* when they carry the transit traffic going through the AS. Routers that run EBGP are called *border routers*.

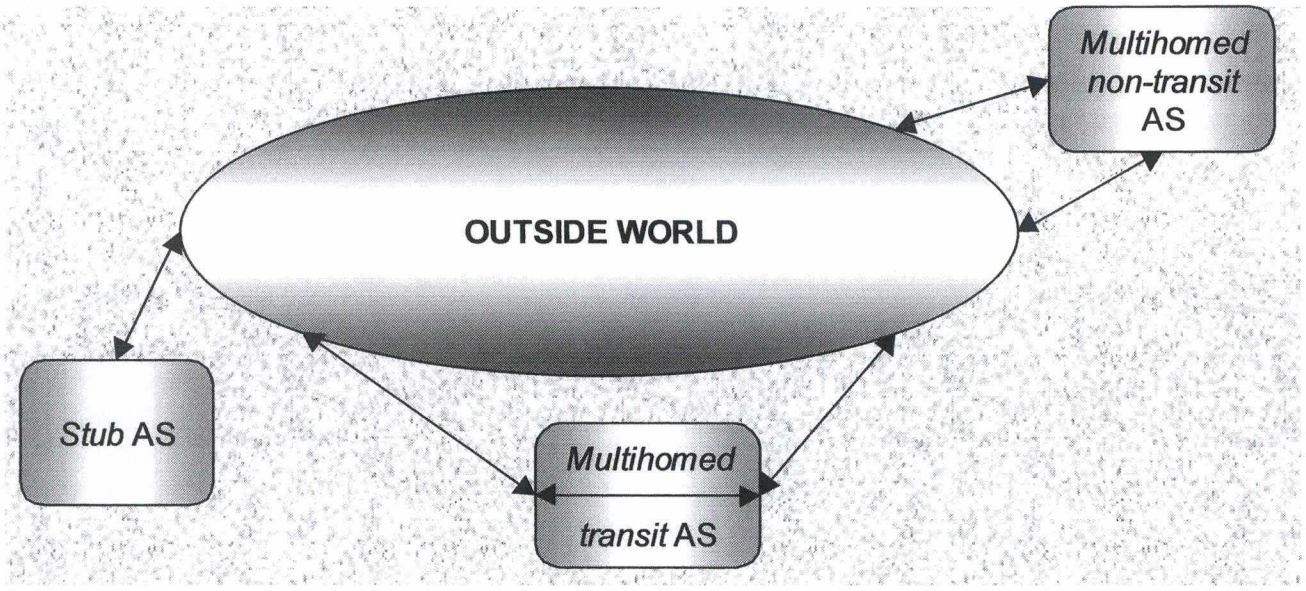

Figure A6.2: ASs types

A6.3 BGP

BGP stands for Border Gateway Protocol. The current version is BGP4 (see [BGP95]), which deployment started in 1993. BGP assumes that routing within an AS is done through an IGP. BGP constructs an "AS-oriented" graph topology of the Internet based on the information exchanged with other ASs. Connections between two ASs form a path; a route between two ASs is made of a list of ASs numbers. BGP ensures a loop-free interdomain topology.

BGP is a path vector protocol used to carry routing information between ASs. The *path vector* term comes from the fact that a BGP route is a list of AS numbers indicating the path a route traverses. In order for BGP to exchange routing information, TCP is used so that transport reliability is ensured and that BGP does not have to care about it by itself.

