This is to certify that the

dissertation entitled

VIRTUAL NETWORK PORTS

presented by

Vibhavasu Vuppala

has been accepted towards fulfillment
of the requirements for

Ph.D. degree in Computer Science

Lionel M. Ni
Major professor

Date 22 August 2002

MSU is an Affirmative Action/Equal Opportunity Institution O-12771

VIRTUAL NETWORK PORTS

By

Vibhavasu Vuppala

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Computer Science and Engineering

2002

ABSTRACT

VIRTUAL NETWORK PORTS

By

Vibhavasu Vuppala

The sudden growth of Internet has exposed some of the problems with its architecture: forwarding performance, traffic engineering, and quality of service. We address these issues through the concept of a Virtual Network Port (VNP), which enables bypassing of the network layer, identifying traffic flows, and providing explicit routes. A VNP is an abstraction for a set of remote network interfaces. We have applied this simple concept to implement layer-3 switching. A VNP provides a conduit below layer-3 to remote nodes and networks. Traffic through a VNP can be shaped, policed, encrypted etc.; a VNP's control procedure provides for generic packet processing. VNP paths are setup implicitly by routing protocols, or explicitly based on QoS constraints. VNP domains are arranged in a hierarchy to provide scalability at the core of an internetwork. We compare VNP with Multi-Protocol Label Switching (MPLS) quantitatively using simulation. Packets through a VNP across multiple networks do not require fragmentation. VNP framework is conceptually simpler, has better semantics, and is more efficient than MPLS. VNP has applications in other domains as well; we describe the design of a scalable IP router using VNP.

To Advaith

ACKNOWLEDGMENTS

Words can never describe my sense of gratitude for Prof. Lionel M Ni. He helped me when I was down, depressed, and discouraged. He believed in me when even I had lost faith in myself. He was always there to guide me on technical and non-technical issues. He is the best teacher I have ever had, and perhaps the best person I have ever known. These few sentences are not good enough for what I want to say, but then nothing in words would ever be.

I thank Dr. Matt Mutka and Dr. Richard Enbody for their guidance and support throughout the academic program. I thank Dr. John Eulenberg for his inspirational work. I am grateful to Dr. Anil Jain, Dr. Abdol Esfahanian, and Dr. Betty Cheng for their support. I thank Hong Xu for suggesting the seminal idea of scalable IP routers.

I thank Quantech Global Services for supporting me financially, and providing me with a flexible work environment without which this work would not have been possible. I am very grateful to Dr. Mukesh Gandhi and Mala Gandhi for supporting my family through the ebbs and flows of life. I thank my friends, Nateswara Rao Dhavala, Ramakrishna Dhulipala, Amit Midha, Geetha Ramamurthy, Raja Sunderraman, Giri Tiruvuri, Srinivas Tumbalam, and especially Satya Kudapa, for their support.

I feel blessed to have had the love and support of my parents (Smt Parvati and Prof Ramamurthy Vuppala) and my sister (Smt Adurthy Aparna) throughout this arduous journey. I am much thankful to my brother-in-law, Adurthy Phani Kumar, for his help and support. Finally, I am very grateful to my wife, Radhika, for all that she has done for me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

The Internet was conceived as an experimental network in the late 1960s by the United States Department of Defense's Advanced Research Projects Agency (DARPA). It started as a research and education tool with four nodes connected by 56 kilo-bits-per-second lines. During the next two decades the Internet grew steadily, inter-connecting hundreds of thousands of nodes in more than 25 countries. The commercialization of the Internet and the introduction of the World Wide Web (WWW) in the early 1990s resulted in an exponential growth, in its traffic, by the mid-1990s. The current Internet links more than 150,000 networks comprising of more than 40 million nodes spanning more than 170 countries. The current backbone has links with bandwidths of up to tens of giga-bits-per-second (OC-192c and above).

Figure 1-1 illustrates the growth in Internet's infrastructure in the recent years. Every aspect of its infrastructure, hosts, networks, domains, and web servers is increasing at a rapid rate. Bandwidth-intensive WWW has taken over File Transfer Protocol (FTP) as the dominant Internet application. For the first time in history data traffic has surpassed the voice traffic. The communications industry has been successful in meeting the perpetually increasing demand for bandwidth; technologies like Asynchronous

Transfer Mode (ATM), Synchronous Optical NETworks (SONET), and Dense Wave Division Multiplexing (DWDM) are making gigabit networks feasible. As a result, the traffic on the Internet is increasing at an exponential rate; it is doubling every six to nine months. This sudden and unexpected growth of the Internet during the end of the century has exposed a number of limitations with its architecture and infrastructure. This work proposes concepts that help in improving the solutions for some of these issues.

1.1 Internet's Architecture[1]

Internetworking is a method of connecting networks, irrespective of the technologies used by the individual networks. For example, internetworking allows hosts on an HIPPI Local Area Network (LAN) to communicate with hosts on an IEEE 802.6 LAN. The TCP/IP protocol suite is the internetworking protocol used on the Internet. Its architecture has the following basic layers:

1. Physical Layer
2. Data Link Layer
3. Network Layer (IP, ICMP)
4. Transport Layer (TCP, UDP)
5. Application Layer

The heart of this protocol suite is the Internet Protocol (IP). IP offers *best effort* delivery of datagrams among the hosts on an internetwork. It makes a reasonable effort to deliver the datagram uncorrupted to the correct destination. It may drop the datagram, corrupt it,

[1] A more detailed understanding of the Internet architecture is available in [27][17].

deliver it to the wrong destination, or it may deliver it out-of-order. The objectives are to minimize packet loss, maximize throughput, and minimize delay.

Each host on an IP internetwork has at least one IP address; it may have more than one IP address if it is connected to multiple networks. To direct a datagram to another host, the sender sends the datagram to one of the recipient's IP addresses. IP

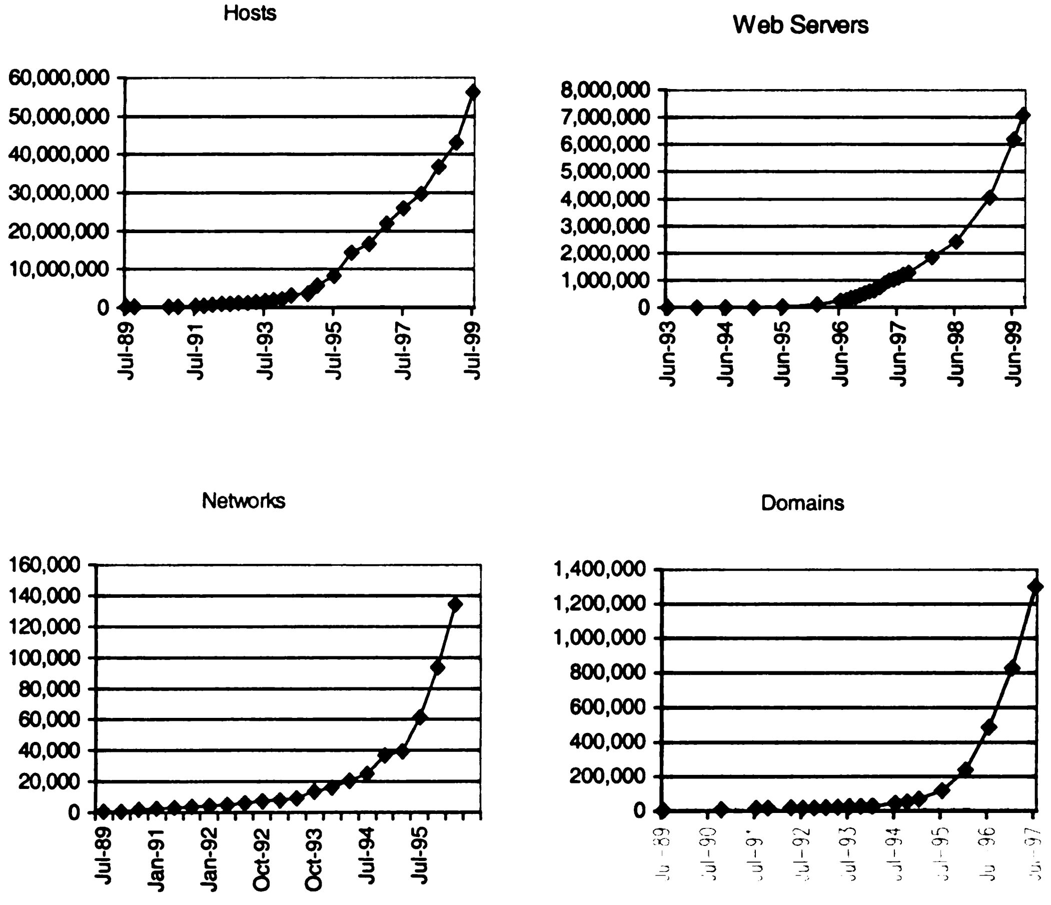


Figure 1-1 Internet Growths

addresses are 32 bits long[2]; they are generally written as four decimal numbers, one per byte, separated by dots (eg 192.168.3.6). A part of this address identifies an individual network or a group of them, whereas the complete address identifies a host on that network.

For each physical network technology, there is a standard way to encapsulate IP datagrams into its framing protocol, and map the IP address into the technology's addressing scheme. For example, *Request for Comments* (RFC) 894 defines how to carry IP datagrams in Ethernet frames; RFC 826 defines a protocol to translate IP addresses into Ethernet addresses.

Individual networks on the Internet are connected using IP *routers*. The routers' job is to forward IP datagrams among the networks. It is a device that accepts an IP datagram from one network, determines the next network the datagram needs to traverse to get to its destination, and places the datagram on that network using the next network's encapsulation.

The Internet is divided into a set of disjoint *Autonomous Systems* (AS). An autonomous system is a collection of routers, networks, and hosts under a single technical administration. Routers inside an autonomous system run *interior gateway protocol* (IGP) routing protocols. Examples of some popular IGPs are: OSPF, ISIS, and RIP. Routing information among autonomous systems is exchanged using *exterior gateway protocols* (EGP). The standard EGP on the current Internet is *Border Gateway Protocol* (BGP).

[2] This work deals with IP version 4 only.

Routing is done hierarchically based on the IP address. Routers keep track of the network topology by exchanging topology information using routing protocols. Starting at the source, a packet hops through a number (possibly none) of routers before reaching the destination. At each *hop*, the router consults its *forwarding base* to choose the next hop for the packet. Each router makes this decision independently based on its analysis of the packet's network layer header. A router also uses the routing protocols to populate its forwarding base.

1.2 Motivation

1.2.1 Forwarding Performance

If a router is unable to handle the incoming traffic volume, it simply drops the excess packets. This results in packet retransmissions, wasted network bandwidth, and application slow-down. Hence, it is important that the forwarding performance of routers scale with the increase in traffic. However, the Internet traffic is increasing at a rate well above the growth-rate, as per Moore's Law, of the processing elements in a router. Therefore, changes in the traditional router architecture, forwarding algorithm, or forwarding paradigm are needed.

1.2.2 Traffic Engineering

With this growth and popularity of Internet, the Internet Service Providers (ISPs) feel the need for maximizing the utilization of their networks. They want to deploy service differentiation to maximize their revenues; they would like to offer higher quality service at a premium. The current ISP networks do not support such capabilities. This is due to inability of conventional IP technologies to support *Traffic Engineering* (TE). Traffic engineering covers a diverse set of network performance optimization issues: guaranteed quality of service (QoS), improving the utilization of network resources by spreading the traffic evenly over the network, and providing features for quick recovery in cases of node or link failures [23]. Traffic engineering is an essential component in the architecture for providing end-to-end QoS guarantees.

Current IGPs are topology driven. Each router keeps track of the overall routing area state database (in the case of link-state routing protocols). It makes independent routing decisions based the shortest path between two nodes computed using a simple metric that has been manually assigned to each link. This computation does not take into account the current load on each link. This results in poor resource allocation. Even though multiple paths may exist between two nodes the packets always follow a single path, the current shortest path. This results in certain parts of the network to become congested, and some others to remain under-utilized at the same time. It is not possible to forward packets with the same destination through different paths based on constraints like priority, fee, or quota.

Traffic engineering is concerned with optimizing the utilization of network resources. It helps network administrators give precise control over the flow of traffic within the network. They can balance the traffic load among the paths and avoid congestion on the network. With the help of TE, ISPs can offer categorized services to their customers. They can provide faster or more reliable paths for customers who are willing to pay more.

1.3 Solutions

During early 1997, when this work started, various solutions for forwarding performance problems were being proposed. These solutions can be divided into three categories: High Speed Routers (HSR), Fast Lookup Techniques (FLT), and Switching Techniques (ST). HSRs use multiple forwarding engines, interconnected by high-bandwidth switching fabric, to forward packets in parallel. FLTs propose new longest prefix match algorithms that improve lookup time into the IP forwarding table. STs forward the packets below Layer-3 by laying data-link switched paths. They use very simple direct lookups for forwarding packets. IETF's Multi-Protocol Label Switching (MPLS) working group has been working towards a standard for switching technique.

HSRs use proprietary hardware, and are expensive. FLTs do not use simple lookups like STs do. However, by 1999, HSRs and FLTs were making forwarding throughput of giga-bits-per-second possible [41][50]; and MPLS was being successfully applied towards addressing traffic-engineering issues [7][67].

1.4 Virtual Network Ports

This work falls in the category of switching techniques. It addresses the aforementioned issues through the concept of Virtual Network Ports (VNP). The idea for VNP originated in the design of a scalable cost-effective router [59]. The concept was later extended to lay switched paths, and to identify flows [60]. It eliminates some of the drawbacks of MPLS: stack operations, fragmentation, and passive labels. The VNP concept has been implemented in a prototype using PCs running Linux.

When this work started, the focus was on the problem of providing scalable forwarding performance. The solution was to design a router with multiple forwarding engines working in parallel. While all the router architectures used proprietary hardware, it was decided to use off-the-shelf commodity components to keep the cost low, and the architecture open. Heterogeneous off-the-shelf systems were used as forwarding engines and protocol processors. High throughput System Area Network (SAN) and Local Area Network (LAN) technologies like Myrinet and ServerNet were available in the market. They were used as the inter-connection or the switching back plane of the router. The scalable router was implemented during the spring of 1997. The concept of Virtual Network Port was used in this implementation to cohere the various forwarding engines as one router. The VNP concept was later extended to enable sub Layer-3 switched paths across routers. It now fell in the category of the switching techniques. It has all the benefits of switching techniques, but is conceptually better and more efficient than MPLS.

1.5 Organization

Chapter 2 introduces the concept of VNP. It describes how VNPs work, and how they are combined to construct switched paths across networks. Chapter 3 describes how VNPs are setup in a large internetwork. It Chapter 4 presents the implementation details of a VNP. Chapter 5 describes the design of a scalable router using VNP. Chapter 6 compares our work with MPLS. Chapter 7 presents the concluding remarks.

Chapter 2

VIRTUAL NETWORK PORTS

The concept of Virtual Network Port (VNP) forms the motif of this work. It originated in the design of a scalable IP router (see Chapter 6). It was necessitated by the desire to avoid processing at layer-3 –IP in this case– in a cluster of forwarding engines. Initially VNPs were restricted in their usage. They could not cluster nodes across physical networks; the nodes involved in VNP had to be on the same physical network. The concept was later expanded to cover nodes across networks and domains.

2.1 Network Port

A *network port* is a communication device in a computer system that can be used to transmit and receive data; it is a data structure in a network subsystem that is used to control transmission activities [31]. It is also called a *network device*, or a *network interface*. There can be three types of network ports on a node: hardware, software, and loopback.

A hardware network port is a physical device that attaches to a network. It is housed in a hardware circuit board called network interface card (NIC) or line-card. A software network port does not have an associated hardware device; it exists only as a software module in the networking subsystem. It is mostly used to provide data services, like encryption and tunneling, to upper layers while presenting them with the same abstraction as a hardware port. A loopback port is a special instance of a software network port. It is a network connection to the node itself. Packets sent on the loopback port do not go out to any network; they loop back to the next layer (layer-3) on the same node.

2.2 Virtual Ports

A *Virtual Network* P*ort*[3] is an abstraction for network port on a remote node. It is an interface to a remote network. It makes a remote network appear directly connected to the local node. A VNP is very similar to a hardware network interface in properties except for the following:

- There is no hardware interface card for a VNP.
- It is unidirectional; packets always flow out of this device but they never come in.
- It does not have any physical or Media Access Control (MAC) address associated with it.
- It is not involved in resolution protocols like Address Resolution Protocol (ARP)

[3] The reason for not naming it Virtual Network Interface is that the term Virtual Interface has been used in literature to denote something different; a Virtual Interface provides a user-space abstraction of a (kernel-space) network interface. The term *Remote Network Port* may sound more appropriate to denote the concept; we had actually used the term during the initial phases of this work. However, we found the term *virtual network port* to be more suitable as the concept had expanded to cover more than just remote ports.

Figure 2-1 shows two nodes N_1 and N_2 connected to the same physical network. The two nodes are connected directly at the data link layer (Layer-2) through ports N_1P_1 and N_2P_1. Node N_2 has two physical network ports N_2P_1 and N_2P_2. Node N_1 has a physical port N_1P_1 and a virtual one N_1P_2. The virtual port N_1P_2 is an abstraction of the remote physical port N_2P_2. Packets transmitted through N_1P_2 actually come out through N_2P_2.

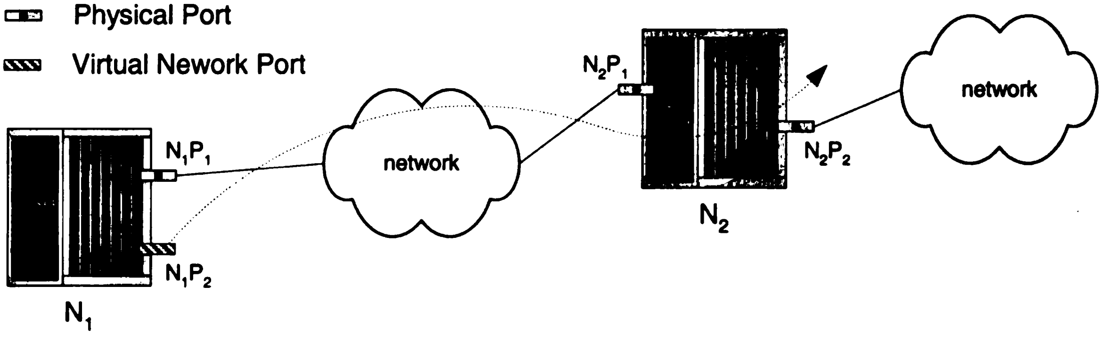


Figure 2-1 Virtual Network Port

2.3 Access Type

A *point-to-point* data link connects exactly two nodes; a *multi-access* data link can connect two or more nodes. A network port can be connected to a point-to-point link or a multi-access network. A VNP is a *point-to-point* VNP if the corresponding remote port connects to a point-to-point link; similarly it is a *multi-access* VNP if the remote port connects to a multi-access network. Just like in the case of physical network ports, transmission through a multi-access VNP requires the address of the destination node to be specified with each packet; the destination address is not needed in the case of point-to-point VNP. Multi-access VNPs help in reducing the number of flows through the core of the network. We will see later that the processing overhead introduced by multi-access VNPs is not all that significant.

2.4 VNP Path

Packets transmitted through a VNP are sent out through the remote port that the VNP represents. This is accomplished by associating a path, to the remote port, with each VNP. The path consists of three components: the address of the remote node, the local port to reach the node, and the remote port on the remote node. It is a tuple of the form ($L_P \rightarrow A_N \rightarrow R_P$) where

- **L_P** - Local Port - is the port on the local node, where the VNP is defined, to be used to reach the next-hop node.
- **A_N** - Remote Node - is the address of the remote node. It can be a layer-2 or layer-3 address. It is *null* if L_P is a point-to-point interface.
- **R_P** - Remote Port - is the port on the remote node. It is the port that leads to the destination. The access types of the remote port and the VNP it defines should be the same.

A packet sent through a VNP is encapsulated in a VNP frame. The header of a VNP frame contains an identifier for the remote port R_P. In case of a multi-access VNP, the VNP frame header also contains the address of the destination node on the remote network. A complete description of the VNP frame format is given in Chapter 5. The VNP frame is enclosed in a layer-2 frame, and sent to the remote node A_N through the local port L_P. VNP layer on the remote node removes the VNP frame, resolves the destination address if needed, and sends the packet through the remote port. VNP frames are handled at the VNP-layer and do not go to Layer-3 at the remote node.

Figure 2-2 shows how a packet flows through a VNP. For the sake of simplicity, we do not show Layer-2 frames; in reality any packet sent over a data-link gets encapsulated in a Layer-2 frame. N_1P_2 is a virtual network port that represents the remote port N_2P_2; its path is ($N_1P_1 \rightarrow N_2 \rightarrow N_2P_2$). A packet for node N_3 arrives at N_1; N_1 sends it through the VNP N_1P_2. The VNP layer encapsulates the packet in a VNP frame. The frame header contains the remote port identifier N_2P_2 and the destination address N_3. The VNP frame is then sent to the remote node N_2 through the local port N_1P_1. The VNP layer at N_2 removes the VNP frame, resolves the address N_3 if needed, and then sends the packet through N_2P_2. The frame does not go through Layer-3 at node N_2.

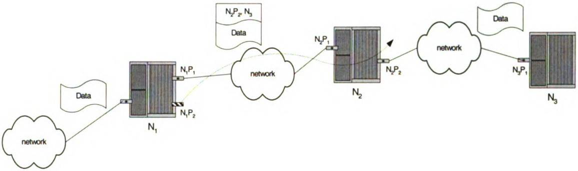


Figure 2-2 Packet Transmission

2.5 Generic VNP

The VNP defined so far was actually of a special type, called the *unicast* VNP. It represents a single remote port, and it simply forwards the packets to the remote port. A *generic* VNP has multiple remote ports associated with it. It may not simply forward the packets to all the remote ports. It may forward them selectively; it may police them; it may even mangle the packets. The *type* of a generic VNP is determined by how it handles the packets.