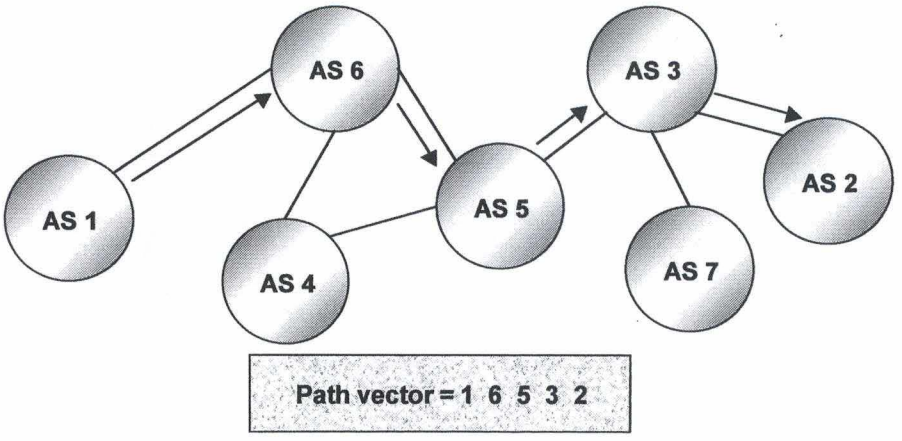

Figure A6.3: BGP path vector

A6.3.1 Peering Connection

Two BGP routers that want to exchange routing information form a transport protocol connection between each other. Such routers are called *neighbors* or *peers*. Peer routers exchange multiple messages to initiate the connection and confirm some parameters like protocol version number. If any disagreement happens between the peers, notification errors are sent and the peer connection is not established.

After peering connection establishment, all candidate BGP routes are exchanged. Every time routing information changes[39], incremental updates are sent to the peers. This approach reduces the CPU overhead and bandwidth use compared with complete periodic updates.

Routes are advertised between BGP peers in UPDATE messages. An UPDATE messages contains a list of *<prefix,masklength>* pairs indicating the list of destinations reachable through each system. The UPDATE message also contains the path attributes, which includes information such as the degree of preference of a particular route. When routing information changes, such as a route being unreachable (say *withdrawn* in the BGP context) or having a better path, the BGP peer informs its neighbors by withdrawing the invalid routes and injecting new routing information. If no routing information changes, the BGP peers only exchange KEEPALIVE messages. KEEPALIVE messages are sent periodically to ensure that the connection is kept alive.

BGP keeps a table version number to keep track of the instance of the BGP routing table. Whenever the table changes, its version gets incremented.

A6.3.2 Routing Information Exchange

Routing updates contain all the necessary information that BGP uses to construct a loop-free topology of the Internet. An UPDATE message contains the following basic blocks:

- Network Layer Reachability Information (NRLI);
- Path attributes;
- Unreachable routes.

The NLRI indicates the networks being advertised, in the form of an IP prefix route. The path attribute list provides the information to detect routing loops and to enforce local and global routing policies. The unreachable routes part of the message is a list of the routes that have become unreachable.

The NRLI is the mechanism by which BGP supports classless routing. An IP prefix is an IP address with an indication of the number of bits that constitute the network number. The NRLI part of the BGP routing update lists the set of destinations about which BGP is trying to inform its peers.

Withdrawn routes are advertised the same way as reachable routes: an *<IP prefix,lengthmask>* pair indicates all routes that have to be withdrawn from the routing table. The lengthmask permits to determine all routes having at least *lengthmask* bits in common with the *IP prefix*. All entries in the BGP table matching the specified prefix need to be removed from the BGP table.

[39] Do not confuse this with periodic refreshes, only actual changes are notified.

The path attributes are a set of parameters meant for keeping track of route-specific information such as path information, degree of preference of a route, next hop value of a route and aggregation information. These parameters are used in the route filtering and decision process (as described in the next section).

A6.3.3 Advanced BGP Capabilities

We have seen so far the basics of BGP: how it works in a very general manner. This section looks at more practical capabilities in terms of routing design.

A6.3.3.1 Building Peer Sessions

Even if BGP is due to serve between ASs, it can be used within an AS between border routers running external BGP. A neighbor connection, also called a *peer connection*, may be established between two routers within the same AS in which case BGP is called *internal* BGP (IBGP). A peer connection may be established between two routers in different ASs, BGP is then called external BGP (EBGP).

External BGP neighbors have a restriction on being physically connected because BGP drops any updates from its external peer if the peer is "non-connected". In practice however, neighbors cannot always be on the same physical segment. Such neighbors are thus logically connected, but not physically. Note that this restriction may be overridden through some extra configuration.

To avoid creating routing loops inside the AS, BGP does not advertise to internal BGP peers routes that are not learned via other IBGP peers. It is therefore important to maintain a full IBGP mesh within the AS. This requirement does not scale well with big ASs, so that a solution to this problem can be found with "route reflectors" (see [BGPRR]).