Figure 2-3.shows a VNP that links to a set of remote network ports. It may choose one or more remote ports for forwarding the packets. This choice can depend on a

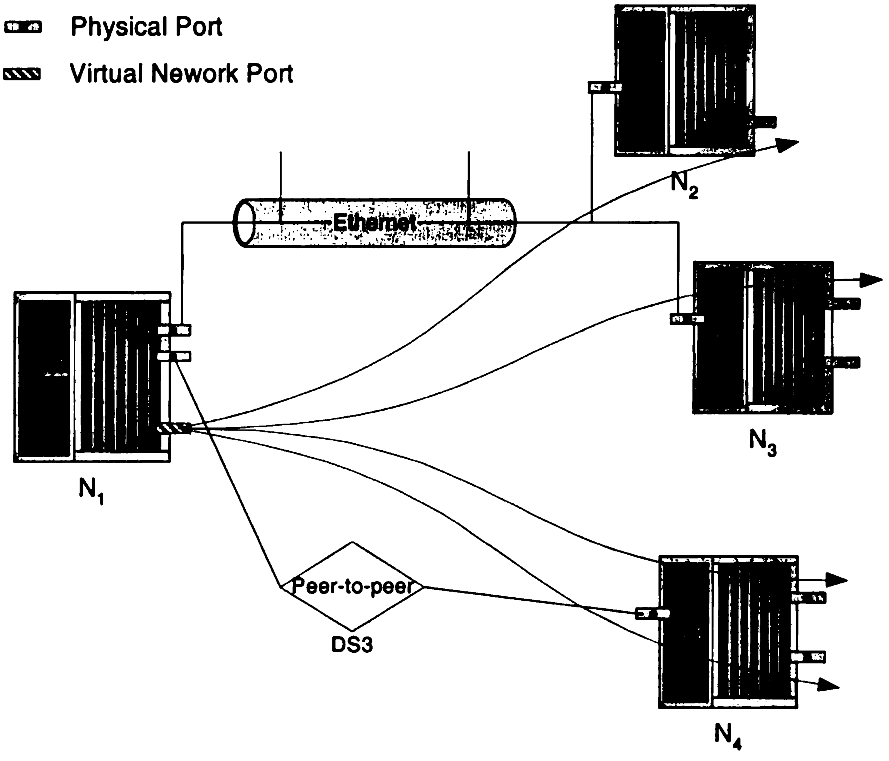


Figure 2-3 Virtual Network Port

number of factors: the load on the links, the source of the packets, the destination etc.

2.6 VNP Attributes

A physical network port has a number of properties: a port identifier, Maximum Transmission Unit (MTU), access type etc. A VNP inherits some of these properties and their values from the remote port in its path. In addition, it has some attributes of its own. The following are the essential attributes of a generic VNP.

- **Port Identifier**: It uniquely identifies a virtual port on the local node.
- **Access Type**: It indicates whether the VNP is a multi-access or point-to-point port.
- **Path-set**: A non-empty set of VNP paths to the remote destinations.
- **Control Procedure**: It is responsible for processing the in-coming packets and forwarding them along the paths in the path set.
- **Maximum Frame Size** (MFS): It is the largest VNP frame that can be transferred through the port.
- **Type**: A VNP's type determines the control procedure and special attributes it can have.

These form the core attributes of a VNP. Depending on its type, a VNP may have other attributes associated with it. For example, a *filter* VNP will have rules associated with it to choose the forwarding paths from the path set.

2.6.1 Port Identifiers

On a node that supports VNPs, each network port -physical or virtual- is uniquely identified by an integer. The identifier P_{max} demarcates physical ports from virtual ports; values that are less than P_{max} represent physical ports, whereas identifiers greater than P_{max} are for virtual ports. The loopback interface is always represented by the special value zero[4]. So a zero for L_P in a VNP path represents the local node, and a zero for R_P represents the next-hop. The value for P_{max} is specific to a node. It helps in determining if a port is physical or virtual; this is used in the switching operation, as we will see later in this chapter. Port identifiers are meaningful only to the local node. A node may not know whether an identifier for a port on another node represents a physical or virtual port.

2.6.2 Path Set

Unlike a unicast VNP, which has a single VNP path associated with it, a generic VNP has a set of VNP paths. Its control component decides how to forward the packets among the paths. When a VNP has multiple remote ports associated with it, all the remote ports must be of the same access type; either all of them must be point-to-point ports or all of them must be multi-access ports. As we will see in the next chapter, simple vector-distance routing protocols can be used for laying unicast VNPs in a domain. However, a multi-path VNP needs more complex routing protocols, like QoS routing protocols, that can deal with multiple paths.

[4] In case there are multiple loopback interfaces, at least one of them is given the zero identifier.

2.6.3 MTU

The Maximum Transfer Unit (MTU) of a network port is the maximum amount of data that can be transferred through the port in a single frame. It depends on the physical network, and is generally a constant for a physical network port. For example, the MTU of most implementations of Ethernet is 1500 octets, whereas for FDDI it is 4470 octets.

The MTU of a VNP is not a constant because VNP header is of variable length. The length of VNP header is dependent on the type of destination address used (DA_{type}) and the type of payload transferred (PA_{type}). The complete format of a VNP frame is detailed in Chapter 4.

Maximum Frame Size (MFS) of a VNP is the largest VNP frame that can be transferred through the VNP. The size of the largest VNP frame that can be sent through the VNP path $L_P \rightarrow A_N \rightarrow R_P$ is restricted by the MTUs L_P and R_P. . The MFS of a VNP is the minimum of all the MFSs in its path-set. The MTU of a VNP is the difference between its MFS and header length. This is summarized below, where VHL is a function that computes the VNP header length for a given payload and address type. An application or a higher-layer protocol must calculate the corresponding MTU before sending packets through a VNP.

$$MFS(T) = Min(MTU(L_P), MTU(R_P)) \text{ where } T \text{ is the path } L_P \rightarrow A_N \rightarrow R_P$$
$$MFS(V) = \underset{T \in PathSetof\,V}{Min}\left(MFS(T)\right) \quad \text{where } V \text{ is a } VNP$$
$$MTU(V) = MFS(V) - VHL(DA_{Type}, PL_{Type})$$

2.6.4 Control Procedure

The control procedure for a virtual port is its "device driver". It is responsible for transmitting the packets through the VNP. The control procedure may not forward the packets on all the paths in the path set. It may choose a subset of its path set for forwarding the incoming packets. The selection can be based on the contents of the packet, the cost of the paths in the path set, or the service guarantees for the flow. In the simplest and most oft-occurring case all the paths in the path set are chosen. The control procedure may police the flow or even modify the packet.

2.7 VNP Table

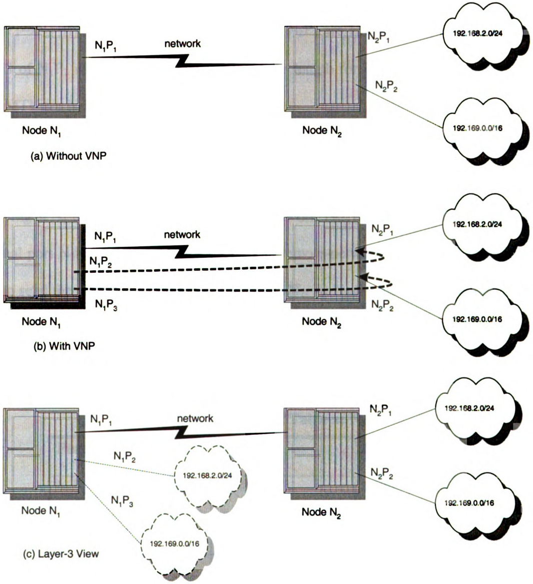
The attributes of all the virtual network ports on a node are stored in the VNP table. Its structure is shown in Table 2-1. It is indexed on the port identifier field, Port-ID. It could be part of the device table that is already on the node.

Table 2-1 VNP Table

Port-ID	Type	Path Set	MFS	Status	...

2.8 A VNP Example

Part (a) of Figure 2-4 shows two nodes N_1 and N_2 connected through a network link. Node N_2 has two network interfaces N_2P_1 and N_2P_2 connecting to IP networks 192.168.2.0/24 and 192.169.0.0/16 respectively. Node N_1 has one network interface N_1P_1 that connects it to node N_2. Packets arriving at N_1 but destined for any of the two



(a) Without VNP

(b) With VNP

(c) Layer-3 View

Figure 2-4 A VNP Example

networks are forwarded by IP to the next-hop N_2 through the port N_1P_1. These packets go through IP layer again at N_2, and are sent to their destinations through the interfaces N_2P_1 or N_2P_2.

Part (b) of the figure shows the same two nodes with virtual-network ports defined at node N_1. The virtual network devices N_1P_2 and N_1P_3 form one-way conduits to N_2P_1 and N_2P_2 respectively. N_1P_2 is a multi-access VNP; its path-set is $\{N_1P_1 \rightarrow N_2 \rightarrow N_2P_1\}$. If the MTUs of N_1P_1 and N_2P_1 are 1500 and 1492 respectively, the MFS of N_1P_2 will be 1492 octets. VNP header length for IP packets is 8 octets. Hence, the MTU for N_1P_2 is 1484 octets.

To the IP layer at N_1 it appears that the two networks are local i.e. connected directly through a hardware network interface as shown in part (c) of the figure. A packet arriving at N_1 but with its destination on the network 192.168.2.0/24 is handed over to the virtual port N_1P_2 with the target address. The remote-network device N_1P_2 uses its path-set to send the packet with the target address to N_2. At N_2 the packet does not go to the IP layer; it is sent out directly through the port N_2P_1. If needed, ARP is done for the target address at node N_2.

Table 2-2 VNP Table at N_1

Port-ID	Type	Path Set	MFS	Length	...
N_1P_1	Physical	{ }	1500	0	
N_1P_2	Unicast	$\{N_1P_1 \rightarrow N_2 \rightarrow N_2P_1\}$	1492	1	
N_1P_3	Unicast	$\{N_1P_1 \rightarrow N_2 \rightarrow N_2P_2\}$	1492	1	

Table 2-1 shows the VNP table at node N_1. Table 2-3 shows some of the VNP entries in the forwarding table at node N_1.

2.9 Switching

A virtual network port represents a set of network ports on remote nodes. The remote nodes have to be on the same physical network; a VNP path cannot have multiple hops. However, a number of single-hop VNPs can be linked together to form a multi-hop VNP path. This linking allows packets to be switched across networks. It also enables construction of traffic paths that are independent of Layer-3 paths.

The structure of a VNP is defined in terms of other network ports: the local port L_P and the remote port R_P. Since a VNP is just like any other network port it can be used to define other VNPs. So far, we have considered the remote port (R_P) to be a physical network port. If the remote port is also a virtual port, the actual physical port being represented is across two network links; the path length has grown to two. Similarly, many single-hop VNP paths can be linked together to form a multi-hop VNP path. This form of linking VNPs together is termed *Switching*.

Table 2-3 Routing Table at N_1

Destination	**Next Hop**	**Port**	**...**
192.168.2.0/24	Local	N_1P_2	...
192.169.0.0/16	Local	N_1P_3	...
....	192.169.1.10	N_1P_3	...
....	192.168.2.2	N_1P_2	...

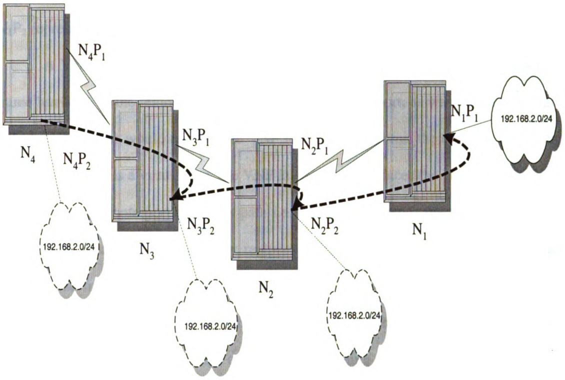


Figure 2-5 VNP Switching

Figure 2-5 illustrates this concept. Node N_1 is directly connected to the network 192.168.2.0/24; node N_2 has a VNP N_2P_2 to this network. Node N_3 has a virtual port N_3P_2 to N_2P_2; N_4 has a VNP N_4P_2 to N_3P_2. The VNP paths for N_4P_2, N_3P_2, N_2P_2 are: $N_4P_1 \rightarrow N_3 \rightarrow N_3P_2$, $N_3P_1 \rightarrow N_2 \rightarrow N_2P_2$, and $N_2P_1 \rightarrow N_1 \rightarrow N_1P_1$ respectively. In the figure the arrows show the VNP paths whereas the clouds show the view from Layer-3.

2.9.1 Data Flow

Figure 2-2 showed how packets flow through a single-hop VNP, whose remote port is a physical network port. When VNPs are switched together, i.e. the remote port is a virtual network port, packets are forwarded in a slightly different fashion. Figure 2-6 shows the flow of packets through a switched VNP. N_1P_2, N_2P_2, and N_3P_2 are point-to-point VNPs; N_1P_2 represents N_2P_2, and N_2P_2 is linked to N_3P_2.

A VNP frame arrives at N_1. The VNP layer at N_1 looks at the VNP header; the frame has N_1P_2 as the remote port. If N_1P_2 were a physical port, as was the case in Figure 2-2, the VNP layer would remove the VNP frame, and transmit the data packet through N_1P_2. Since N_1P_2 is a virtual port, the VNP layer simply replaces the port identifier in the frame with the remote port in N_1P_2's path i.e. N_2P_2; it then sends the packet to the next-hop in N_1P_2's path, N_2. N_2 similarly replaces N_2P_2 in the VNP header with N_3P_2, and forwards the frame to N_3, and so on. Finally at the egress node, where the remote port would be a physical port, the VNP frame is discarded, and the data packet is delivered to the destination. The following steps outline the forwarding of VNP frames.

6. Let P be the port in the VNP frame's header [P=Frame.Header.PortID].
7. Lookup for P's path in the VNP table; let ($L_P \rightarrow A_N \rightarrow R_P$) be the path.
8. Replace port in the frame header with R_P [Frame.Header.PortID=R_P].
9. Transmit the frame through L_P to A_N.

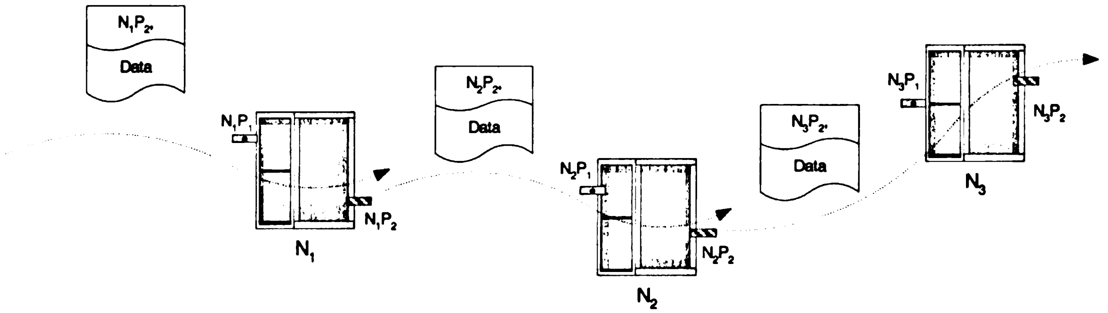


Figure 2-6 Packet Switching

2.10 Cascading

A virtual port can be defined using another virtual port as the transport, i.e. as local-port L_P. This cascading of VNPs can be used to reduce the number of virtual ports on a node as illustrated in Figure 2-7. Node N_4 has two VNPs defined for the two ports connected to N_1. One way to accomplish this is to define two VNPs corresponding to the two ports at each intermediate node, as was done in the case of switching. The other way is to define a virtual point-to-point port N_4P_3 to the loop-back port of N_1. The intermediate nodes have only one VNP defined, to the loop-back port of N_1. At node N_4, the VNPs N_4P_4 and N_4P_5 use the VNP N_4P_3 as the local port L_P. The paths for N_4P_3, N_4P_4 and N_4P_5 are ($N_4P_1 \rightarrow N_3 \rightarrow N_3P_2$), ($N_4P_3 \rightarrow$null$\rightarrow N_1P_1$), and ($N_4P_3 \rightarrow$null$\rightarrow N_1P_2$) respectively. Conceptually N_4P_3 is a local point-to-point connection to N_1; N_4P_4 is the virtual representation of N_1P_1 through this point-to-point connection.

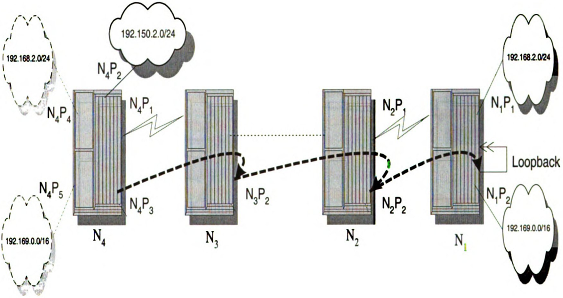


Figure 2-7 VNP Cascading

2.10.1 Data Flow

Figure 2-8 shows frames through a cascaded VNP. Node N_1 has two VNPs: N_1P_2 and N_1P_3. N_1P_2 is a point-to-point VNP to node N_4 i.e. to N_4's loopback interface. N_1P_3 is a cascaded VNP to the remote port N_4P_2. It cascades over N_1P_2 to N_4; its path is (N_1P_2→null→N_4P_2).

A VNP frame with port N_1P_3 in its header arrives at N_1. N_1P_3 is a virtual port, so N_1 replaces the port in the frame with the remote port in N_1P_3's path i.e. N_4P_2. It then hands over the frame to the local port, N_1P_2, for transmission. N_1P_2 treats the frame as payload. It encapsulates the frame in another VNP frame, sets the port in the header as N_2P_2, and sends it to the next-hop N_2. To N_2, it appears like just another VNP frame. It switches N_2P_2 with N_3P_2, and forwards the frame to N_3. N_2 does not see the embedded frame. Similarly N_3 replaces the port in the header with N_4P_0, the loopback port of N_4,

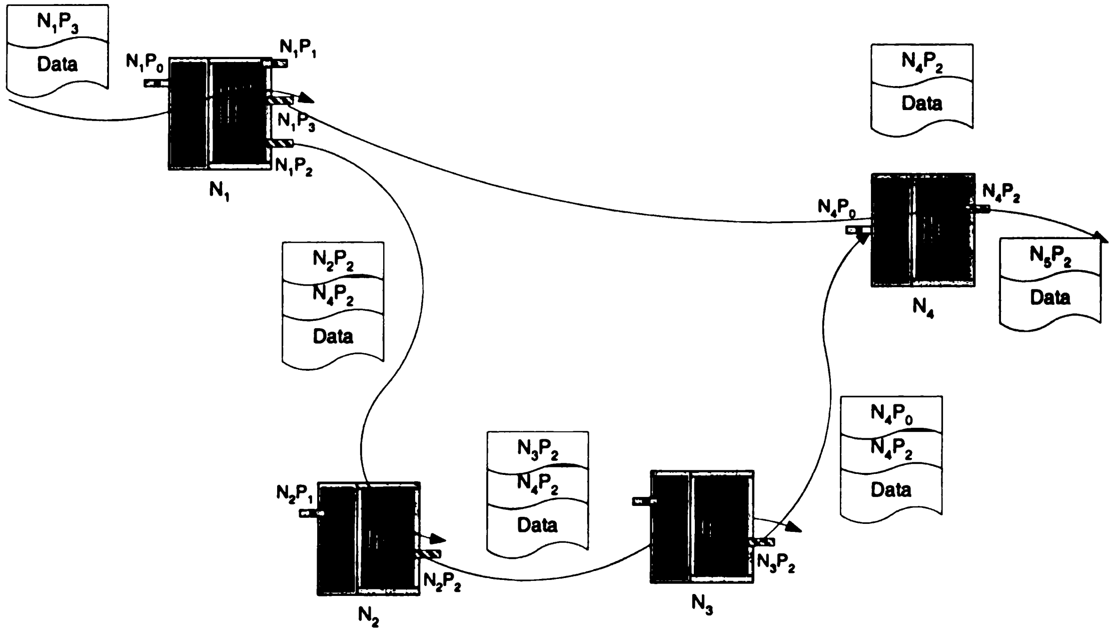


Figure 2-8 Packet Cascading

before forwarding it to N_4.

VNP layer at N4 de-encapsulates the frame because N_4P_0 is not a virtual port; otherwise the frame would be switched. The payload i.e. the inner VNP frame is then sent through N_4P_0. N_4P_0 is the loopback port; hence the inner frame is handed over to the VNP layer again. This time it switches the frame to the next-hop because the port in the header, N_4P_2, is a virtual port.

2.11 MTU with Switching

The MTU of a VNP is the difference between its MFS and header length. When the remote port is a VNP, the packet is simply switched; no more headers are added to the VNP frame. Hence the MFS of a VNP path is the minimum of *MTU(L_P)* and *MFS(R_P)*; and not *MTU(R_P)* as we had mentioned before.