A6.3.3.2 Updates Sources

While routing stability is a big issue in today's Internet, there is a close correspondence between:

- Route fluctuations and the stability of the Internet access links and
- How routing information was injected into the Internet via BGP.

Routing information may be injected into BGP dynamically or statically. Dynamically injected routes come and go from the BGP routing table depending on the status of the networks they identify. Statically injected routes are permanently maintained in the BGP routing table.

Dynamically injected routing information is done by enabling IGP routes to be redistributed into BGP. Some levels exist within the "dynamic" aspect because it is possible to specify a subset of the prefixes that should benefit from dynamic injected routing. This method has the advantage of enabling to limit the number of routes that are present in the BGP routing table (for example because the number of prefixes that have to be advertised might be higher than the number of entries the router can tolerate).

To statically inject routing information into BGP, IGP routes that need to be advertised to other peers are manually defined as static routes. This ensures that these routes will never disappear from the routing table and thus will always be advertised.

A6.3.3.4 The Routing Process

BGP is a simple and flexible protocol. Routes are exchanged between peers via UPDATE messages. BGP routers receive the UPDATE messages, run policies over the updates and advertise the routes to other BGP peers. When several routes exist for a particular destination, BGP does not advertise all known routes but it chooses the best one and sends it. A BGP router not only advertises routes learned from other BGP peers but it also advertises networks that belong to its own AS. Valid local routes originated in the system and the best routes learned from BGP peers are installed in the IP routing table. The IP routing table constitutes the information on which the final routing decision is made.

The BGP process can be modeled as follows:

1. A pool of routes is received by the router from its peers,
2. An Input Policy Engine filters the routes or manipulates their attributes,
3. A decision process decides which routes the router will use,
4. An Output Policy Engine filters the routes or manipulates their attributes,
5. A pool of routes is advertised by the router to its peers.

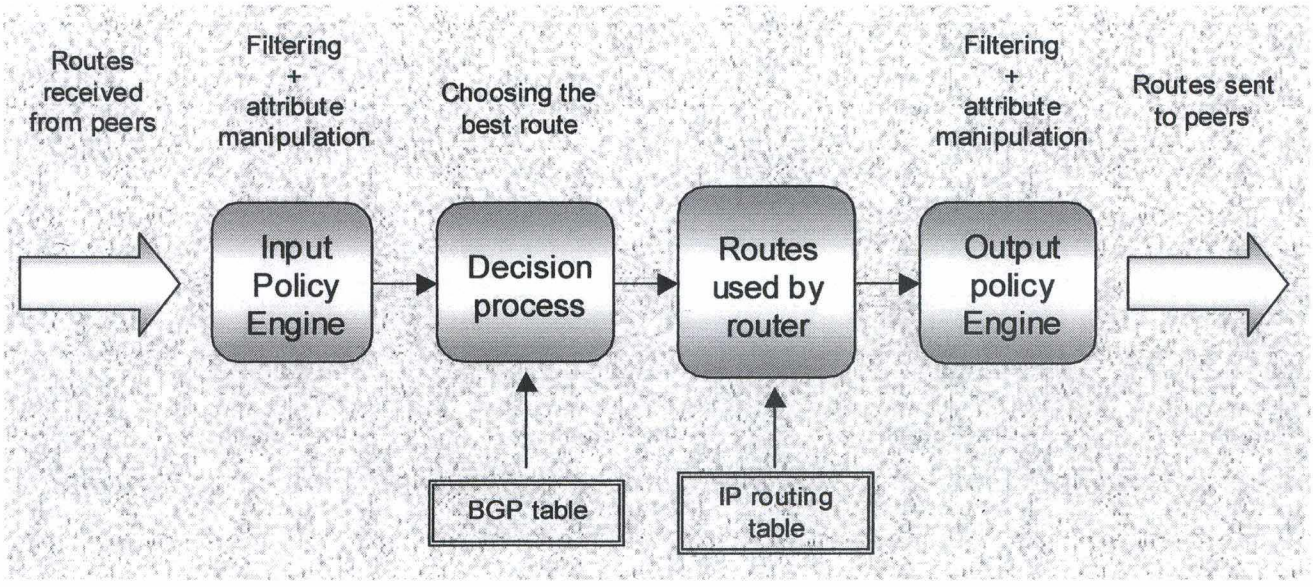

Figure A6.4: BGP Routing Process

A6.3.3.5 Controlling BGP Routes

The BGP attributes are a set of parameters that describe the characteristics of a prefix route. The BGP decision process uses these attributes to select its best routes. These attributes can be manipulated by a BGP router in order to affect the routing behavior.