In the case of cascading, L_P is a virtual port. Even though the MTU of a VNP is variable the MTU of VNP L_P is fixed, for a given VNP path, because the header length used over a VNP L_P can be determined from the VNP path. The header length of a VNP L_P is *VHL(Type(A_N), VNP),* where A_N is the remote node's address in VNP's path. For a VNP path, the destination address type, *Type(A_N)*, is fixed. The payload in this case is always a VNP frame, so the payload type is also fixed (as *VNP)*. Hence, taking switching and cascading into consideration, the MTU of a VNP is:

$$MFS(P) = MTU(P) \quad where\, P\, is\, physical\, port$$
$$MFS(T) = Min(MTU(L_P), MFS(R_P))\; where T\, is\, the\, path\, L_P \rightarrow A_N \rightarrow R_P$$
$$MFS(V) = \underset{T \in PathSetof\, V}{Min}\left(MFS(T)\right) \quad where V\, is\, a\, VNP$$
$$MTU(V) = MFS(V) - VHL(DA_{Type}, PL_{Type})$$

In Figure 2-5, the MFS of N_2P_2 is the minimum of $MTU(N_2P_1)$ and $MFS(N_1P_1)$. Similarly the MFS of N_3P_2 is the minimum of $MTU(N_3P_1)$ and $MFS(N_2P_2)$. So the MFSs of the VNPs at various nodes along the same VNP conduit will not be the same but will be optimal i.e. it will be the maximum MFS that can be used without fragmenting the packets flowing through the VNP conduit. This optimality is true only for unicast and multicast VNPs. A generic VNP may choose different forwarding paths for different packets; and hence the MFS may not be optimal for all the paths but the packets will still be forwarded without fragmentation. In the case of cascading, MTU of VNP does take into account the cascaded VNP headers; so the packets through a VNP need never be fragmented, no matter how deep it is cascaded. In the next chapter we will see how MTU of a multi-hop VNP is computed with changes in the VNP's path.

2.12 A Multicast Example

As mentioned before, a VNP represents a set of multi-access networks or nodes. We now illustrate this by defining a multi-cast group using VNPs in three different ways, as a set of hosts, as a set of multi-access networks and as a set of routers. Just like labeled flows, VNP flows are by nature protocol independent tunnels. They can be used to tunnel multi-cast traffic over non-multicast networks. Referring to the example of Figure 2-7, suppose certain hosts on the networks 192.168.2.0/24 and 192.150.2.0/24 are part of a multi-cast group but nodes N_3 and N_2 do not participate in multi-casting. A multi-cast VNP can be defined on N_4 with the path-set as $\{N_4P_4 \rightarrow M_1 \rightarrow 0, N_4P_2 \rightarrow M_1 \rightarrow 0\}$ where M_1 is the layer-3 multi-cast address. This defines the VNP as a collection of hosts that are part of the multicast group. The first path in the path set, $N_4P_4 \rightarrow M_1 \rightarrow 0$, makes the packet switch through to N_1P_1. At the node N_1, M_1 would be resolved to its Layer-2 address.

Another way to define the multi-cast VNP is with the path set $\{N_4P_3 \rightarrow null \rightarrow N_1P_1, 0 \rightarrow null \rightarrow N_4P_2\}$. This defines the virtual port as a collection of networks that lead to the multicast recipients. There is a third way to define the multicast VNP, as a set of routers that lead to the multicast group. In this case the path set will be $\{N_4P_3 \rightarrow null \rightarrow 0, 0 \rightarrow null \rightarrow 0\}$. In the first and the last definitions a target address need not be specified when transmitting a packet through the VNP because the destinations are nodes, whereas in the second method the target address (M_1) must be specified

2.13 VNP Addressing

VNP does not use any special addressing mechanism to identify nodes on a network. It uses the existing Layer-2 or Layer-3 addressing. We have come across two types of addresses in dealing with virtual network ports: the next-hop address in a VNP path and the destination address for packets through a multi-access VNP. The format of the next-hop address depends on the local port in the VNP path. If the local port is a physical port, then the next-hop address is a Layer-2 address. The destination address in packets through a multi-access VNP is generally a Layer-3 address, especially in the case of IP networks. However, it can be a Layer-2 address too. Cascading, over multi-access VNP, is a special case of this kind of addressing. In this case, the next-hop address can be a Layer-3 or Layer-2 address. We will see in the next chapter that Layer-3 addressing is used for setting up VNPs across network domains.

Issues like this make it hard to place VNP in the OSI model hierarchy. As far as packet flow is concerned, VNP layer falls between the data link layer (layer-2) and the network layer (layer-3). However, it does use Layer-3 mechanisms for its operation.

Chapter 3

VNP ARCHITECTURE

This chapter describes how Virtual Network Port paths are built in a large network. VNPs are setup in a hierarchical fashion. The hierarchy is based on dividing the nodes into disjoint sets called domains. A domain can be split into multiple sub-domains, or can be contained in another domain. Non-cascaded VNPs are constrained to be within a domain; these VNPs are used as transports by VNPs that straddle across multiple domains. The VNPs that cross multiple domains are always cascaded. It is possible to use non-cascaded VNPs across domains, but it is not generally done to avoid scaling problems.

3.1 VNP Domains

A set of interconnected nodes that understand VNP protocols forms a *VNP-domain.* The members and boundary of a VNP-domain are setup manually, and are determined by technical or administrative reasons. A VNP-domain has two disjoint parts: *edge* and *core.* The nodes in a VNP-domain that connect to nodes outside of the VNP-domain form the edge of the VNP-domain; they are called the *edge nodes.* The rest of the nodes in the VNP-domain, that interconnect the edge nodes, form the core of the VNP-domain, and are called the core nodes. Figure 3-1 shows two VNP-domains with respective core, edge, and VNPs. The edge nodes connect to the edge nodes of other VNP-domains, or to the nodes that do not follow VNP protocols. VNPs always originate and end at the edge nodes.

3.2 Setting up VNPs in a VNP-domain

Inside a VNP-domain, a VNP is set from an edge node to destinations on other edge nodes. The destination could be a port on the edge node, or the edge node itself. Figure 3-2 shows the procedure for setting up a VNP to a remote *Destination* in the same VNP-domain. The *Destination* is used to generate a VNP identifier using the procedure **Gen-VID()**. The implementation of this procedure depends on the node. For most nodes it is a function, which maps a *Destination* to a unique integer. However, for ATM switches it

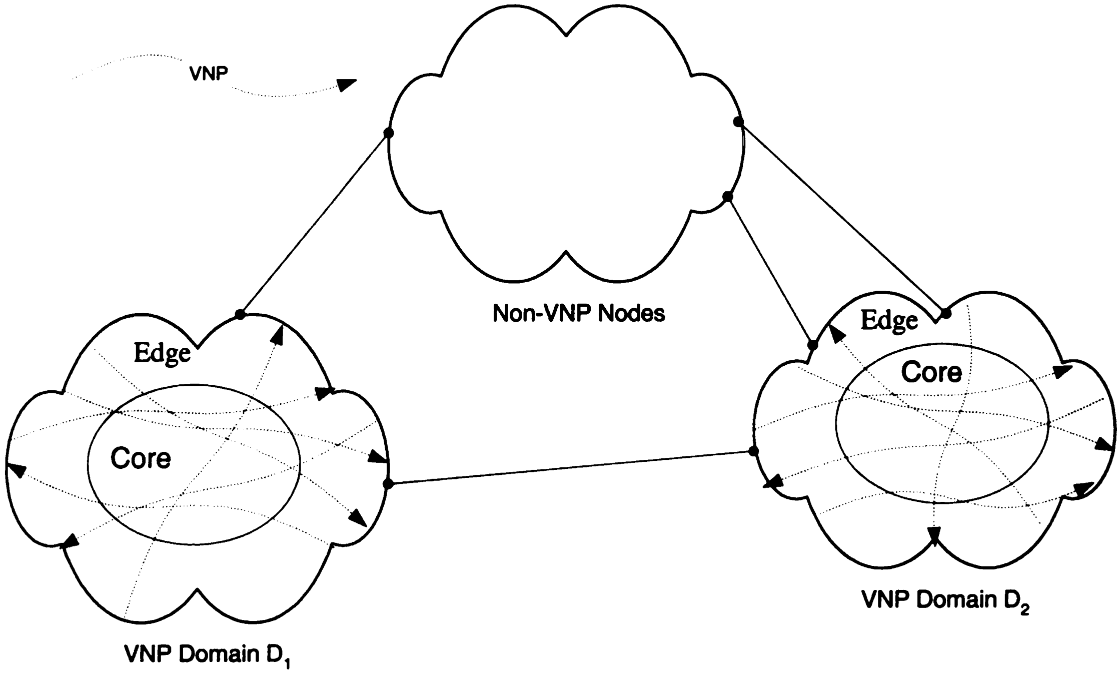


Figure 3-1 VNP Domains

may return different integers for the same *Destination*. If this VNP already exists, there is nothing for the SetupVNP procedure to do. A set of next-hops to the *Destination* is determined by the procedure **NextHopsSet()**. The following are performed for each next-hop ($A_N[i]$).

- The local port to reach the next-hop is determined ($P_L[i]$).
- The next-hop is then contacted to obtain VNP identifier ($P_R[i]$) corresponding to the *Destination*, and its MFS ($MFS_R[i]$).
- Size of the largest VNP frame that can be sent to the next-hop is calculated ($MFS_{Path}[i]$).

The MFS of the VNP is determined by finding the minimum, of all the MFSs of remote ports ($P_R[i]$) and MTUs of local ports ($P_L[i]$). If a local port $P_L[i]$ is a VNP, its MTU is calculated using the formula, **MTU**($P_L[i]$) = **MFS**($P_L[i]$) - **HeaderLen**($A_N[i]$.AddressType, VNP). The length of a VNP header depends on type of the destination address and type of the load; the **HeaderLen**() procedure calculates this length. Finally, an entry is created in the VNP Table with the VNP identifier, MFS and the path-set.

3.2.1 VNP Ends

The **SetupAVNP()** procedure lays a VNP path from a source to a destination. The source triggers a chain of processes that follow the VNP path to the destination. Any node can initiate this chain. However, the data always enter through edge nodes, hence initiation of VNP paths is restricted to the edge routers.

Similarly, it is possible to create VNPs to all possible destinations in a domain but it is not very efficient. Generally, VNPs are created only to destinations on the edge routers, and then only to those edge nodes that need optimized data paths. Hence the VNP paths are restricted to start and end with edge nodes of a VNP domain

3.2.2 Destination

A *Destination* can be any one of the four types: port, node, network, or group. When *Destination* is a port, the VNP is setup to that remote port. For *Destination* as a node, the VNP is setup to the loopback port of the node. When the *Destination* is a network, the

```
SetupAVNP(SourceNode, Destination, Constraint) {
        Check for resources; if resources not available return Error;

        Port = Gen-VID(SourceNode, Destination, Constraint);
        if Port already exists then return Port;
        if this is the egress node then return Port;

        $A_N$[0..Max] = NextHopSet(Destination, Constraint);
        for i = 0, Max {
                $P_L$[i] = Local port to reach $A_N$[i];
                $P_R$[i] = Get the VNP identifier for (SourceNode, Destination, Constraint)
                        from $A_N$[i];
                $MFS_R$[i] = Get MFS of $P_R$[i] from $A_N$[i];
        }
        $MFS_{VNP}$ = Min( MTU($P_L$[i]), $MFS_R$[i] );

        Add an entry to the VNP-Table with ID=Port, MFS=$MFS_{VNP}$ and
                PathSet={ $P_L$[0]→$A_N$[0]→$P_R$[0], … , $P_L$[Max]→$A_N$[Max]→$P_R$[Max] };

        return Port;
}
```

Figure 3-2 Procedure to setup a VNP inside a VNP-domain

VNP is setup to any port that connects to that network. Whereas when *Destination* is a group, the VNP is setup to the loopback ports of the set of nodes that form the group.

The different types for *Destination* need to be identified uniquely, and are represented differently. A node is identified by one of its IP addresses. Many a times, an IP address is assigned to the loopback port of the node, and that IP address is used to identify the node as a destination. A network is identified by the network-part of its IP address and its prefix-length. For example, a network, the first 24 bits whose IP address match 192.168.2.0, is identified by (192.168.2.0, 24). A group, as a *destination*, is identified by an IP multi-cast address.

When *Destination* is a port, it is identified in one of two ways. All the ports that have IP addresses associated with them are identified by the IP address. Unnumbered point-to-point links and ports that do not have an IP-address associated with them, are identified using (Node-ID, Port-ID) tuple. Node-ID is the IP address of the node, and Port-ID is the SNMP MIB's *ifindex* parameter or the IP address of the node at the other end of the link.

It is possible to use (Node-ID, Port-ID) pair to represent all types of destinations. This results in a generic addressing mechanism that does not use any layer-3 addressing. It requires a completely new way of addressing, distribution, and aggregation, without any dependence on Layer-3. However, it will not be very practical. VNPs are used in networks that already have network layer addressing, mostly IP. Hence IP addressing is used for uniquely identifying destinations in a VNP domain.

3.2.3 Access Type

The access type of the port, i.e. whether it is multi-access port or point-to-point port, is determined from the *Destination* type. When *Destination* is a port, the access type of the VNP will be the same as the access type of the remote port. The access type is ‘point-to-point’ when the destination is a node or a group. When the *Destination* is a network, the access type is ‘multi-access’.

3.2.4 Next Hops

Next-hops in a VNP path are determined based on various *constraints*. In the case of traditional routing, the constraint is to find the next-hop that results in the shortest path. With VNPs, the constraint can be to satisfy several QoS or traffic-engineering parameters. Based on the constraint, the NextHopSet() procedure uses the underlying routing methodology to find the set of next-hops; it does not advocate any new routing protocols. In case of shortest-path constraint, it consults the routing information base. For QoS constraints, it uses the information base of any of the underlying QoS routing protocols. It is also possible to specify the nodes in the path explicitly. In this case, either the complete path is specified (strict), or only some of the intermediate nodes are specified (loose).

3.2.5 Control Type

The VNPs created by the above procedure generally have unicast control. The control type can be 'load balancing', 'de-aggregation', 'multi-cast' etc. The control type is set by the routing protocol's actions, or it is set manually.

3.2.6 Multi-Path

Certain link-state IGPS, like OSPF, identify multiple equal-cost paths to a destination. In such cases, the **NextHopSet()** procedure may return a set of next-hops corresponding to the multiple paths. The control type of the VNP is set to *'Load Balancing'* so that the packets are striped across the paths.

```
SetupEVNP (SourceNode, Destination, Constraint) {
    Check for resources; if resources not available return Error;

    Port = Gen-VID(SourceNode, Destination, Constraint);
    if Port already exists then return Port;
    if this is the egress node then return Port;

    From the VNP Edge database, find the boundary router $E_1$ that leads to Destination;
    If the LocalNode and $E_1$ are directly connected then
        let $V_1$ be the physical link between them
    else
        $V_1$ = SetupVNP(LocalNode, $E_1$, Constraint);

    $V_2$ = Get VNP ID for (SourceNode, Destination, Constraint) pair from $E_1$;
    $MFS_{VNP}$ = Min ( MFS($V_1$), MFS($V_2$) );
    Add an entry to the VNP-Table with ID=Port, MFS=$MFS_{VNP}$ and
            PathSet = {$V_1 \rightarrow$null$\rightarrow V_2$};

    return Port;
}
```

Figure 3-3 Procedure to setup a VNP to an external destination

3.3 VNPs to External Destinations

The nodes at the edge of a VNP-domain connect to outside the VNP-domain. They maintain information about destinations external to the VNP-domain. This is generally in the form of aggregated IP prefixes. In an inter-network, the number of destinations external to a VNP-domain may far exceed the number of destinations inside the VNP-domain. Using the procedure of Figure 3-2 for setting up VNPs to external destinations will create too many VNPs in the core of the VNP-domain because almost every VNP has to go through the core. It does not scale well to the number of destinations. Instead, another procedure is used to setup VNPs to external destinations.

This procedure is shown in Figure 3-3. In this procedure the egress node -E_1-, which leads to the destination, is first identified. A VNP, V_1 in Figure 3-4, is then setup to the egress node from the ingress node. This VNP is used as the transport to the external destination. The ingress node creates another VNP, V_2 in Figure 3-4, which

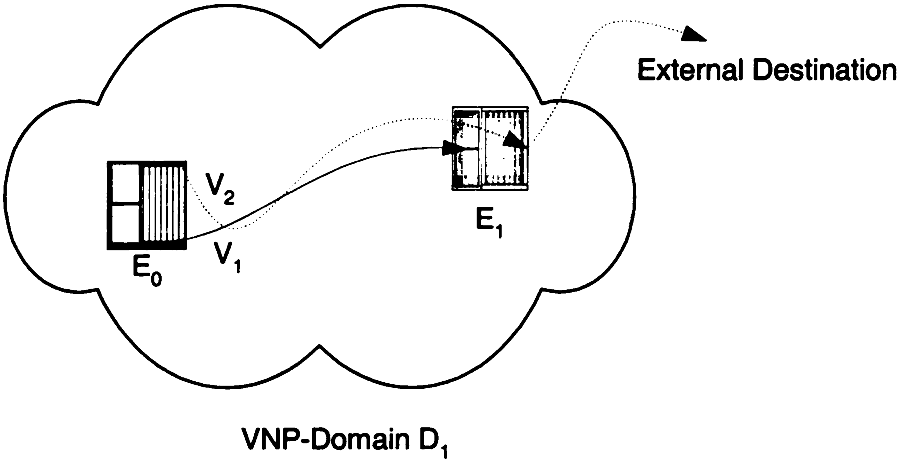


Figure 3-4 VNP to External Desitnation

cascades over V_1. The VNP-tables on the core nodes do not contain entries for V_2; they just have an entry for V_1.

This procedure requires edge nodes to keep information about external destinations that are reachable through other edge nodes. This information base is called the *VNP Edge database*, and is exchanged through *VNP Edge protocol.*

3.3.1 VNP Edge Protocol

The procedure of Figure 3-3 requires that all the edge nodes are aware of external destinations reachable from one another. The edge nodes exchange the destination information among themselves using *VNP Edge Protocol.* It is similar to a routing exchange protocol. Each edge node periodically informs all the other edge nodes in the VNP-domain, of all the external destinations reachable through it. The edge nodes use this information to setup cascaded VNPs to external destinations. The VNP Edge Protocol may become redundant if the edge nodes use a link-state protocol to exchange routing information.

3.4 Independent Mode

Another way to setup VNPs is to distribute information about them in a manner similar to Layer-3 routing updates. In this method, a node creates an entry in the VNP table but does not ask the next-hop for any information about its corresponding VNP; so the path-set and MFS are not known initially. Each node then distributes information about the VNP -its identifier and the corresponding destination port- to its neighbors. The information received from the neighbors is used to fill-up the gaps -path-set etc.- in the

VNP table. With this method, it is possible to have VNPs that are initially 'broken' i.e. the path from the source to the destination-port will not be complete. Packets traveling through such VNPs are discarded at the 'break points'.

3.5 Reducing the number of VNPs in the core

The number of VNPs in the core of a VNP-domain may increase exponentially with the number of VNPs on the edge. This puts a strain on the resources of the core nodes. There are two ways to control this. In the first method, the VNP-domain is broken into smaller VNP-domains to reduce the number of edge nodes. The core also gets broken into smaller sizes. In the second method, the core of the VNP-domain, or parts of it, are

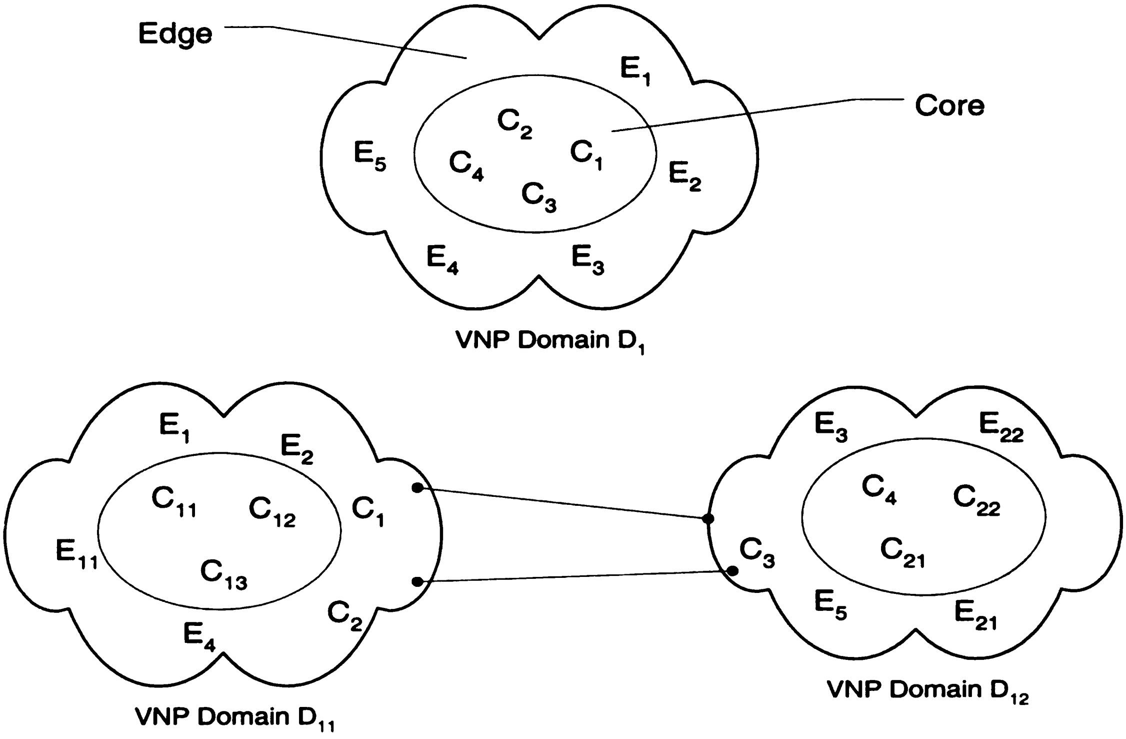


Figure 3-5 Splitting VNP Domains

converted into sub VNP-domains. The core of the VNP-domain now contains the edge of the new VNP-domain.

3.5.1 Splitting a VNP-domain

In this method, a VNP-domain is divided into a number of disjoint VNP-domains. This is shown in Figure 3-5. VNP-domain D_1 is split into two smaller disjoint VNP-domains: D_{11}

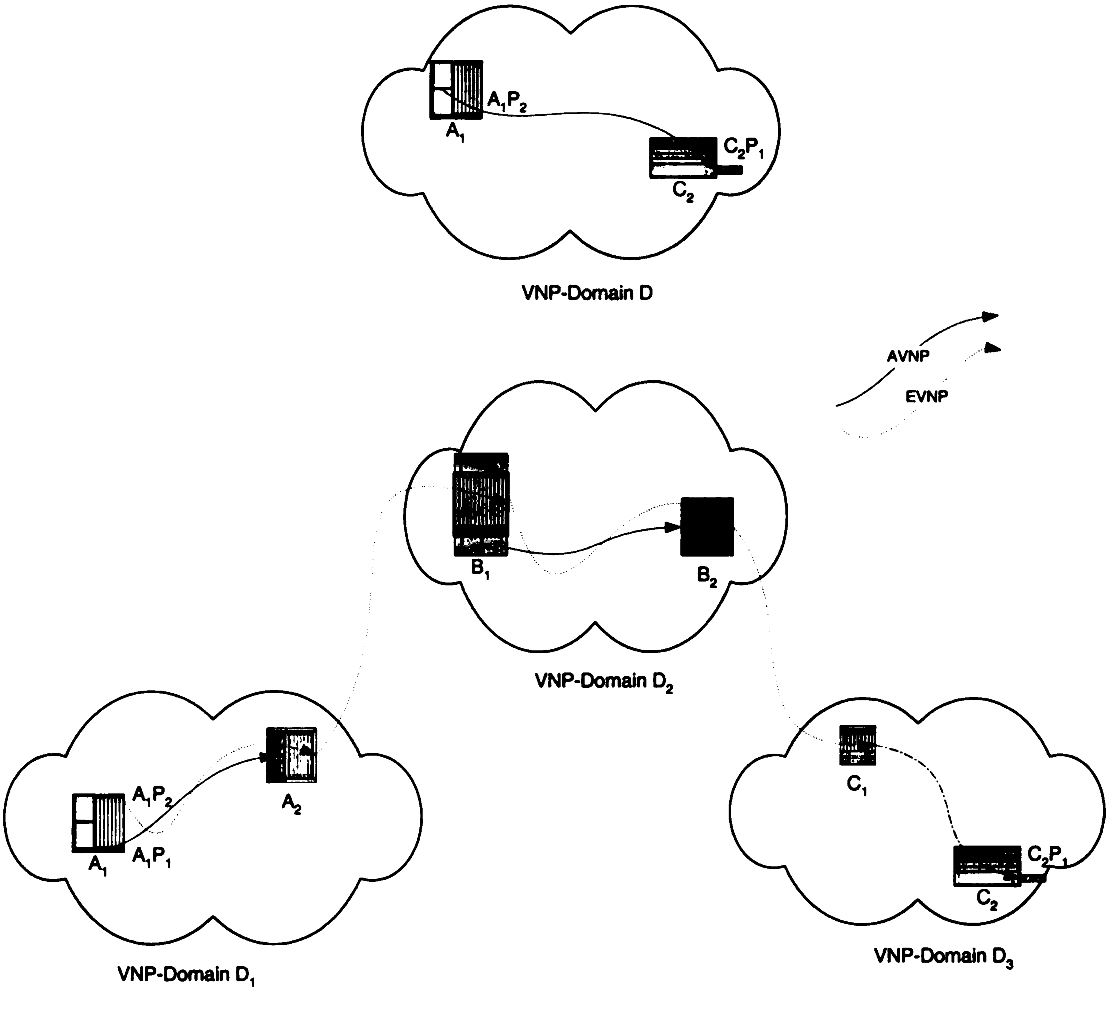


Figure 3-6 VNP Across Domains

and D_{12}. The edge and core nodes are split across the two new domains. Some of the core nodes in the D_1 have now become edge nodes in D_{11} and D_{12}.

3.5.1.1 Intra-domain and Inter-domain VNP

When a VNP domain is split into a number of sub-domains, a VNP that was limited to the VNP-domain now runs across multiple VNP-sub-domains. A VNP, the nodes in whose path are confined to a single VNP-domain, is called an *Intra-domain VNP* (AVNP). If the nodes in the path of a VNP belong to more than one VNP-domain, it is called an *Inter-domain VNP* (EVNP). Inter-domain VNPs are generally cascaded over multiple intra-domain VNPs to form conduits across VNP-domains. The core of each new VNP-domain keeps track of AVNPs only. The cores do not *see* the EVNPs i.e. they do not have entries in the VNP table for EVNPs. The edge nodes of the new VNP-domains cascade the EVNPs over the AVNPs. Each of the new VNP-domains uses the procedure of Figure 3-2 to setup AVNPs. The procedure of Figure 3-3 is used to setup EVNPs.

Figure 3-6 shows a VNP-domain, D, which has a VNP, A_1P_2, from the edge node A_1 to a port on the edge node C_2. This VNP is setup using the procedure of Figure 3-2. Domain D is then split in to VNP-domains D_1, D_2, and D_3. The VNP A_1P_2 now runs across the new domains, and is setup according to the procedure in Figure 3-3. Node A_1 creates an AVNP, A_1P_1, to edge node A_2. It then asks A_2 for the VNP-ID to destination C_2P_1. Node A_2, in turn, asks the next-hop B_1 for the VNP-ID to the destination. B_1 notices that it is an external destination, so it repeats the procedure by setting up an AVNP to the edge node B_2. This process continues till the destination is reached. Node C_1 creates an AVNP directly to C_2P_1 because it is an internal destination.

3.5.2 Core as VNP-Domain

Another way to reduce the size of VNP table on core nodes is to convert the core into another VNP-domain. Once a core is converted into a VNP-domain, it gets split into the partitions of edge and core. Figure 3-7 shows the VNP-domain D_1, of Figure 3-5, with its core converted to another VNP-domain D_{11}. Some of the core nodes of D_1 remain as its core nodes, whereas the rest of them become the edge and core nodes of D_{11}. New point-to-point AVNPs are setup among the edge nodes that previously were in the path of at least one VNP. Only the newly formed edge nodes keep the entries for existing VNPs; the newly formed core does not. The new core has entries only for the new point-to-point VNPs among the new edge nodes, thus reducing the size of the VNP table on the core nodes. The edge nodes cascade the existing VNPs over the new edge-to-edge VNPs. With this method, it is possible to have a cascaded VNP path between two edge nodes of a VNP-domain. For example, the edge node E_1 may have a cascaded VNP, which goes

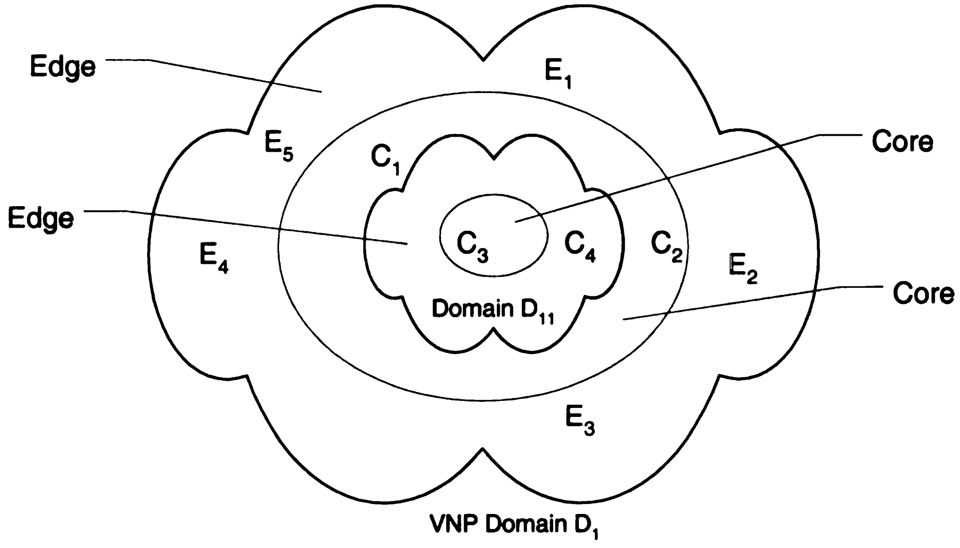


Figure 3-7 Converting Core into VNP-Domain

through domain D_{11}, to the edge node E_2. Whereas when a VNP is split into smaller domains, as shown in Figure 3-5, two edge nodes in the same domain do not have cascaded VNPs between them.

3.6 Routing Updates

VNP paths in an inter-network must change with changes in the inter-network. The core nodes in a VNP domain use routing updates to modify the intra-domain VNP paths. The edge nodes in a VNP domain use the VNP edge protocol to update the inter-domain VNP paths.

Whenever there is a change in the network topology, the IGP or EGP routing protocols update the routing information bases. With every such update, the VNP core node checks to see if any VNP in its VNP-table is affected by the update. This is done by matching the destinations in the routing-updates with VNP destinations. If the next-hop for a destination has changed, the path-set of an affected VNP is modified accordingly. The new next-hop is asked for its VNP ID for the destination, as per procedure of Figure 3-2. For routing updates to the paths of intra-domain VNPs, the edge nodes do exactly what the core nodes do.

However, VNP edge nodes do a little more with routing updates. They exchange updates in routes, to destinations outside the VNP-domain, among themselves using the VNP edge protocol. Hence they get updates from both core nodes and edge nodes. Since the updates obtained through the VNP edge protocol are for external destinations only, the inter-domain VNPs are checked for changes in the paths. The destination in the update is matched against destinations of inter-domain VNPs. If the next-hop for a destination has changed, a procedure similar to the one in Figure 3-3 is used to modify the path-set of corresponding VNP.

3.7 Host-to-Host VNPs

So far, VNPs were considered in the context of core and backbone networks. They were setup from an edge router to another edge router. The procedure for establishing VNPs from a host to another host is similar. Hosts always use the procedure of Figure 3-2 irrespective of whether the destination is internal or external to the domain. The source host checks with the next-hop router for the VNP-ID for the destination host. The router uses the methods of Figure 3-2 and Figure 3-3 to lay the VNP to the destination host.

Once a host-to-host VNP is established, the source host uses it to transport other VNPs, with different service constraints, to the destination. These cascaded VNPs can then be used by different applications requiring different quality of service. The destination generates an application VNP-port, and informs about it to the source host using VNP Host Protocol. The source node then cascades a VNP to this remote VNP port.

3.8 Propagating MTU changes

The procedures of Figure 3-2 and Figure 3-3 compute the MFS of the VNP that is being setup. Whenever there is a change in the topology, the MFS of VNPs may get altered. The method for updating MFS of a VNP, with changes in topology, is different for inter-domain and intra-domain VNPs.

3.8.1 For Intra-domain VNP

For intra-domain VNPs, the core nodes inform all the edge nodes in the domain regarding any changes in the MFS. Whenever there is routing update, the core nodes re-compute the MFS of VNPs affected by the updates. If there is a change in the MFS, the core nodes inform the edge nodes. This is done in a number of different ways.

The core nodes multi-cast the destination of the VNPs whose MFS has changed, to the edge nodes. The edge nodes then re-compute the MFS of corresponding VNPs by probing the next-hops. This is similar to the way MFS is computed in the procedure of Figure 3-2; the process starts at the source edge node and ends only at the destination node. In this method, a core node does not need to keep track of the *pre-hops*, i.e. the previous nodes in the VNP-path, of VNPs. However, on an edge node, there can be multiple VNPs associated with a destination. Therefore, the edge nodes will have to re-compute the MFS of all the VNPs corresponding to a destination, even though it need be done only for one of them.

This is avoided in the second method, where the core nodes keep more state information about the VNPs. Each core node, with every local VNP, keeps track of the nodes to which it has sent the VNP-ID for that VNP. Whenever a core node receives the request for a VNP-ID, it notes down the address of the requesting node. This requires only a small change to the procedure of Figure 3-2. Whenever a core node detects change in the MFS of a VNP, due to routing updates or a MFS-update from other nodes, it informs all the corresponding previous-hops for that VNP. The pre-hops in turn follow the same procedure, and inform their pre-hops, and so on. Finally, the chain stops with the edge nodes. This method has two advantages. Only those edge nodes whose VNPs are affected, are informed about the change. Secondly, the edge nodes do not have to probe for the re-computation of the MFSs. They get the updated MFS automatically. Whereas in the first method, the edge nodes are informed about the destinations for which the MFS need to be recomputed; the edge nodes have to initiate the process of re-computing the new MFS.

3.8.2 For Inter-Domain VNP

The edge nodes always keep track of the (external) nodes that have requested for VNP-IDs. The change in MFS for an inter-domain VNP, at the edge nodes, is triggered by three events

- Next-hop of the VNP has changed
- MFS of the intra-domain VNP that was used for cascading the inter-domain VNP, has changed
- There is an update from the next-hop indicating a change in the MFS

In all these events, the edge node re-computes the MFS of the VNP. It then informs all the pre-hops of the VNP of the change in the MFS. The pre-hops in turn do the same. This process goes on till the source nodes are reached.

3.9 Implementation On IP Networks

Even though VNPs can be implemented over any kind of networks, they are generally implemented on IP networks. Such implementations depend on the underlying IP infrastructure. An IP inter-network is divided into independent and disjoint Autonomous Systems (AS). VNP-domains map directly into AS; each AS is represented as a VNP-domain. An AS may use a link-state protocol or a vector distance protocol as its Interior Gateway Protocol (IGP). When a link-state protocol is used, an AS may get further divided into *areas*. In such cases, a VNP-domain may correspond to each area. It is not necessary to have VNP-domain correspond to the divisions, i.e. AS and areas, in IP networks. However, it makes the implementation of VNP-domains easier and more efficient.

Autonomous systems currently use BGP, a vector-distance protocol, to exchange routing information among themselves. All the destinations external to the AS are represented by VNPs to the AS boundary routers. AS boundary routers become edge nodes of a VNP-domain, and indicate the destinations for which inter-AS VNPs are to be configured.

A combination of methods is used to setup VNPs inside an AS. If the AS uses a link-state routing protocol, both the link-state database and the routing table can be used to setup the VNPs. Only the routing table can be used to setup VNPs across autonomous systems. If the AS is divided into areas, similar to OSPF areas, then link-state database is used inside the area, and routing table is used to setup VNPs across areas.

3.9.1 Using Link State Database

In an autonomous system that uses link-state Interior Gateway Protocols like OSPF and IS-IS, each router maintains a directed topological graph of the AS. The vertices of the graph consist of routers and networks. An edge connects two routers if they are attached via a point-to-point network. An edge connecting a router to a network indicates that the router has an interface to the network. The graph is developed and updated through link state advertisements. A router generates its routing table from this graph by calculating a tree of shortest paths with the router itself as the root. The tree gives the entire route to any destination network or node. The shortest-path tree is used to generate the next-hop to the desired port over the shortest path. Destinations external to the domain appear as stub networks connected to boundary routers.

Like any other node, each edge node has the complete topology of the domain. This is used to several advantages in implementing VNPs. Modifications to VNP-paths with routing updates, is made more efficient. Calculation of MFS with changes in VNP paths also becomes efficient and simpler. Each edge node knows the external destinations reachable through other edge nodes. Hence, the VNP Edge Protocol becomes redundant when link state database is used.

3.9.1.1 Routing Updates

Routing updates result in the modification of VNP-paths. As described in Section 3.6, both core and edge nodes participate in this process. However, when link-state IGP is used in a VNP-domain, the participation of core nodes is eliminated. In this method, the edge nodes initiate all the modifications to VNP paths.

Topological changes result in a new SPF tree. An edge router compares the new SPF tree with the previous one, and finds the destinations for which the paths have changed. For destinations inside the area, it executes the procedure of Figure 3-2 to re-setup the corresponding AVNPs. For destinations outside the area, it executes the procedure of Figure 3-3 to re-establish the corresponding EVNPs. The core nodes remain passive to the routing changes from the perspective of VNPs.

3.9.1.2 MFS

The determination of the MFS of a VNP is more efficient if link-state database is used for setting up the VNP. In this environment, a VNP is associated with the corresponding destination edge node. The MTU of each link is maintained in the link-state database. This is not usually propagated by link-state protocols, but it can be easily added to them. This does not cause any increase in the advertisement traffic because the MTU of a link hardly ever changes. The VNPs are always setup among edge routers, so it is very easy for the edge routers to calculate the MFS of a VNP. An edge router knows the path of a VNP, and the MTUs of all the links along the path. This information is available in the link-state database. An edge router just calculates the minimum of all the MTUs along the path. This will not work if a VNP was cascaded on to another one by a core router in an area. However, note that cascading is not allowed in VNPs inside an area, but only across areas. Any changes in the VNP path are immediately reflected in the determination of the MFS. Hence, the calculation of MFS for VNPs inside an area is very efficient. However, the method for calculating the MFS of an inter-domain VNP remains the same as explained in Section 3.8.

Whenever a new SPF tree is generated, due to routing updates, the edge nodes calculate the MFS of VNPs independently just by consulting the new SPF tree. This can be done for all the VNPs that originate at the edge node, or just those for whom VNP-path has changed.

3.9.2 Using IP Routing Table

Using link-state database to setup VNPs has many advantages. It does not need Layer-3 addressing to identify ports. It is possible to create a completely independent VNP layer, which any layer-3 can use. Physical links are part of the database; hence they can be identified and addressed easily.

However, not every autonomous system uses link-state routing protocols. Many autonomous systems use vector-distance IGPs like RIP and EGRIP. Vector-distance protocols exchange connectivity and proximity information between routers and networks. They do not exchange information regarding physical links. Hence, it is not possible to identify physical ports from the information base collected through vector-distance protocols.

The AS can be still divided into VNP-domains to reduce the number of VNP entries on the core nodes. However, the edge nodes cannot derive information about external destinations from the routing table. Hence, the VNP Edge Protocol is needed in this case, unlike with link-state database. The methods of dealing with routing updates and MFS changes remain the same as described in sections 3.6 and 3.8.

3.10 Explicit Path

The VNP paths considered thus far were implicit i.e. they were derived from the network topology through link-state database or routing table. They are the shortest paths, as per the routing protocol, to the destination. Any change in the topology changes the path. They 'follow' IP routes. A VNP explicit path may not follow the paths set by routing protocols. The hops along such a path are specified explicitly, and they may not change with the topology. A *strict* explicit path specifies all the nodes along a path in an order; the VNP path must cover all the nodes, in the order, and must not have any other nodes. A *loose* explicit path specifies certain intermediary nodes along the path in an order; the VNP path must cover all the nodes in the order but it may traverse other nodes. The specification of an explicit path constitutes the type - loose or strict - of the path and the intermediate nodes in an order.

The procedure for setting up of explicit VNP paths is similar to the one for implicit paths. The path specification is given as *Constraint*. The **NextHopSet()** procedure determines the next-hop according to the path specification. A new VNP identifier is generated for each invocation of the procedure, unlike for implicit paths where the identifier is associated with the destination.

3.11 Constraint-based Path

Routing is the process of finding paths among nodes, under certain constraints. The constraint in common routing protocols is to find the shortest path between two nodes, whereas the constraint in an explicit route is the nodes that can be traversed. Other constraints, like bandwidth and delay, can also be imposed on the way a route is chosen. The way to setup a constrained VNP path is the same as shown in Figure 3-2. The set of next hops is dependent on the constraints. It is possible that a node cannot meet the constraints. In such case, the VNP is not setup. All the nodes need to keep track of resource allocations and check if the constraints can be satisfied. Another way is to have the ingress node reserve the resources first and then setup an explicit path.

Chapter 4

IMPLEMENTATION

4.1 VNP Frame format

The format of a VNP data frame is shown in Figure 4-1. A VNP frame has the VNP header followed by its *payload* i.e. the data. The payload can be data from an application, a packet from another layer, or another VNP frame. VNP header is designed to be generic so as to carry data from different protocol. But it is optimized to provide minimal overhead for IP traffic. Hence the VNP header is of variable length.

The *Port-ID* field in the header is the identifier of the port through which the packet needs to be delivered. The 2-bit PT (Payload Type) field in the header indicates the type of the load. It can have the values: *VNP*, *IP*, *Any* or *Unknown*. The value of VNP for PT indicates that the VNP frame contains another VNP frame. This value of PT is used for VNP cascading. When PT has the value *Any* the optional field *SNAP ID* indicates the packet type. The *SNAP ID* field has the same format as the IEEE Sub Network Access Protocol (SNAP) Identifier [31]. It is five octets long; the first three octets contain a value assigned to a particular organization; the remaining two octets identify a specific protocol defined by that organization.

The value of *IP* for PT indicates that the load is an IP packet. Even though SNAP covers IP protocol the reason for having a special value in PT for IP is efficiency. The bulk of traffic through VNPs is going to be IP packets, and SNAP identifier introduces an overhead of five octets per packet. The special value for PT saves this overhead. It is possible that certain non-IP applications also do not want to incur the overhead of SNAP identifier. In such cases the egress node will associate the payload type with the VNP. Such VNPs are application specific and carry data only for that application. For such traffic PT is set to *Unknown.*

The VNP header has at least 4 octets but it can be longer depending on whether it is carrying the destination address and the load type. A multi-access VNP requires a destination address whereas a point-to-point VNP does not. The optional *Destination Address* field in the VNP header indicates this address. The destination-address field is of variable length. It has a SNAP identifier followed by the actual address. The SNAP identifier indicates the protocol for the address because the destination address can belong to various protocols. The SNAP identifier in the destination address field is not required for IP addresses. The 1-bit DAP (Destination Address Protocol) field in the header indicates whether the destination address is an IP address field or not. The SNAP identifier for the destination address protocol may not be the same as the SNAP identifier for the payload.