The AS_path attribute is a sequence of AS numbers a route traverses to reach the destination. The AS that originates the route adds its AS number when sending the route to its external peers. Each AS receiving the route prepends its own AS number to the list and passes it on to other BGP peers. *Prepending* is the act of adding the AS number to the AS_path list at its end. BGP uses the AS_path attribute to ensure a loop-free topology in the Internet. If the route is advertised to the AS that originated it, that AS will see itself as part of the AS_path attribute and will not accept the route. BGP speakers prepend their AS number when advertising the route to other ASs (external peers) but when a route is passed to a BGP speaker within the same AS, the AS_path information is left intact. Because

BGP prefers a shorter path to a longer one, it is possible to include dummy (redundant) AS path numbers to increase path length and hence influence the traffic trajectory.

The local preference attribute is a degree of preference given to a route to compare it with other routes for the same destination. A higher local preference indicates that the route is more preferred. Local preference is local to the AS and is exchanged between IBGP peers only; it is not passed to EBGP peers. An AS connected via BGP to other ASs will get updates about the same destination from different ASs. Local preference is then used to set the exit point of an AS to reach a certain destination. Because this attribute is communicated within all BGP routers inside the AS, all BGP routers will have a common view on how to exit the AS.

A6.3.3.6 Route Filtering and Attribute Manipulation

A BGP speaker can choose what routes to send and what routes to receive from any of its BGP peers. Route filtering is essential in defining routing policies. An AS can identify the inbound traffic it is willing to accept from other neighbors by specifying the list of routes it advertises to its neighbors. Conversely, an AS can control what routes its outbound traffic uses by specifying the routes it accepts from its neighbors. Filtering is also used on the protocol level to limit routing updates flowing from one protocol to one another.