The six-bit Time-To-Live (TTL) field guards against wastage of resources in case of routing loops. The TTL field is decremented by one at each hop; the packet is discarded when it reaches zero. A 3-bit field represents the traffic class for the packet. The variable length of a VNP header does not hinder the forwarding speed because most of the nodes just swap the port-ID and forward the packet. Only the egress nodes need to look at the optional fields.

4.2 Traffic Classes

Virtual network ports support traffic classes. A 3-bit traffic class can be specified along with the packet for transmission over a VNP. A value of '0' for class indicates best-effort traffic and '7' indicates the highest priority. If the local physical port L_P supports traffic classes, the 3-bit quantity is converted to an "equivalent" for the local port and the packet transmitted with this equivalent class. Different physical ports may provide different types of traffic classification so the conversion of the VNP traffic class to the equivalent physical class is dependent on the node and the physical port. The 3-bit class is also put in the header of the VNP frame so that all the hops in the path can classify the packet using this field. If the underlying physical ports do not provide for traffic classes, the VNP sends the packets as best-effort traffic. It is possible to implement Weighted Fair Queues (WFQ) [40] in such cases (in the DevXmit function introduced later in this chapter) but at the cost of degrading the forwarding rates. WFQs can also be implemented as control procedures for certain VNP to provide classes on selected basis.

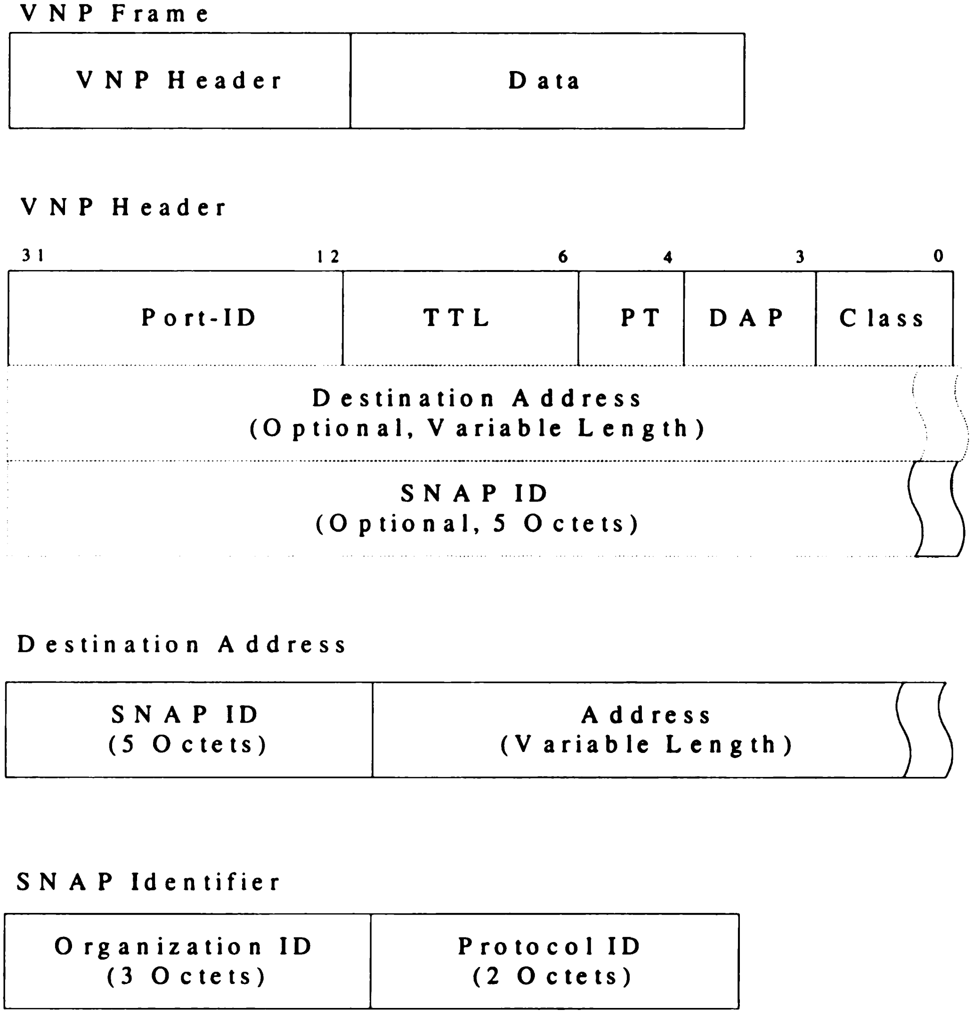


Figure 4-1 VNP Frame Format

4.3 VNP Software Components

Figure 4-2 shows the organization of the VNP-layer in the TCP/IP layered architecture. It falls between Layer-3 and Layer-2 of OSI's 7-layer architecture. But it is difficult to classify the VNP-layer among the OSI's 7 layers based on its functionality. It is used for inter-networking (network layer) but does not have its own addressing; it uses the addressing mechanisms of other network layers. At the same time, it can be used for bridging. VNP control procedures can also provide higher-layer functionality like encryption.

There are two major components in the VNP layer: VNP Packet Handler and VNP Transmit Procedure. VNP Transmit Procedure is VNP's "device driver" whereas VNP Frame Handler is its "protocol processor". A VNP uses the Transmit Procedure to forward data. The Frame Handler processes any VNP frames that arrive at a node. Figure 4-3 shows the flow of IP datagrams through these software components on two nodes connected through a VNP. In the figure, the ovals represent software components. VTP is the Transmit Procedure; VPH is the Packet Handler.

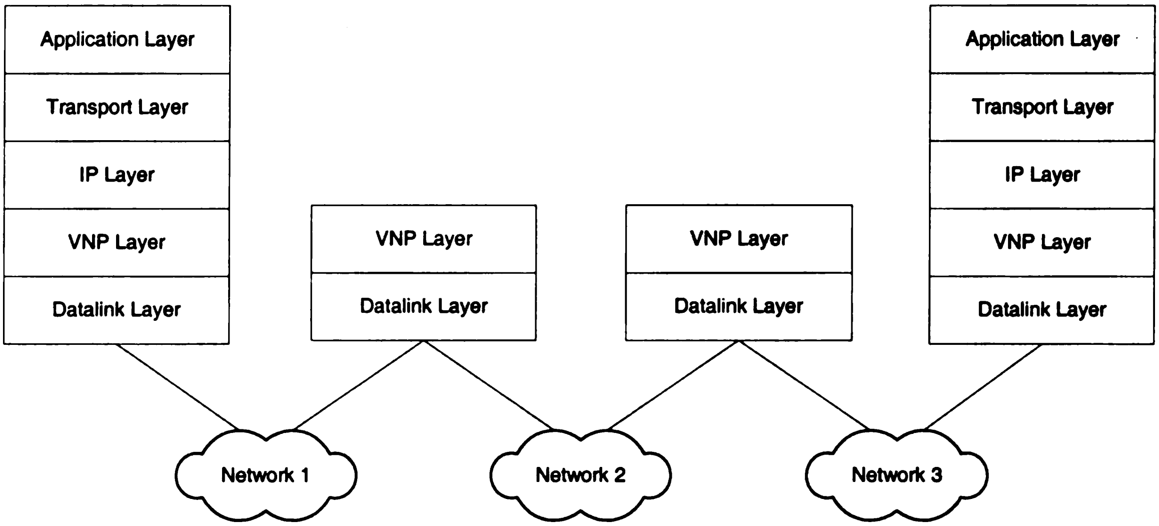


Figure 4-2 VNP Layer

Datagrams headed for the network connected to N_2P_2 arrive at node N_1 at Layer-2 (L2 in the figure). They are multiplexed by L2 over to IP layer. From its routing table, IP finds that the datagrams are to be sent through the virtual port N_1P_2. It hands them over to the Transmit Procedure (VTP) of N_1P_2. After consulting N_1P_2's path-set, VTP invokes the device driver for N_1P_1 to send the datagrams to N_2. When the datagrams arrive at N_2 they are multiplexed over to the VNP Packet Handler (VPH). VPH looks at the port-ID in the VNP header (it will be N_2P_2), and transmits the datagrams by invoking N_2P_2's device driver.

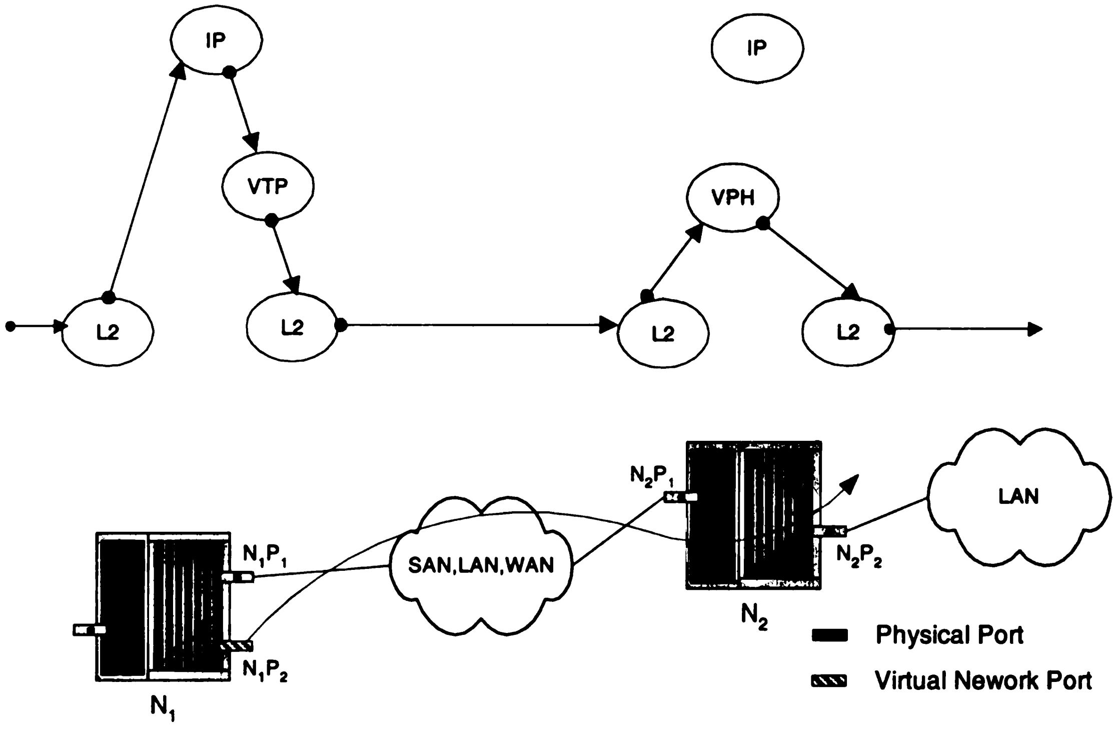


Figure 4-3 VNP Software Components

4.3.1 Packet Handler

Pseudo-code for VNP packet handler (VPH) is shown in shown in Figure 4-4. It looks at the port-ID in the VNP header. If it is a virtual port then the frame is handed over to the *Switch* function corresponding to the port. If the port is a loopback or a physical port then the payload is given to the port for transmission. For transmission at link layer, the payload type must be identified; this is determined by looking at the PT field in the VNP header. If the port in the incoming frame is invalid then the packet is dropped.

```
VNPPacketHandler(Packet) {
    switch Type(Packet.Head.Port) in {
        case LOOPBACK:
        case PHYSICAL:
            Header = Packet.Head;
            Packet = Packet – Packet.Head; /* Remove Top Header */
            PacketType = LoadType(Header);
            DevXmit(Header.Port, Header.Address, Header.Class, PacketType, Packet);
            break;
        case VIRTUAL:
            Call the Switch function corresponding to the Port Type;
            break;
        default:
            Drop the packet;
            break;
    }
}
```

Figure 4-4 VNP Packet Handler

4.3.2 DevXmit

DevXmit, shown as Figure 4-5, is the function to transmit data through any port. All layers including IP and VNP use it for transmission through a port. For a virtual port, it calls the *Cascade* function corresponding to the port. For the loopback port it passes the packets over to the appropriate layer. For a physical port, it checks if the target is the local node, in which case the packet is sent to Layer-3. Otherwise it calls the transmit

```
DevXmit(Port, Address, Class, PktType, Packet)
    switch Type(Port) in {
        case LOOPBACK:
            if ( PktType == VNP ) then
                VNPPacketHandler(Packet);
            else
                NetPacketHandler(PktType, Packet); // Send to upper layers
            break;

        case PHYSICAL:
            if ( Address == NULL )
                ReportError() ;
            else if ( Address == LocalNode ) then
                NetPacketHandler(PktType, Packet); // Send to upper layers
            else {
                Resolve Address if needed;
                Convert Class to an equivalent for Port;
                Call Xmit function corresponding to the Port Type;
            };
            break;

        case VIRTUAL:
            Call the Cascade function corresponding to the Port Type;
            break;

        default:
          Drop the packet;
          break;
    }
}
```

Figure 4-5 Generic Transmit Function

function corresponding to the port. Before that it converts the traffic class for the packet to the equivalent for that physical port.

4.3.3 Transmit Procedure

Corresponding to each port type there is a control procedure. For a physical port the control procedure is the device driver that transmits a packet through the port. A VNP control procedure has two transmission functions: Switch and Cascade. The control

```
MultiCastSwitch(Packet) {
    for Path in VNPTable[Packet.Port].PathSet {
        Packet.Port = Path.$R_P$;
        DevXmit(Path.$L_P$, Path.$A_N$, Packet.Class, VNP, Packet);
    }
}

MultiCastCascade(Port, Address, Class, PktType, Packet) {
    VNPHeader  Header;
    Header.Class = Class;
    Header.DAP = Address.Type == IP;
    SetLoadType(Header, PktType); // Set PT field

    for Path in VNPTable[Port].PathSet  {
        Header.Port = Path.$R_P$;
        If (Address != null) SetDestAddress(Header,Address);
        NewPacket = Header + Packet;
        DevXmit(Path.$L_P$, Path.$A_N$, Class, VNP, NewPacket)
    }
}
```

Figure 4-6 Multi-cast Control Procedure

procedure for a multicast type VNP is shown in Figure 4-6. For each path in the path-set of the incoming port, the Switch function swaps it with the remote port (R_P), and transmits the frame through the local port (L_P) to the next hop (A_N). The cascade function forms a new header, attaches it to the top of the packet and then transmits the packet through local port to the next-hop. For multicast VNP all the paths in the path set are used and the packet is left as-is. For other types of VNPs only a subset of the path-set may get used; the packet may be modified (e.g. encrypted), or the traffic class may be changed.

4.4 Prototype

A prototype for the framework was implemented on Pentium PCs running Linux. The virtual network port's software components, VPH and VTP, were implemented as one Linux loadable kernel module. The Linux ifconfig(8) command was used to set the attributes like MTU for the virtual ports. Another program, rnconfig, was developed to set other attributes of a virtual port. The module provides, through *ioctl* function, for modifying and retrieving the various attributes of the virtual port. The VNP table is also configured using the rnconfig command. The maximum number of VNP devices that can be configured is a parameter to the loadable module. The frame demultiplexor portion of the Linux network kernel invokes the VNP packet handler. Hence, no changes to the Linux kernel sources were needed. The virtual ports were configured manually. The implementation of VNP setup protocol is under way. Linux's network kernel has features of adding new type of packet handlers, so no changes were required for the kernel. No changes were made to the IP layer either. Only multicast VNPs were implemented; implementation of other types of VNPs is underway.

Chapter 5

DESIGN OF A SCALABLE ROUTER

The very first application of virtual network ports was in the design of a scalable IP router. VNPs were used to cluster off-the-shelf forwarding engines interconnected with commodity high-speed switching fabric. Packets into the router would go through at least two forwarding engines; and VNPs were used to avoid this multiple Layer-3 processing.

5.1 Architecture

The rapid growth in network data and bandwidths has put the focus on scalability as an important aspect of a router's features. It has been observed that a tier-1 router's performance needs to double every 10 months [41]. Router scalability has been mostly limited to forklift upgrades i.e. discarding the old router and buying a new one. This is due to the limitations in router architectures. Scalability requires a fundamental change in the design of a router.

There has been an emergence of very high speed, economical and scalable SAN/LAN interconnects, such as Myrinet [7] and ServerNet [28]. These interconnects provide bandwidths of gigabits per second. For example, a Myrinet link composed of a full duplex pair of channels provides a bandwidth of 1.28 Gbps. These switches can be incrementally expanded to provide higher aggregate throughput and thus scalability. A Myrinet multi-port switch can be connected to host computers or other Myrinet switches, resulting in a larger configuration.

The high bandwidth switching interconnects are used with off-the-shelf processors to architect a highly scalable, high-throughput IP router. Figure 5-1 shows the architecture of the proposed scalable router. It comprises of a set of autonomous *routing nodes* connected by one or more high-speed switches. The routing nodes are general-purpose computers. Each node has one or more of internal and external network interfaces. The *internal network interfaces* attach to the switching fabric, and the *external network interfaces* attach to LAN or WAN networks. The external network interfaces of different routing nodes may be attached to different networks or to the same network. Packets enter the router through the external network interfaces of the routing nodes. A node may have multiple internal interfaces for higher throughput or fault tolerance. The switching fabric may have multiple parallel paths from *Node-1* to *Node-2* through independent switches.

The routing nodes route the input packets in parallel through the switching fabric to the outbound routing nodes. The outbound nodes switch the packets out on to the destination network without going through the IP layer. The architecture provides for high degree of scalability. If more ports are required, more routing nodes or external network interfaces can be added. More internal network interfaces or switches can be added to handle higher aggregate bandwidth. High-end multiprocessors can be used as routing nodes to accommodate larger bandwidth, such as OC-48 and above, at external network interfaces.

The nodes of the proposed router use virtual network ports to work as a unit. Each node in the router defines virtual network ports to the external network interfaces of all the other nodes. An IP packet enters the router through one of the routing nodes. This ingress node performs traditional IP routing[5] to find the target address (the next hop address or the destination address) for this packet.

It then hands over the IP packet with target address to appropriate the VNP device. The VNP device sends the packet through the egress port on the egress node to the target address. The packet does not go through the IP layer at the egress node. The

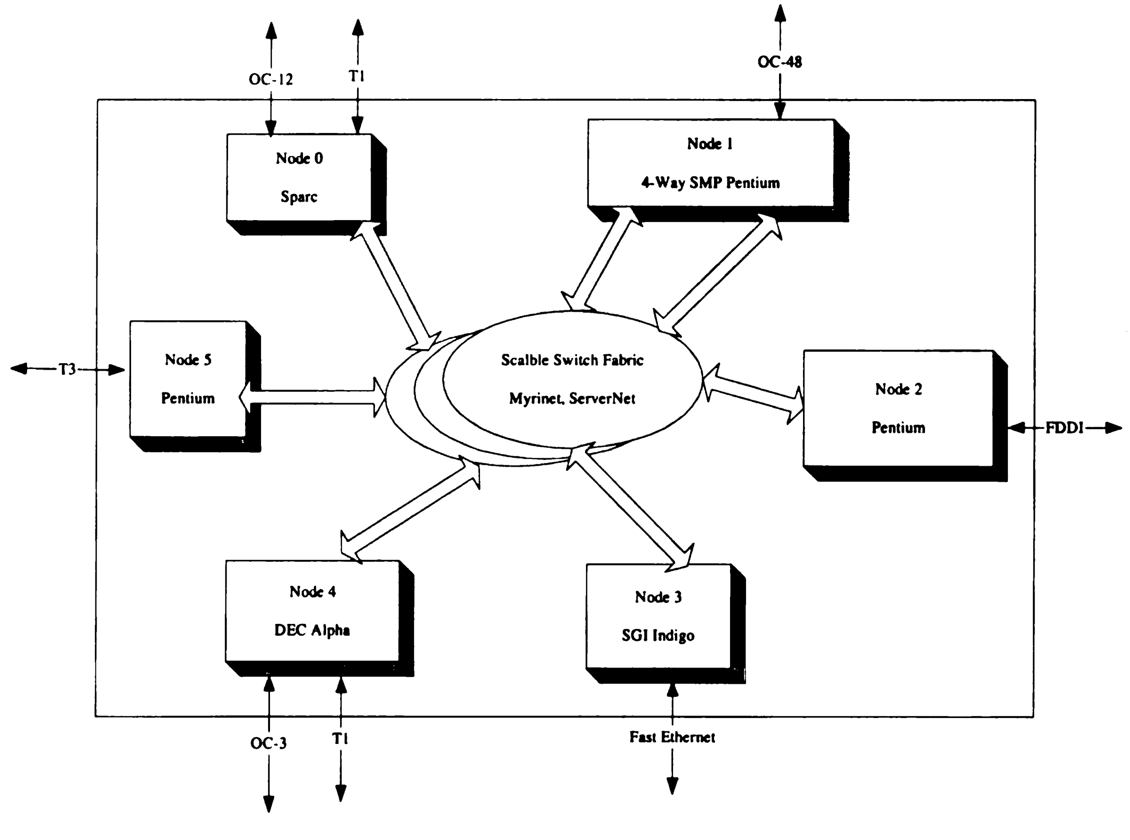


Figure 5-1 Block Diagram of the Proposed Router

[5] Unless the packet is coming from a VNP enabled node, in which case the packet does not go through IP

ingress node acts like a traditional router and the egress node similar to a switch. Note that routing and fragmentation is done at the ingress node, and ARP is done at the egress node.