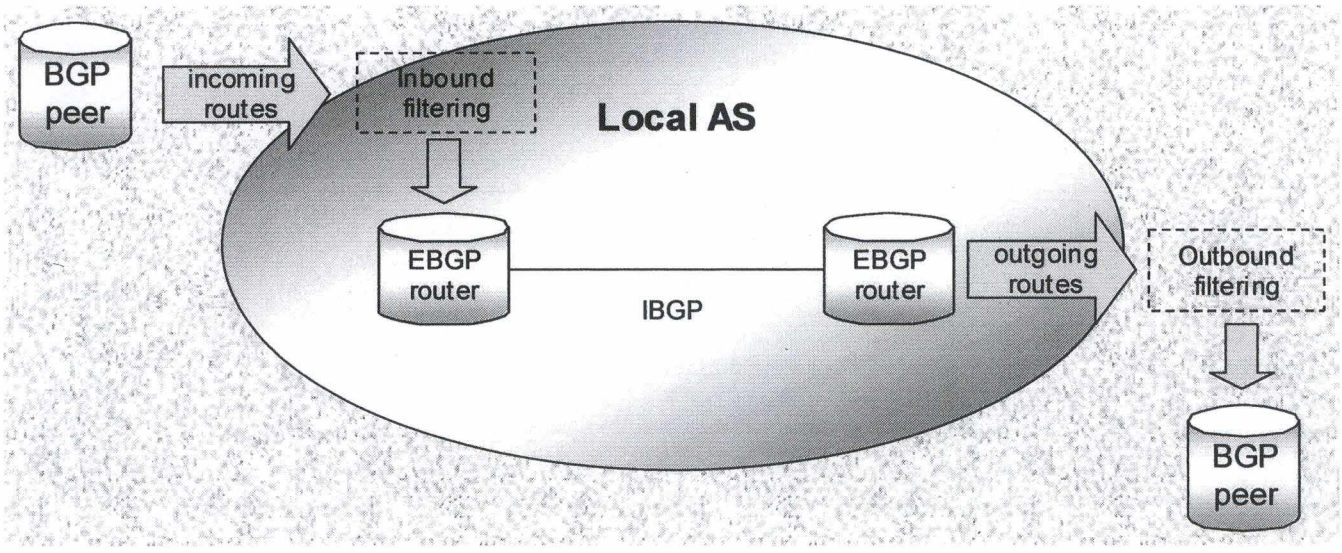

Figure A6.5: Route filtering mechanism

A6.4 Important Routing Features

ISPs and corporate networks connected to ISPs require adequate control over their internal traffic flow. Redundancy, symmetry and load balancing are crucial issues facing anyone implementing high-throughput connections to the Internet.

Redundancy is achieved by providing multiple alternate paths for the traffic, usually by having multiple connections to one or more ASs. *Symmetry* means having traffic that leaves the AS from a certain exit point return through the same point. *Load balancing* is the capability to split the traffic over multiple links.

A6.4.1 Redundancy

Although corporations and providers would like to achieve in-interrupted connectivity, connectivity problems sometimes occur. A router's connection to the Internet involves the router, cabling, administrators... At any time, the connectivity may be jeopardized by human error, software error, physical error... For all these reasons, redundancy is desirable. Because redundancy and symmetry may constitute conflicting design goals, finding the correct balance between them is critical. The more redundancy a network has, the more unpredictable the traffic entrance and exit points would be.

In addition to the reliability motivations, geographical constraints often are part of the game. Many contemporary companies are national, international or multinational in nature. The AS is a logical entity than spans different physical locations. A corporation with an AS that spans several geographical points may take service from a single provider or from different providers in different regions.

Because redundancy refers to the existence of alternate routes to and from a network, this means that additional routing information needs to be kept in the routing tables. To avoid the extra routing overhead, default routing becomes an alternate practical tool, in order to provide us with backup routes when the primary one fails.

A6.4.2 Symmetry

Symmetry is easy to achieve if a single entrance and exit point exists. However, due to redundancy and the presence of multiple connections, traffic tends to be asymmetrical. Asymmetrical traffic is not that bad, it may be acceptable in some situations depending on the overall physical topology. To accommodate symmetry, a primary link should be chosen and a best-effort one should be made to enable the majority of traffic to flow on this link. Redundancy might be accommodated by enabling other links to be backup links that will be used if the primary link fails.

A6.4.3 Load Balancing

Load balancing deals with the capability of splitting the traffic over multiple connections. A common misconception about balancing is that it means an equal distribution of the load. Perfectly equal distribution of traffic is elusive even in situations where traffic flows in a network under a single administration. Load balancing tries to achieve a traffic distribution pattern that will best utilize the multiple redundant links. Achieving it requires a quite good understanding of what traffic you are trying to balance: incoming or outgoing.

The pattern of inbound and outbound traffic go hand in hand with the way you advertise your routes and the way you learn routes from other ASs. Inbound traffic is affected by how the AS advertises the networks that are reachable via itself to the outside. Outbound traffic is affected by the routing updates coming from outside ASs.

A6.4.4 Policy Routing

Policy routing is a means of controlling routes that relies on the source, or source and destination, or traffic rather than destination alone. It can be used to control traffic inside an AS as well as between ASs. Policy routing is used when you want to force a routing behavior different from what the dynamic routing protocols dictate. Static routing enables to direct traffic based on the traffic destination. Policy

routing, on the other hand, enables you to direct traffic based on traffic source or a combination of source and destination.

One practical application of policy routing is its use in *firewalls*. A firewall is a device that applies security requirements to traffic. Depending on the network setup, administrators might want to direct some or all traffic toward a firewall device. Policy routing may be configured on a router bordering external networks to force incoming traffic to be directed to the firewall. After the firewall applies policies and encryption, traffic will be sent to its destination.

Policy routing should not replace dynamic routing but should instead complement it. Policy routing has several important drawbacks:

- Extra configuration is needed to identify traffic sources or a combination of traffic source and destination. Other traffic should not be disrupted.
- Policy routing is CPU-intensive because it requires some extensive packet header parsing.