The information about the egress node, the egress port, and the internal network interface can be incorporated at various layers. For instance, it can be made part of the IP routing table, but that will involve changes to IP layer. Instead we use VNP devices for keeping this information. It has the advantage of using the intelligence of the IP layer to discover information about the egress node but without any changes to the IP layer.

5.2 Prototype

A prototype router was implemented using four Pentium PCs running Linux interconnected through a four-port Myrinet switch. The prototype VNP implementation, described in Chapter 4, was used for setting up VNPs. The VNPs were configured manually. The device driver for Myrinet was slightly modified to handle VNP frames.

5.3 Performance Measurement

The proposed router architecture has two stages: the ingress stage and the egress stage. This separation of stages provides scalability but may affect the performance. So we looked at the effects of the egress stage on the performance rather than the overall performance of the router. It was found that this separation of stages did not have any

layer.

effect on the throughput or frame loss rate but resulted in less than 5 percent increase in latency.

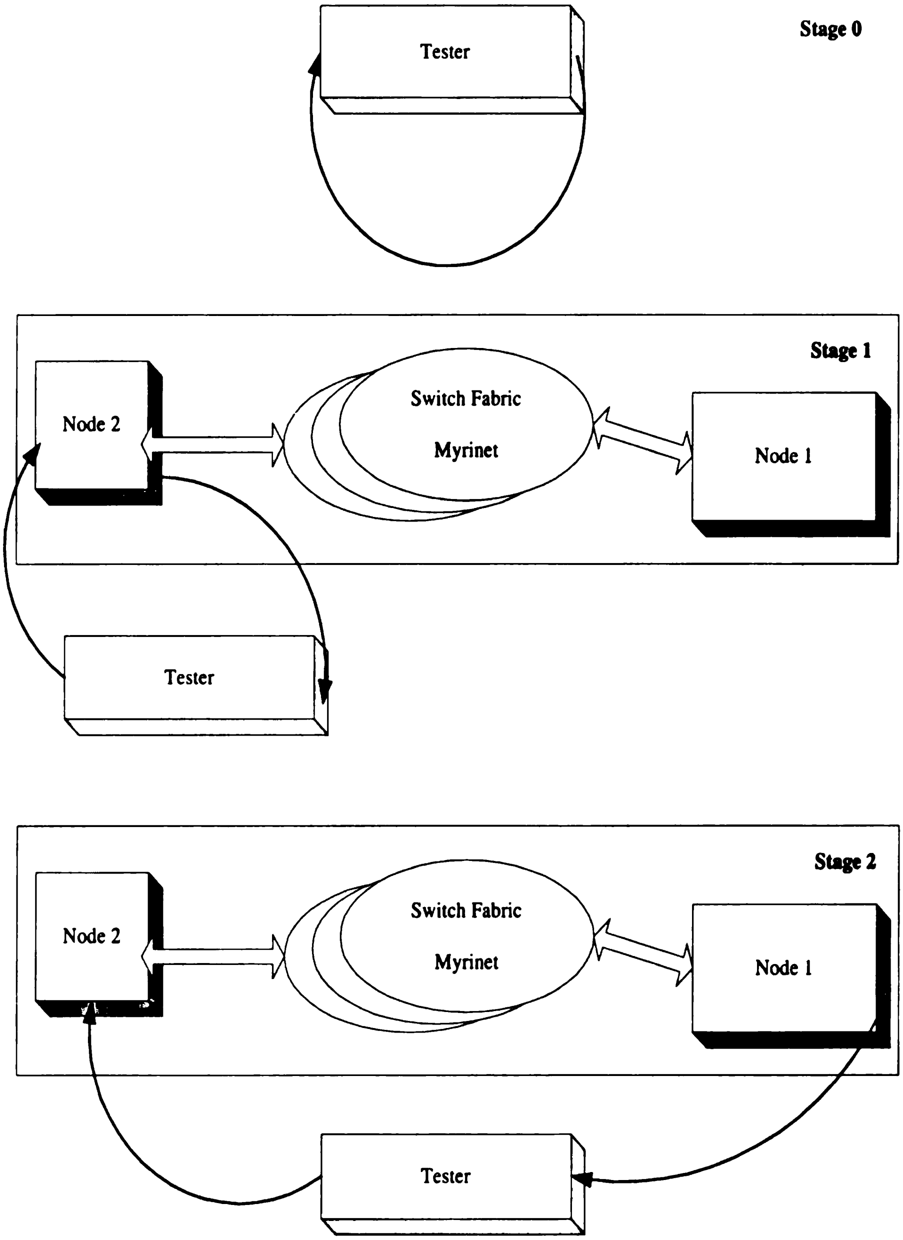


Figure 5-2 Test Configurations

The various benchmarking test setup configurations are shown in Figure 5-2. The Tester is used to generate network frames for the benchmarks. The reason for the three configurations, Stage-0, Stage-1, and Stage-2 is to isolate the effects of the Tester, ingress node, and egress node respectively. In Stage-2 the Device Under Test (the prototype router) is connected to the transmitting and receiving ports of the Tester. In Stage-1 only the ingress node is connected to the Tester whereas in Stage-0 the transmitting and receiving ports of the Tester are connected directly. Any limitations of the Tester become visible in the Stage-0 configuration; it was found that the Tester used could not generate frames at the theoretical maximum for small frame sizes. In the Stage-1 configuration effects of the IP implementation become evident. It was found that a major portion of the latency was due to the ingress node. The Stage-2 configuration shows the overhead due to the egress stage and the remote-network device. All the benchmarks were run for these three configurations. A two-node router was used for these tests. A SparcStation-20 running Solaris 2.5.1 was used as the Tester. Only 10Mb/s Ethernet ports were used for the benchmarks. Null (port 9) UDP packets were used for all the tests. Methodologies specified in RFC 1944 [11] were followed for performing the benchmarks.

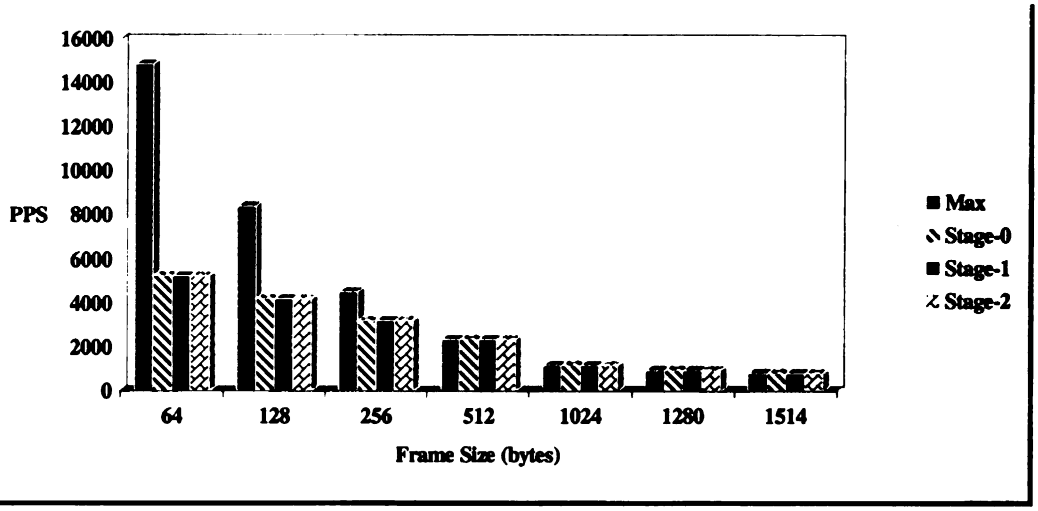


Figure 5-3 Throughput

5.3.1 Throughput

Throughput is defined as the maximum rate at which none of the offered frames is dropped by the device. Throughput for various test configurations and the theoretical frame rates for 10Mb/s Ethernet are shown in Figure 5-3. It is seen that the egress stage of the router does not have any impact on the throughput of the router. Note that the Tester could not generate frames at the theoretical frame rate for smaller frame sizes. Frames of size 1518 octets were getting fragmented and hence 1514 octet sized frames were used.

5.3.2 Frame Loss Rate

Frame loss rate is defined as the percentage of frames, that should have been forwarded by a network device under steady state load, that were not forwarded due to lack of resources [12]. It was measured for frame sizes of 64, 128, 256, 512, 1024, 1280 and 1514 octets. No frame loss was noticed for all the test configurations.

5.3.3 Latency

Latency for store and forward devices is the time interval starting when the last bit of the input frame reaches the input port, and ending when the first bit of the output frame is seen on the output port [12]. The measurements shown in Figure 5-4 do not show the latency as defined above. Instead it shows the difference between the time the frame was transmitted, and then received by the Tester. This includes the transmission times on the media. Note that Stage-1 and Stage-2 involve an extra leg of transmission delay compared to Stage-0. It is seen that there is less than 5 percent increase in this latency due to the egress stage of the router. A good part of this increase is suspected on the misaligned Linux network buffers in the implementation of VNP device.

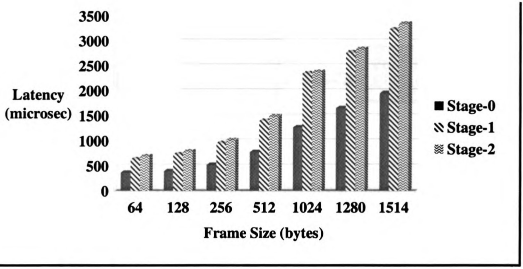


Figure 5-4 Latency

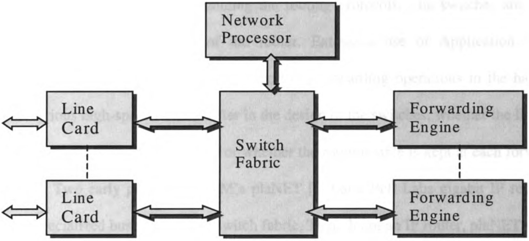


Figure 5-5 Architecture of a High Speed Router

5.4 Related Work

5.4.1 High Speed Routers

Earlier router architectures were bus-based. They had a single processor that was involved in forwarding and protocol processing. The network interfaces and the processor were connected through a bus. It was not possible to scale the throughput due to the bottlenecks of the processor and the bus. Recent architectures incorporate multiple forwarding processors and protocol processors interconnected with the network interfaces through one or more gigabit switches. Such a router has four components, line cards, forwarding engines, network processors, and switching interconnects. The line card is the interface to external data links. The forwarding engine inspects packet headers and determines the outgoing line card to send the packets through. The network processors compute the routing table by running the routing protocols. The switches are used to interconnect the components of the router. Extensive use of Application Specific Integrated Circuits (ASIC) is made to perform forwarding operations in the hardware. The various high-speed routers differ in the design of the switches, whether the line cards also act as the forwarding engines or whether the routing table is kept at each forwarding engine. Two early prototypes, IBM's plaNET [14] and Bell Labs gigabit IP router [6], used specialized buses instead of switch fabric. Though not an IP router, plaNET was one of the first gigabit switches and supported a number of routing schemes [40]. Recently there have been a number of commercial implementations. Ascend's GRF IP Switch, Cisco's GSR [15], BBN's Multi-Gigabit Router [41][42], Torrent's IP9000 [53] and Pluris' Massively Parallel Router [44] are some such examples. Tiny Tera [33] illustrates the design of high-performance switches for high-speed routers.

5.4.2 Fast Lookup Methods

Classless Inter-Domain Routing [22] was introduced in the early 1990s to curtail the growth of routing table entries. It required the longest prefix match of routing table entries to determine the next-hop. This has turned out be the most compute expensive part of forwarding. Initial implementations of longest prefix match used radix tries [53]. Content Addressable Memory (CAM) was used in [30] to speed up the matching process through hardware. Caching has been shown to improve the lookup performance in [43]. Recently, an innovative data structure for collapsing the routing table into a very small forwarding table was proposed in [17]. The forwarding base was small enough to fit into the cache of a general-purpose processor to result in several million lookups per second. Another new lookup algorithm using binary search on hash tables organized by prefix lengths is described in [63]. With this approach, in the worst case, the number of lookups required is $\log_2$ of the address length in bits. Torrent's ASIK algorithm [57] is claimed to be very fast but its details have not been made available.

Chapter 6

MPLS VS VNP

Switching techniques forward packets below Layer-3. They do this by establishing data-link switched paths between ingress and egress nodes over the underlying hardware. This is done either dynamically by identifying flows in the traffic, or statically based on network topology, or both. At ingress node the incoming packet is mapped to a switched path (or tree in case of multicast) based on certain forwarding criteria. From then on the packet is forwarded along the switched path to the egress node, below Layer-3. The intermediate nodes along the path perform ATM-like label swapping to forward the packets. Ipsilon's IP-Switching [37][38][39], Cisco's Tag-Switching [16]. IBM's Aggregate Route-Based IP Switching (ARIS) [58], Toshiba's Cell Switch Router (CSR) [29], NEC's IPSOFACTO [1] and Cascade's IP-Navigator [11] are some representative switching techniques. They mostly differ in their protocols for setting up the switched paths, the granularity of flows they support, and the underlying hardware they work on. IETF's Multi Protocol Label Switching (MPLS) Working Group [47] is currently drafting a specification for a label-switching standard. A good comparison of the various techniques is described in [20]. Our approach is conceptually very different from all the other techniques but is close to MPLS in implementation. Therefore we now describe MPLS in a little detail as per [47].

6.1 MPLS

A *label* is a short, fixed length locally significant identifier. A router that supports MPLS is called a Label Switch Router (LSR). A set of contiguous LSRs under the same administration defines a MPLS domain. A Label Switched Path (LSP) corresponding to a *class* of packets is a sequence of LSRs <R_1, R_2,..., R_n> such that

- Each LSR R_i, $1 < i \leq n$, allocates a label L_i for the *class*
- Each LSR R_i, $1 \leq i < n$, is made known of the label (L_{i+1}) of its downstream LSR R_{i+1}.
- The ingress LSR R_1 checks if a packet belongs to the *class*. If so it attaches the label L_2 to the packet and forwards it to R_2.

Classification of packets can be based on their network destination, domain egress node or on some other criteria. MPLS forwarding works this way: when a packet labeled with L_i arrives at R_i, it replaces L_i with L_{i+1} and forwards the packet to R_{i+1}. A packet may have a stack of labels attached to it but the forwarding is based exclusively on the label at the top of the stack. An LSR maintains three data structures for the purpose. A Next Hop Label Forwarding Entry (NHLFE) contains information about the next hop and label stack operations. The Forwarding Equivalence Class to NHLFE (FTN) maps packet classes to NHLFEs. The Incoming Label Map (ILM) maps the incoming labels to NHLFEs. Each NHLFE contains the following information,

- The next hop for the packet

- The data link encapsulation to use when transmitting the packet
- The way to encode the label stack when transmitting the packet
- The operation to perform on the packet's label stack, it is one of the following operations
  - Swap the label at the top with a specified new label
  - Pop the label at the top
  - Swap the label at the top with a new label, and then push one or more specified new labels onto the label stack.

The LSRs use Label Distribution Protocol (LDP) to distribute information about the labels and to coordinate stack operations.

6.2 Comparison

The use of high performance processors and special hardware in high-speed routers makes them expensive. Even though improvements in table lookup methods are impressive they are not as fast as the switching techniques. Switching techniques involve a simple exact-match table lookup and hence provide very high forwarding speeds. They can be implemented in the hardware and when implemented over switching fabrics like ATM, they provide forwarding at physical line speeds. But switching techniques do not scale well with the number of flows in the network. They need to keep track of the flows through a router that has limited resources. Switching techniques require a set of routers to run the switching protocol, whereas high-speed routers and fast lookup methods improve the performance of individual routers and do not require any new protocols. Switching techniques cannot completely eliminate the Layer-3 processing either. Ingress, egress and aggregate routers still require packets to go through the network layer. Packets that do not have the flow setup for them or have invalid labels will also need to go through Layer-3. So the advantages of switching techniques over the other two techniques for forwarding are debatable [35]. But most of the switching techniques have the added benefits of identifying flows, explicit routing, and layer-2 tunneling. This helps in providing for quality of service, traffic engineering and protocol independent tunnels. Hence the switching techniques have an edge over the other two techniques in this regard. The scaling issue in switching techniques is tackled by providing coarse-grained aggregated flows. We envision all the three being deployed on an inter-network and even on the same router.

Using virtual network ports seem to involve a lot more processing and memory than label switching techniques like MPLS. Looking at the procedures in Chapter 4 the only extra processing needed, when compared to label-switching techniques, is determining whether the port is loop-back, physical, or virtual. Having a fixed maximum for physical port identifiers (P_{max}) can do this very efficiently. A node by design cannot have more than a certain number of physical ports, and this can be the fixed maximum. The more complex VNP header format does not affect the forwarding speed either. Only the end-nodes look at the various fields in the packet, the intermediate nodes just look at the port-ID, and forward the packet. The generic control procedures provide flow processing, and should not be used at nodes where forwarding speed is desired. They are to be used on nodes with adequate processing capabilities. VNPs need more memory compared to labels for their attributes like MTU but do away with the stack operations. These attributes provide extra functionality compared to labels, and can also be cached.

VNPs can calculate the attributes when needed; only the often-used ports will have the attributes in the memory.

Our technique is a more generalized approach than MPLS. Using only point-to-point VNPs and multicast control procedure results in more than the functionality of MPLS. VNPs act as flows to networks and not just nodes. This facilitates single-hop paths to the destination. Conceptually VNPs are easier to discern than labels. Labels do not define the ends (ingress and egress nodes) of a flow clearly, whereas VNPs do. This makes label stacking implicit and efficient. The pop or push operations on labels are explicitly listed in the NHLFE entry of MPLS whereas in VNP they are not. The pop operation does not require a lookup into the VNP table whereas in MPLS the NHLFE entry needs to be looked up to perform any stack operation. Even though the implicit NULL label and penultimate hop popping of MPLS make the pop operation equally efficient, it is not possible on all LSRs.

If the MTU of the outgoing port is smaller than the packet size, MPLS needs to fragment the packet [49]. Since VNPs have the smallest MTU of the whole path associated with them, fragmentation is not an issue. Moreover the MTU value is the optimal one for multicast VNPs. A VNP is like any other physical network port on the node so the interface between higher layers and VNPs is already there. For destination based VNP flows, no change is needed to the IP layer at all.

Whereas labels are passive virtual ports are active, they do a lot more than forwarding. They can filter, police, shape, encrypt or do almost anything with the traffic flow. This helps with per-flow processing of packets at each node. Label switching fails at route aggregates. The performance at aggregate routers can be improved with the special aggregate VNP control procedures.

6.3 Quantitative Analysis Using Simulation

We simulated a network of MPLS and VNP nodes to analyze the overhead incurred by VNP. We measured the times taken for the three basic operations involved: switch, aggregate, and de-aggregate. For VNP, the corresponding operations are: switch, cascade, and un-cascade; for MPLS, they are: switch, push, and pop respectively. The simulator was written in C. All the experiments were run on an SGI Origin-200 with R10000 processor and 512MB of RAM.

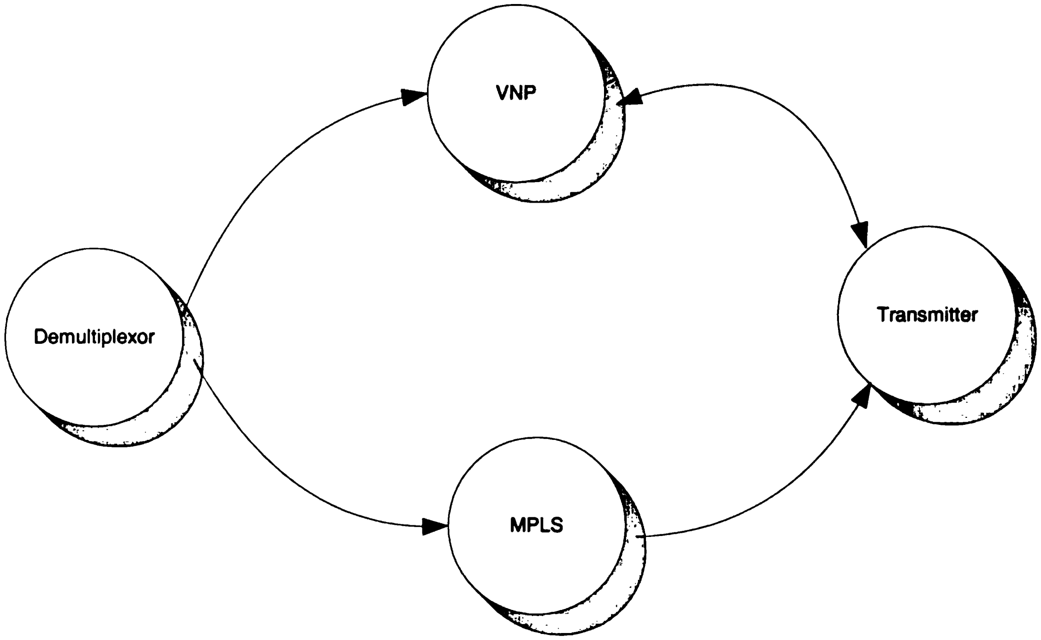


Figure 6-1 Simulation Modules

6.3.1 Architecture

Each node in the simulator has four modules: Multiplexor, VNP, MPLS, and Transmitter. This is illustrated in Figure 6-1. The Demultiplexor receives frames coming into a node from a physical device. It checks the protocol field in the frame, and passes the frame to the corresponding protocol handler. In our case, there are only two protocol handlers: VNP and MPLS. The protocol handler modifies the frame, and gives it to the Transmitter. The Transmitter sends frames out of a node through a network device. Since a VNP is a network device too, the Transmitter may send the frame back to the VNP module. This happens in the case of cascading and un-cascading. However, frames from MPLS module are never sent to the VNP module because MPLS module always requests the frames to be sent through a physical device. The Transmitter handles VNP and MPLS frames differently; hence it needs to be notified of the frame type.

6.3.2 Methodology

The time taken for processing a frame is measured as the time the frame spends in the respective protocol handler module. The clock starts when a packet leaves the Demultiplexor, and stops when the packet enters the Transmitter. The VNP and MPLS

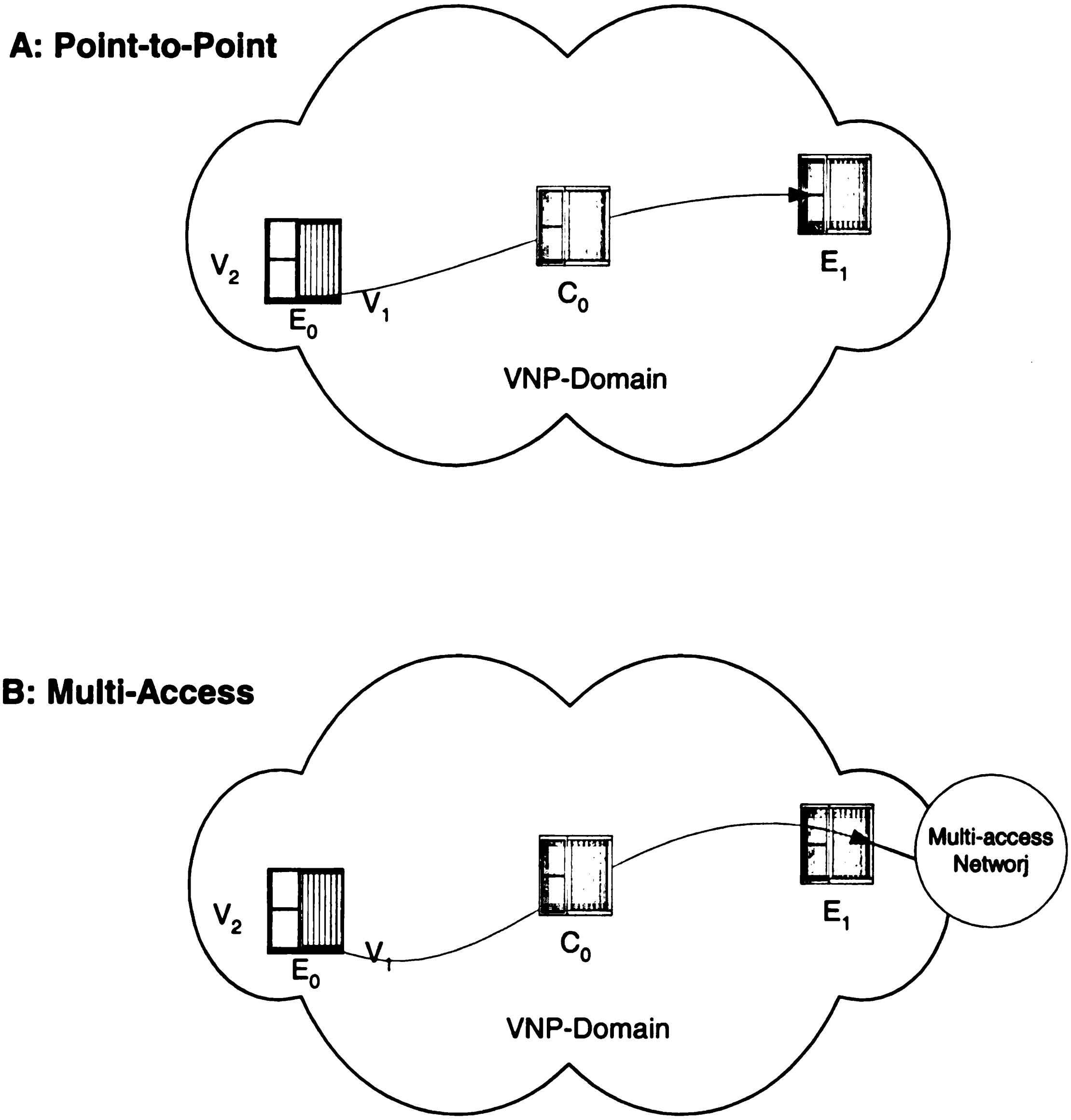


Figure 6-2 Simulation: Switching

tables were setup manually before the start of the experiments. For each experiment, two million packets were sent through the VNP and MPLS paths. The first million packets were for bringing the system to a steady state; the time taken to process the last one million packets was noted down.

6.3.3 Switching

In the first experiment, we compared the time taken to switch packets using MPLS and VNP. Figure 6-2 shows the network configurations we used for the simulation. Routers E_0 and E_1 are edge routers, and C_0 is a core router. VNP and MPLS paths were so setup that E_0 is the ingress node, C_0 an intermediate node, and E_1 the egress node. Packets were generated so that they enter at E_0, travel over the VNP or MPLS path, and exit at $E_{1.}$ We measured the following processing times.

1. The time to add VNP or MPLS headers at node E_0
2. The time to switch VNP or MPLS frames at node C_0.
3. The time to remove MPLS or VNP headers at node E_1

In the first configuration, as shown in part A of the figure, we used a point-to-point VNP.

6.3.3.1 Multi-access VNP

Multi-access VNPs help in reducing the number of VNPs in a node. They are useful in the core, where the number of flows can get prohibitively large. Multi-access VNP packets carry the next-hop address in their header, whereas point-to-point VNP packets do not; this may introduce a processing overhead on the nodes. We measured the times taken for each operation on both point-to-point and multi-access VNPs. The next-hop address in a multi-access VNP packet can be of any protocol including Layer-2 protocols. For our experiments, we used only IP next-hop addresses. For the second part of the

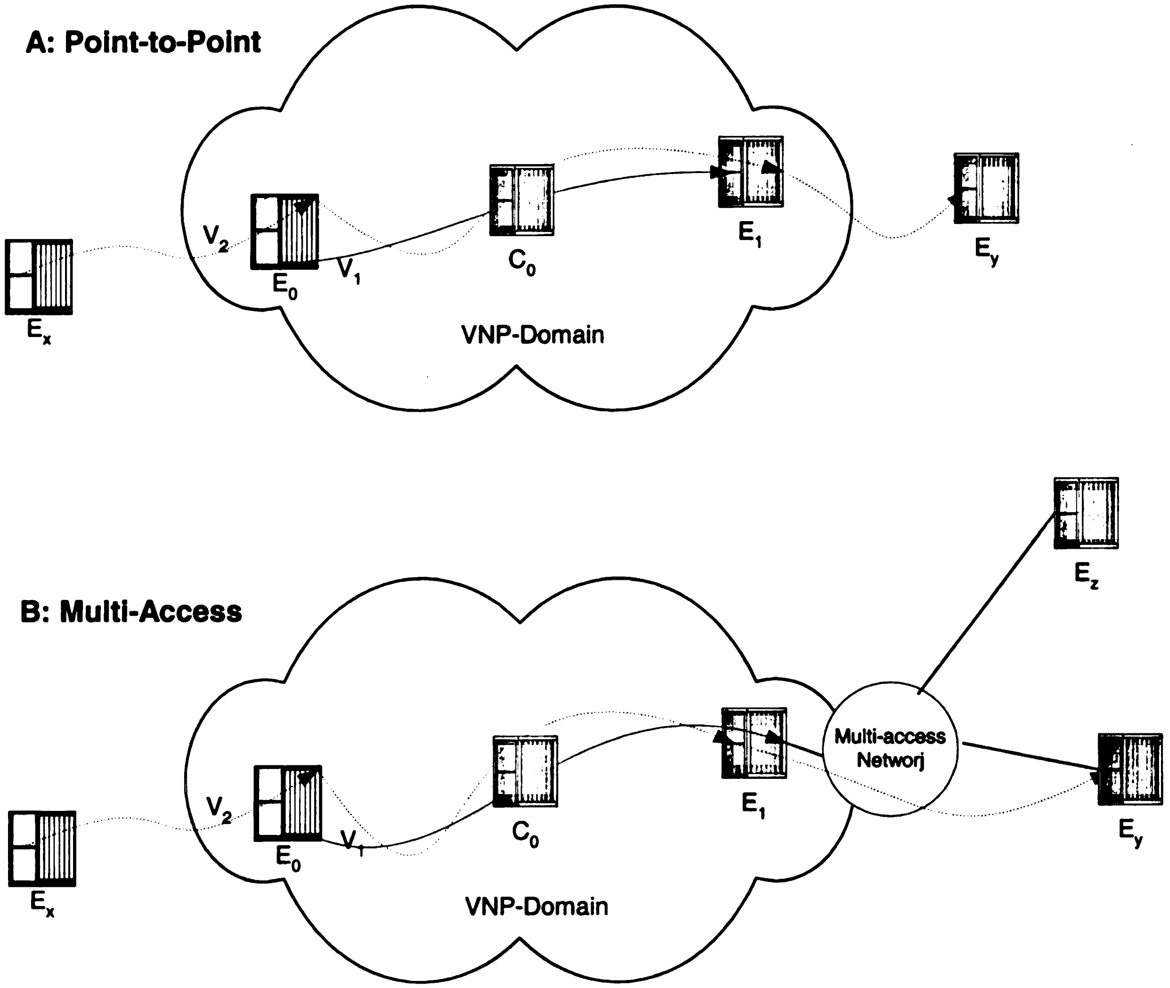


Figure 6-3 Simulation: Aggregation

experiment, as shown in part B of the figure, we used multi-access VNPs, The test packets were generated to have a node (not shown in the figure) on the multi-access network as the destination.

6.3.3.2 MPLS

MPLS paths are always between two nodes i.e. point-to-point, so the concept of multi-access is not applicable to them. However a set of LSPs, from the ingress node to each node on the multi-access network, can be used to provide the equivalent connectivity. This results in an increase in the number of LSPs on the nodes along the LSPs, but does not change the time taken for switch, push, or pop operations.

6.3.4 Aggregation and De-aggregation

Figure 6-3 shows the network configurations used for measuring the processing involved in aggregation and de-aggregation. It is similar to the two configurations used in switching. Nodes E_0, C_0, and E_1 belong to a VNP-domain. C_0 is a core router, whereas E_0 and E_1 are edge routers. V_1 is a point-to-point VNP from E_0 to E_1; it passes through C_0. E_x and E_y are two other edge routers; they do not belong to the same VNP domain as E_0, C_0, and E_1 do. V_2 is a point-to-point VNP from E_x to E_y; it is cascaded over V_1.

For MPLS, we created two LSPs: L_1 and L_2. L_1 is an LSP with E_0 as the ingress, and E_1 as the egress node, and has C_0 in its path. L_2 is an LSP with E_x as the ingress node, and E_y as the egress. MPLS tables on E_0 and E_1 were setup to push and pop labels respectively such that L_2 aggregates over L_1.

Packets were generated to enter E_x, travel over V_2 and L_2, and finally reach E_y. E_0 aggregates the packets over to V_1 and L_1 respectively; E_1 de-aggregates them back to V_2

and L_2. The core router C_0 is involved only in switching the packets. We measured the following processing times.

1. The time to aggregate V_2 and L_2 over V_1 and L_1 respectively, at E_0
2. The time to switch VNP or MPLS frames at node C_0.
3. The time to de-aggregate into V_2 and L_2 at node E_1

Table 6-1 Simulation Options

	VNP	**MPLS**
Operation	Switch	Switch
	Cascade	Push
	Un-cascade	Pop
	Add, Remove Header	Add, Remove Header
Access	Point-to-point, Multi-access	Point-to-point
Payload	IP, VNP, Any, Unknown	IP

6.3.5 Payload

A VNP can carry packets of different types. An MPLS path can carry packets of only one type, generally IP. To achieve the same functionality MPLS has to have multiple LSPs, one for each packet type. This results in enormous wastage of resources on each node along the LSP. However, this advantage of VNP comes at a cost; VNP frame has complex header format. We wanted to check if the complex header format results in a processing overhead.

In case of MPLS, only TCP/IP packets were used as the payload. However, we used four different payload configurations for point-to-point and multi-access VNPs. These configurations were:

1. **IP**: TCP/IP packets were used in the experiments
2. **VNP**: VNP frames were used as the payload (inside another VNP frame). This is always the case during cascading.
3. **Any**: Any type of payload can be used in this case. We chose IPX packets. It involved adding the SNAP identification to the header.
4. **Uknown**: This is similar to MPLS. VNP is unaware of the type of the payload it is carrying; only the end nodes know about it. Even though any type of packets can be used, X.25 packets were tested as the payload.

6.3.6 Results

In summary, we measured processing efforts for five types of operations, four types of payload, and two types of network access. These various options are shown in Table 6-1.

6.3.6.1 Switching

Figure 6-4 shows the processing times for the switching experiments. Switching times for point-to-point and multi-access VNP are the same, and are irrespective of the payload. This is simply because the switching operation involves only swapping the port IDs; it does not depend on anything else. Hence, VNP provides the advantage of low resource usage, through multi-type payload and multi-access ports, at no extra switching cost. We had expected this, even prior to the experiments. However, we were surprised that VNP switching was faster than MPLS switching. We expected both of them to be almost the same. The reason for this speedup is *operation identification.*

When a frame is given to the MPLS module, it has to determine what kind of operation –switch, push, pop– to perform on the frame. It extracts the label off the frame, and indexes into the MPLS table to identify the operation and the local port. It then forwards the frame through the local port. The VNP module also has to do the first part i.e. extracting the port ID off the frame. However, it identifies the operation just by looking at the port ID and the local port. This saves few memory references, and results in faster switching.

Processing efforts for the *Add Header* operation for VNP configurations look unremarkable. As expected, it takes longer for multi-access VNPs compared to point-to-point VNPs; payloads of type *Any* are more expensive than the others. This is due to the longer VNP frame headers required by mulit-access VNP and the *Any* type payloads.

The *Remove Header* operation for MPLS is very fast. The MPLS header is of fixed length. All that is needed is to offset the start pointer in the network packet buffer by four octets. The variable length of the VNP header requires looking at the packet and VNP table to determine the header size. This takes longer for multi-access VNP compared to point-to-point VNP. However, the payload type has no effect on this operation.

6.3.6.2 Aggregation

Figure 6-5 shows the processing times for aggregation, switching, and de-aggregation at the nodes E_0, C_0, and E_1 respectively of Figure 6-3. As expected, the processing times for switching, at C_0, are exactly the same in both the aggregation and switching experiments, for both MPLS and VNP. Hence, aggregation does not adversely affect the switching operation in any way.

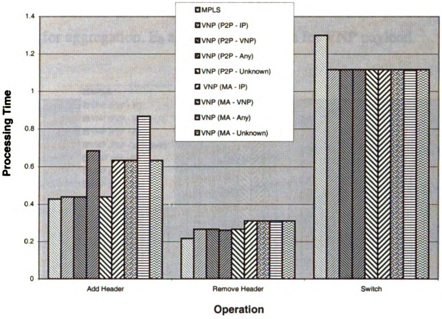


Figure 6-4 Simulation Results: Switching

The processing times for point-to-point VNP were lower than that for MPLS,. The aggregation operation involves adding headers, and we have seen that multi-access VNP takes longer to do it than MPLS. The reason for the discrepancy is operation identification, as was the case in switching. It takes longer for MPLS to identify the

operation, than point-to-point VNP to add headers. As expected, the processing times for multi-access VNP were higher than MPLS

The processing time is the same for all payload types for a given VNP. This seems to contradict the processing times for adding headers; apparently aggregation time for *Any* payload should be more. However, this is not the case because aggregation always generates frames with payloads of type VNP. When a VNP frame, irrespective of its payload type, reaches E_0 (Figure 6-3), it becomes the payload for the encapsulating frame. Hence, for aggregation, E_0 always adds headers for VNP payload.

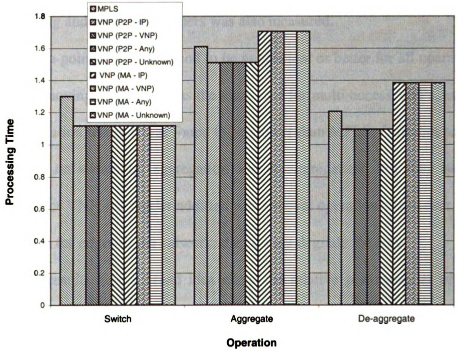


Figure 6-5 Simulation Results: Aggregation

The processing times for de-aggregation operation follow a similar pattern. De-aggregation mostly involves removing headers from frames. Point-to-point VNP takes less time for de-aggregation than MPLS because it takes longer for MPLS to identify the operation than VNP to remove headers. In the case of multi-access VNP, the remove header operation dominates the operation identification. Hence it takes longer for multi-access VNP to de-aggregate.

6.3.7 Summary

We simulated VNP and MPLS to estimate the overhead incurred by VNP due to its relatively complex nature. We used both point-to-point and multi-access VNPs. Four different types of payloads were used for VPN; MPLS was tested with a single payload type. Three major operations were compared: switch, aggregate, and de-aggregate. Time taken for adding and removing headers was also measured.

Point-to-point VNP was found to be as efficient or better for all operations except adding and removing headers. It was the same case for multi-access VNPs, except for de-aggregate operation where it was found to be slower than MPLS. Different payload types had no affect on switching, aggregation and de-aggregation. But they did adversely affect, in case of VNP, the header addition and removal operations. However such effects are confined to the edges of the network; they do not degrade the core.

These results give a broad idea about the relative performances of MPLS and VNP. They tell that VNP performance is comparable to MPLS, and sometimes even better; and that it is not drastically different from MPLS' performance. The actual processing times are dependent on a number of factors that the simulator did not and cannot take into account. For example, we did not try to align the packets to word-boundaries because we did not try to simulate the underlying hardware. In real life, packet alignment to word-boundaries plays a significant role in packet processing performance. Programming style or tricks can significantly alter the performance too, especially in our case where even a few memory accesses matter a lot. However, this in itself is an indication of how close VNP's performance is to MPLS'.

6.4 MPLS with VNP

Some of the ideas from VNP can be incorporated into MPLS without making a lot of changes in the MPLS specification, or its implementations. An LSR should consider certain labels to be special. Like the other labels these special labels have meaning only to the LSR. The special labels should be discernible from the others very quickly and efficiently. One way to do this is to have a node-dependent label value everything below which is considered special. If an LSR is an egress LSR for an LSP, it distributes only a special label up the LSP. So all the aggregate flows that terminate at an LSR always bring packets with the special labels into the LSR. The LSR always pops out the label at the top if it is a special label.

It should be possible to specify a local MPLS label as the data link transport in a NHLFE entry. This label is always pushed onto the incoming packets that map to the NHLFE entry. Then the packet is processed again according to the label on the top. These changes in MPLS eliminate the need for explicit push and pop operations in the NHLFE entries. An NHLFE entry with MPLS as data link implies push operation, and an incoming packet with a special label implies pop operation.

To eliminate fragmentation, a field named MFS is added to each NHLFE. It represents the maximum MPLS frame size that is allowed to transmit using a NHLFE. The rules for calculating it are the same as those for VNP except that the L_P is the local data link transport and R_P is the new swapping label in NHLFE. The MPLS and higher layers should treat (MFS – 4) as the NHLFE entry's "MTU". The LDP protocol needs to distribute the corresponding MFS value with each label.

Chapter 7

CONCLUSIONS

We have presented the preliminary work on virtual network ports. The concept originated with the design of a scalable router to help scale the forwarding performance. It later evolved into a connection-oriented switching methodology for addressing traffic engineering and service-guarantee needs. However, a number of issues are still under study.

7.1 Future Work

7.1.1 Generic Controls

We have looked at only unicast and multicast control procedures. A unicast remote-network device conceptualizes the basic flow of packets in an inter-network. It forwards packets along its path with no other processing. A *generic control procedure* on the other hand is not restricted in how it handles the in-coming packets. It may forward the packets selectively to the paths in its path-set, police the traffic or even may modify the packet contents. The path selection can be based on the fields in the packet, the cost of each path, or the service guarantees for the flow. A generic VNP is useful in traffic engineering, de-aggregation, and implementing service guarantees among other things.

To remote nodes, the generic VNPs appear just like regular multi-cast VNPs or physical ports. The special attributes of the generic VNPs are not propagated along the conduit. In a long VNP conduit there can be different type of VNPs, some may perform simple switching while the others may involve extensive packet processing.

7.1.1.1 Filter VNP

A filter VNP selectively forwards the packets to one or more of the paths in its path-set based on contents of the packet. If there are multiple physical paths to a destination, a filter VNP encapsulates them into one path and chooses among them. A filter VNP's path-set contains the various paths to the destination, and the control procedure chooses among them. The selection criterion can be based on the load on the path, or the source of the traffic. It can also be used to extract out the flows from an aggregation to feed QoS VNPs. A special attribute of filter VNP defines the selection criteria and the corresponding paths. An ISP can setup a filter VNP to choose among multiple paths to the same destination.

7.1.1.2 QoS VNP

VNP control procedures can help in supporting per-flow service guarantees as specified in IETF's Integrated Services mode [9]. A QoS VNP's control procedure implements traffic shaping and traffic policing based on methods like Token Bucket [40][66]. Each path in the VNP's path-set has an individual traffic reservation. A token bucket is maintained for each path in the path set for traffic shaping. For policing, the control procedure checks if the incoming packet is conformant to the traffic specifications of each path. If the packet is conformant it is forwarded as-is otherwise it is forwarded over that path with its class lowered. A VNP whose control procedure implements Weighted Fair Queues (WFQ) will replace those physical ports of a node, which do not support traffic classes.

Figure 7-1 shows the vision for VNP framework. VNPs implement the *packet classifier* and *packet scheduler* of IETF's Integrated Services model. A QoS flow will contain a combination of filter, QoS, and regular VNPs along the same VNP conduit. The resources along the path will be setup by a protocol like RSVP [10][64]. It will provide the parameters for the QoS VNPs along the conduit. A traffic-shaping QoS VNP will be used at the source host or a customer router to shape the flow according to the reservation. Traffic-policing QoS VNP will be used to monitor the flow at each ISP's ingress router. The rest of the VNPs along the flow will be unicast or multicast VNPs. It is possible to use multiple traffic shaping and policing VNPs along the conduit. A filter VNP can be used near the source to classify the packets for the flow.

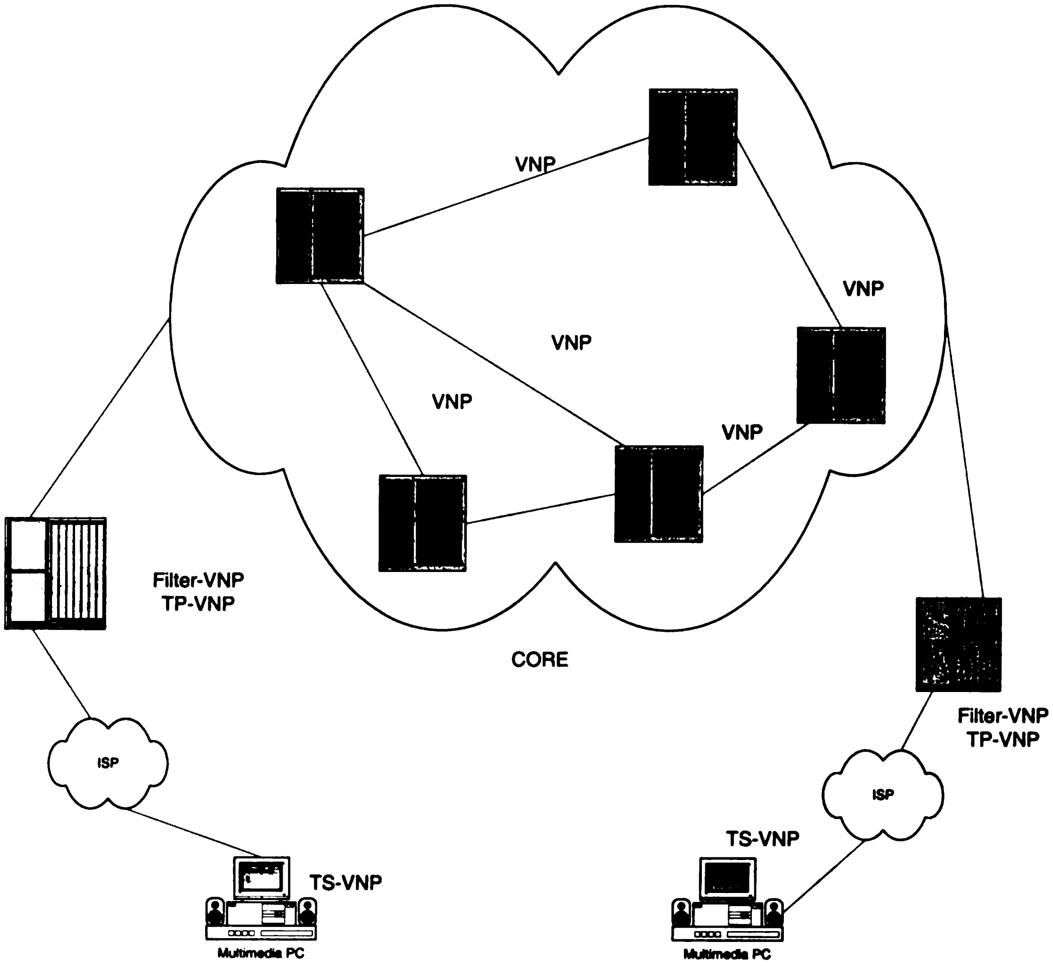


Figure 7-1 VNP Framework Vision

7.1.1.3 Aggregation

An aggregate VNP represents an IP aggregate route [22]. It allows destination-based flows to continue across an IP route aggregation. Its path-set corresponds to the sub-prefixes of the aggregate route. Its forwarding function de-aggregates the incoming packet over the path-set. It looks at the destination address and sub-prefixes and chooses the right path. This is similar to routing table lookup at Layer-3 but can be optimized because it is on a considerable shorter and known range.

If none of the sub-prefixes of the aggregate route is a less-specific prefix of any other, the longest prefix match becomes equivalent to exact match. In such case the selection can be done by hashing, using only a portion of the destination address. Even if the longest prefix match is needed, it will be done on a very small search space. If a VNP represents the aggregate W.X.Y.Z/N, the forwarding function need not look at the first N bits for choosing the right path. The VNP can use the relevant portion of the forwarding table data structures for this purpose instead of having its own. Generally the forwarding base is represented as a trie, and the VNP can have a pointer to the sub-trie corresponding to the aggregate W.X.Y.Z/N at depth N.

7.1.2 Hardware Switches

Hardware switches like ATM are being used on the ISP and corporate backbones. Architecturally, they are very different from conventional routers. Like router, a switch has a number of ports. The forwarding table on a switch contains input port, input channel, output-port and output channel. Incoming frames carry a channel identifier that is used to index into the forwarding table. The input channel identifier is replaced with the corresponding output channel identifier and the frame is sent through the output port. A set of channels that is switched together forms a virtual path. The switch may or may not allow virtual channels to merge. Implementation of VNPs on such switches needs to be explored.

7.1.3 Reduction

The virtual port structures and their processing can be optimized. For example if R_p in the path for a packet is '0' (loop-back) and the L_P is a physical port then the packet need not be switched, it can be directly delivered to the target. The VNP frame can be removed, and the packet delivered through L_P because the packet is anyway destined for that node

7.1.3.1 Multicast

A multicast VNP with the path-set $\{L_1 \rightarrow A_1 \rightarrow R_1, L_1 \rightarrow A_2 \rightarrow R_2, \ldots, L_1 \rightarrow A_n \rightarrow R_n\}$ should be possibly reduced to $\{L_1 \rightarrow M \rightarrow R\}$ where M is a multicast address which groups A_1, A_2, ..., A_n. This involves negotiation among all the concerned nodes to agree upon the same value for the remote port *R*. One way to facilitate this is to differentiate unicast and multicast port-identifiers, similar to [35]. Another way is to allocate unicast port-identifiers from the beginning of the identifier space in increasing fashion, and multicast port-identifiers from the end in decreasing order. In some cases it may not be possible to reduce it to a single path and it may be reducible to $\{L_1 \rightarrow M_1 \rightarrow G_1, L_1 \rightarrow M_2 \rightarrow G_2, \ldots, L_1 \rightarrow M_k \rightarrow G_k\}$ where $k < n$. The attempts for reduction may be driven by time, traffic, or demand.

7.1.3.2 Aggregation

If a node N has a number of virtual ports, V_i, with corresponding path-sets as $\{L \rightarrow A \rightarrow R_i\}$, it may aggregate them through cascading. The decision depends, among others, on whether it originates traffic for any V_i or not. In case of aggregation it does the following

- It creates a virtual port V with the path-set $\{L \rightarrow A \rightarrow 0\}$
- It informs its neighbors to replace any port having path-set $\{P \rightarrow N \rightarrow V_i\}$ with $\{P_n \rightarrow null \rightarrow R_i\}$ where P_n is a new virtual port with the path-set $\{P \rightarrow N \rightarrow V\}$
- It then removes all V_i, $1 \le i \le k$.

The neighbours still have the same granularity of flows but N has reduced the number of ports by cascading the incoming flows.

Similarly, multi-access network ports can be used to aggregate flows. Consider a node N with virtual ports, V_i, $1 \le i \le k$, and corresponding path sets as {L→A_i→R_i}, where L is a multi-access network port. The node N may choose to aggregation in the following way

- It informs its neighbors to replace any port having path-set {P→N→V_i} with {P_n→A_i→R_i} where P_n is a new virtual port with the path-set {P→N→L}
- It then removes all V_i.

7.1.4 Virtual Private Networks

Connecting geographically disparate corporate private networks through WAN links can get very expensive. Such private networks can be connected over cheaper public network links using VNPs, thus implementing *Virtual Private Networks* [23][36][48]. A VNP is setup between the gateways to the public network. Further VNPs are setup between the private networks; these VNPs are cascaded over the VNP on the public network. A control procedure that encrypts its payload is used for the VNP over the public network.

7.1.5 IP as Transport

A VNP generally uses the data link layer to transport packets. For cascading it uses the VNP layer as the transport. It is possible to define a VNP with the complete Layer-3 as its control procedure. This allows for using Layer-3 as the transport, and providing Layer-3 tunnels. This helps with connecting VNP domains that are separated with networks that do not support VNP protocols. When used on the edge nodes of intranets connected by a shared internetwork, it helps in setting up virtual leased lines

7.1.6 Bi-directional multi-access VNP

A physical network port is bi-directional in nature. It was simpler to think of multi-access VNPs as unidirectional; they were easier to implement too. Multi-access VNPs should emulate their physical counterparts in this regard. To accomplish this, a point-to-point VNP has to be setup from egress node to the ingress node; the remote port should also accept packets for the ingress node, and respond to for ARP queries for the ingress node.

7.2 Concluding Remarks

The sudden growth of Internet has exposed some of the problems with its architecture: forwarding performance, traffic engineering, and quality of service. MPLS provides a solution by bypassing the network layer, identifying traffic flows, and providing explicit routes. But it has some drawbacks. The label stacking operations need to be specified explicitly. Switching over interfaces with different MTUs may require fragmentation. The labeled flows are passive and require external processing for traffic manipulation. The proposed method provides a better switching framework through the simple concept of virtual network port. A VNP provides a conduit to a set of remote network ports below Layer-3. VNPs can be combined in different ways for hierarchical routing and label reduction. A VNP's control procedure provides for generic packet processing. Virtual network port framework is conceptually simpler, has better semantics, and is more efficient than labels. It has the following advantages

- Automatic and efficient label stacking
- No fragmentation, since a VNP has the minimum MTU over its path
- Multi-access VNPs help with reduction in memory and other resources
- Payloads from different protocols can use the same VNP
- Active control procedures
- Almost no changes required at IP layer for destination based VNPs.
- Captures more attributes of flow paths, like cost and status
- One less hop, the packet does not go through Layer-3 even at the egress edge router
- Better conceptualization of flows than labels
- No significant processing overhead (due to its complex headers)

BIBLIOGRAPHY

[1] ACHARYA Arup, Rajiv Dighe and Furquan Ansari, "IPSOFACTO: IP Switching Over Fast ATM Cell Transport", IETF Internet Draft, July 1997.

[2] AHMED T., Zitterbart Martina, "Multiprocessing in High Performance IP routers", Protocols for High-speed Networks III (C-9), Elsevier Science Publishers B. V. (North Holland), 1993.

[3] AHMED T., Koufopavlou O, Zitterbart M, Abler J, "On the Design of a Multigigabit IP Router", Journal of High Speed Networks, Vol. 3, No. 3, 1994.

[4] ARNAIZ Alejandro, Doug Sherman and Bob Gohn, "Fast IP: Enhancing Performance and Control in Switched Networks", White Paper, http://www.3com.com, October 1997.

[5] Ascend Inc., "GRF IP Switch", http://www.ascend.com

[6] ASTHANA A., C. Delph, H.V. Jagadish, and P. Kryzanowski, "Towards a Gigabit IP Router", Journal of High Speed Networks, Vol. 1, No. 4, pp. 281-288.

[7] AWDUCHE Daniel O., "MPLS and Traffic Engineering in IP Networks", IEEE Communications Magazine, Dec 1999, pp. 42-47.

[8] BODEN N. J., D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic and W. Su, "Myrinet - A Gigabit-per-second Local-area Network", IEEE Micro, Feb 1995.

[9] BRADEN R., D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture: an Overview", RFC 1633, June 1994.

[10] BRADEN R., L. Zhang, S. Berson, S. Herzog and S. Jamin, "Resource Reservation Protocol", RFC 2205, September 1997.

[11] BRADNER S., "Benchmarking Methodology", RFC 1944.

[12] BRADNER S., "Benchmarking Terminology", RFC 1242.

[13] Cascade Communications Corp, "IP Navigator", White Paper, http://www.casc.com/, December 1996

[14] CIDON I., I. Gopal, P. M. Gopal, R. Guerin, J. Janniello, and M. Kaplan, "The plaNET/ORBIT high speed network", Journal of high speed networks, vol. 2, no. 3, pp. 1-38, Sept. 1993.

[15] Cisco's Gigabit Switch Router, http://www.cisco.com/gsr

[16] Cisco Systems, "Scaling the Internet with Tag Switching", White Paper, http://www.cisco.com

[17] COMER Douglas E., "Internetworking with TCP/IP Volume I: Principles, Protocols, and Architecture", Prentice Hall, 1995.

[18] COMER Douglas E., David L Stevens, "Internetworking with TCP/IP Volume II: Design, Implementation, and Internals", Prentice Hall, 1991.

[19] DEGERMARK Mikael, Andrej Brodnik, Svante Carlsson and Stephen Pink, "Small Forwarding Tables for Fast Routing Lookups", ACM SIGCOMM 97, September 1997.

[20] DAVIE B, P. Doolan, and Y. Rekhter, "Switching in IP Networks". San Mateo, CA: Morgan Kaufmann, 1998.

[21] DEMIZU Noritoshi, "Multi Layer Routing", http://infonet.aist-nara.ac.jp/member/nori-d/mlr

[22] FULLER V., T. Li, J. Yu and K. Varadhan, "Classless Inter-Domain Routing: an Address Assignment and Aggregation Strategy", RFC 1519, September 1993.

[23] GHANWANI Anoop, Bilel Jamoussi, Dan Fedyk, Peter Ashwood-Smith, Li Li, and Nancy Feldman, "Traffic Engineering Standards in IP Networks Using MPLS", IEEE Communications Magazine, Dec 1999, pp. 49=53.

[24] GLEESON Bryan, Arthur Lin, Juha Heinanen and Grenville Armitage, "A Framework for IP Based Virtual Private Networks", IETF Internet Draft, September 1998.

[25] GUPTA Pankaj, S. Lin, and N. McKeown, "Algorithms for Packet Classification", IEEE Network, Vol. 15, No. 2, March 2001.

[26] GUPTA Pankaj, S. Lin, and N. McKeown, "Routing lookups in hardware at memory access speeds," Proc. IEEE INFOCOM, Apr. 1998, pp. 1240-1247.

[27] HALABI B., "Internet Routing Architecture," Cisco Press, New Riders Publishing, 1997.

[28] HORST Robert W. and Dave Garcia, "Servernet SAN I/O Architecture", Hot-interconnects V, Stanford, 1997.

[29] KATSUBE Y. et al, "Toshiba's Router Architecture Extension for ATM : Overview", RFC 2098.

[30] LAMPSON Butler, Venkatachary Srinivasan, George Verghese, "IP Lookups using multiway and multicolumn search", IEEE/ACM Transactions on Networking, June 1999, Vol. 7, No. 3, pp. 324-334

[31] MARTIN James, Kathleen Kavanagh Chapman, Joe Leben, "Enterprise Networking: Data Link Subnetworks", Prentice Hall, 1996.

[32] MCAULEY Anthony J. and P. Francis, "Fast routing table lookup using CAMs", Proceedings of INFOCOM, pp 1382-1391, March-April 1993.

[33] MCKEOWN N., M. Izzard, A. Mekkittikul, B. Ellersick and M. Horowitz, "The Tint Tera: A packet switch core", IEEE Micro, January 1997, vol. 17, pp. 26-33.

[34] MOY J., "OSPF Version 2", RFC 1853, March 1994.

[35] MPLS Working Group, "MPLS Mailing List Archive", IETF MPLSWG, ftp://ftpeng.cisco.com/mpls/mpls

[36] MUTHUKRISHNAN Karthik and Andrew Malis, "Core IP VPN Architecture", IETF Internet Draft, Oct 1998.

[37] NEWMAN Peter, T. Lyon and G. Minshall, "Flow Labeled IP: A connectionless approach to ATM", Proceedings of IEEE Infocom, March 1996.

[38] NEWMAN Peter, G. Minshall, T. Lyon and L. Huston, "IP Switching and Gigabit Routers", IEEE Communications Magazine, Jan 1997.

[39] NEWMAN Peter, Greg Minshall and Thomas L Lyon, "IP Switching – ATM Under IP", IEEE/ACM Transactions on Networking, vol 6, no 2, April 1998, pp 117-129.

[40] PARTRIDGE Craig, "Gigabit Networking", Addison-Wesley, April 1995

[41] PARTRIDGE Craig, Philip P. Carvey et al, "A 50-Gb/s IP Router", IEEE/ACM Transactions on Networking, vol 6, no 3, June 1998, pp 237-248.

[42] PARTRIDGE Craig, W. Milken, J. Rokoaz and S. Storch, "BBN's Multi Gigabit Router", http://www.net-tech.bbn.com

[43] PARTRIDGE Craig, "Locality of route caches", NSF Workshop On Internet Statistics Measurements and Analysis, San Deigo, CA, USA, February 1996.

[44] PERLEMAN Radia, "Interconnections: Bridges and Routers", Addison-Wesley, 1992.

[45] Pluris Inc, "Pluris Massively Parallel Routing", White Paper, http://www.pluris.com

[46] REKHTER Y., B. Davie, D. Katz, E. Rosen, G. Swallow, "Cisco Systems' Tag Switching Architecture Overview", RFC 2105, February 1997.

[47] ROSEN Eric C., Arun Viswanathan, Ross Callon, "Multiprotocol Label Switching Architecture", IETF Internet Draft, March 1998.

[48] ROSEN Eric C. and Yakov Rekhter, "BGP/MPLS VPNs", IETF Internet Draft, November 1998.

[49] ROSEN Eric C., Yakov Rekhter, Daniel Tappen, Dino Farinacci, Guy Fedorkow, Tony Li and Alex Conta, "MPLS Label Stack Encoding", IETF Internet Draft, September 1998.

[50] RUIZ-SANCHEZ Minguel A., Ernst W. Biersack, and Walid Dabbous, "Survey and Taxonomy of IP Address Lookup Algorithms", IEEE Network, Vol. 15, No. 2, March 2001.

[51] SHENKER S. and C. Partridge, "Specification of Guaranteed Quality of Service", RFC 2212, September 1997.

[52] PARULKAR Guru, D.C. Schmidt and J.S. Turner, "IP/ATM: A Strategy for integrating IP with ATM", Proceedings of ACM SIGCOMM, Cambridge MA, September 1995, pp 49.

[53] SKOWLER Keith, “A tree-based routing table for Berkeley Unix”, Technical Report, University of California, Berkeley, 1993.

[54] STALLINGS William, “Data and Computer Communications”, Prentice Hall, 1997.

[55] SWALLOW George, “MPLS Advantages for Traffic Engineering”, IEEE Communications Magazine, December 1999, pp 54-57.

[56] Torrent Networking Technologies Corp, “The IP9000 Gigabit Router Architecture”, Technical Paper, http://www.torrentnet.com.

[57] Torrent Networking Technologies Corp, “High Speed Routing Table Search Algorithms”, Technical Paper, http://www.torrentnet.com.

[58] VISWANATHAN A., N. Feldman, R. Boivie and R Woundy, “ARIS : Aggregate Route-Based IP Switching”, IETF Internet Draft, March 1997.

[59] VUPPALA Vibhavasu and L. M. Ni, “Design of a Scalable IP Router”, Hot Interconnects V, Aug 1997.

[60] VUPPALA Vibhavasu and L. M. Ni, “Virtual Network Ports: A New Switching Framework”, International Conference on Computer Communications (ICCC’99), Tokyo, September 1999.

[61] VUPPALA Vibhavasu and L. M. Ni, “Implementing Layer-3 Switching Using Virtual Network Ports”, International Conference on Computer Communications and Networking (ICCCN), Boston, October 1999.

[62] VUPPALA Vibhavasu and L. M. Ni, “Virtual Network Ports: Design, Implementation, and Applications”, The fifth IEEE Symposium on Computer Communications (ISCC), Antibes-Juan le Pins, France, July 2000.

[63] WALDVOGEL Marcel, George Verghese, Jon Turner, and Bernhard Plattner, "Scalable High Speed IP Routing Lookups", ACM SIGCOMM 97, September 1997.

[64] WRIGHT Gary R., W. Richard Stevens, "TCP/IP Illustrated Volume 2: The Implementation", Addison Wesley, 1995.

[65] WROCLAWSKI J., "The use of RSVP with IETF Integrated Services", RFC 2210, September 1997.

[66] WROCLAWSKI J., "Specification of the Controlled-Load Network Element Service", RFC 2211, September 1997.

[67] XIAO Xipeng, Alan Hannan, Brook Bailey, and Lionel M. Ni, "Traffic engineering with MPLS in the Internet", IEEE Network, Vol. 14, No. 2, March 2000, pp 28-33.

[68] YEHUDA Afek, Anat, Bremler-Barr, Sariel Har-Peled, "Routing with a clue", ACM Transactions on Networking, Vol. 9, No. 6, December 2001.

[69] ZITTERBART M., "A Multiprocessor Architecture for High Speed Network Interconnections", IEEE INFOCOMM, 1989.