

Strategies and Algorithms for Virtual Private Networks with Quality of Service Guarantees

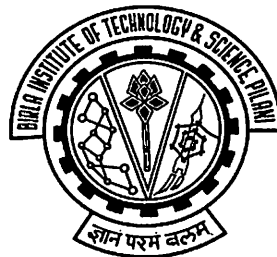
THESIS

Submitted in partial fulfilment
of the requirements for the degree of
DOCTOR OF PHILOSOPHY

by

CHITTARANJAN HOTA

Under the Supervision of
Prof. G. Raghurama




**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
PILANI (RAJASTHAN) INDIA
2005**

**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
PILANI (RAJASTHAN) INDIA**

C E R T I F I C A T E

This is to certify that the thesis entitled “**Strategies and Algorithms for Virtual Private Networks with Quality of Service Guarantees**” and submitted by Chittaranjan Hota ID. No. 2001PHXF014 for award of Ph.D. Degree of the Institute, embodies original work done by him under my supervision.

Signature in full of the Supervisor: 

Name in capital block letters: PROF. G. RAGHURAMA

Designation:

Dean, Faculty Division – II &
Chief, Admissions & Placement
Birla Institute of Technology
and Science
Pilani – 333 031 (Rajasthan)
INDIA

Date: 15 Oct 2005

Acknowledgements

I am grateful to my supervisor Prof. (Dr.) G. Raghurama, Dean, Faculty Division-II, and Chief, Admissions & Placement Unit, Birla Institute of Technology and Science, Pilani, Rajasthan for his insightful suggestions in both the content and presentation of this thesis. It was his encouragement, guidance, support and patience that saw me through and I am ever grateful to him.

Special thanks are due to Prof. S. Venkateswaran, Vice Chancellor; Prof. L. K. Maheswari, Pro-Vice Chancellor and Director BITS, Pilani; Prof. K. E. Raman, Deputy Director (Administration); Prof. V. S. Rao, Deputy Director (Off-Campus); Prof. Ravi Prakash, Dean, Research and Consultancy Division, Prof. A. K. Sarkar, Dean, Instruction Division, Prof. R. N. Saha, Dean, Educational Development Division, Prof. B. R. Natrajan, Dean, DLPD, Birla Institute of Technology & Science, Pilani, for giving me an opportunity and encouragement for PhD program. Further, I sincerely acknowledge the resource support, encouragement and help received from Prof. J. P. Mishra, Unit Chief, IPC Unit, BITS, Pilani, at various stages of the work. I sincerely thank members of the doctoral advisory committee, Prof. Rahul Banerjee, Prof. Sundar Balasubramaniam, and Prof. Sanjay Srivastava, for their critical remarks and help in improving the work. I would also like to express my gratitude to all the support staff at Cisco Networking Research Laboratory, BITS, Pilani for their help and support during the testbed implementation phase of the research work.

Part of this research has been carried out at University of New South Wales, Sydney, Australia. I am grateful to UNSW for providing me with Australian Vice Chancellors Committee scholarship for the period of my stay at Sydney, and also I would like to thank Prof. Paul Compton, Head, School of Computer Science & Engineering; Prof. Sanjay Kumar Jha, Head, Network Research Group; Dr. William Lau, Associate Lecturer, UNSW, and other members of Network Research Laboratory for their help and support during my stay over there. I appreciate the help rendered by Mr. Philip Rosenbaum, research scholar in NRL, UNSW in installing and working with the CPLEX optimization tool.

My special gratitude goes to Prof. Aditya P. Mathur, Department of Computer Sciences, Purdue University, USA for his valuable remarks and recommendations in improving the work during the regular PhD seminars at BITS, Pilani. I would also take this opportunity to thank Prof. Tham Chen Khong, Associate Professor, National University of Singapore, Singapore for his valuable suggestions.

The finalization of this Thesis would not be possible without the love and support from my parents, my wife and my sons. I am grateful to them for the continuous encouragement and support during the years of my research.


(Chittaranjan Hota)

Abstract

In today's world, many e-business initiatives such as workforce optimization, customer care, e-commerce, and corporate communication often depend on unprecedented levels of network access and agility. Pursuing these initiatives needs complex networking, dedicated IT resources for 24-hour monitoring and management, and cost-effective integrated voice and data services. In the face of these demands, many business houses are reassessing their corporate networking infrastructures and seriously considering outsourcing alternatives. Service providers offering managed IPVPN services can help corporate business meet today's challenges. The primary function of an IPVPN is to provide cost-effective, secure connectivity over a shared infrastructure with identical policies and service attributes that the enterprise enjoys within its dedicated private network. Towards this objective, the IPVPN solution must deliver the following essential attributes: high availability, network security, quality of service (QoS), scalability, and ease of management. Traditionally, VPNs have been deployed for reasons of economy of scale, but were mostly configured statically and were not designed to offer any quality of service guarantee.

This thesis describes algorithms for building dynamic and resource-assured managed VPNs that can satisfy users of the VPN as well as increase the revenue generation for service providers. These algorithms span the areas of optimal tunnel path finding, dynamic bandwidth management, unicast and multicast restoration of VPNs. It also proposes an architectural framework for deploying VPNs.

Building up an optimal tunnel path between a single source and multiple destinations in an Intranet/Extranet VPN scenario has been a recent research issue. Without activating any core router, we can find out an optimal path between source and destinations using well-known optimal path finding algorithms. This solution can again be optimized by selectively activating a set of core routers till the funds metric is exhausted. This problem is modeled as a NP-hard problem. This thesis proposes heuristic algorithms to solve this NP-hard problem. It also compares the developed algorithms with the existing well known shortest path heuristic algorithms.

Network QoS pertains to latency, jitter, and packet loss—metrics that determine the quality of the services delivered over the network from end to end. A related network attribute, class of service (CoS) defines the specific level of service required for different traffic types, such as voice, video, or data. Enterprises that deliver mission-critical applications over IPVPN typically look for a service provider that offers multiple CoS. The QoS mechanisms within IPVPN enable multiple CoS by classifying traffic types and assigning priority to those that are mission-critical or delay-sensitive, such as voice and video. QoS mechanisms also enable a VPN to manage congestion across varying bandwidth rates. Enterprise bandwidth and connectivity needs change over time, sometimes suddenly, as the business expands, consolidates,

merges, or begins encouraging telecommuting services. A need for connectivity to extranet partners or customers also boosts bandwidth and connectivity requirements. To remain agile, the enterprise needs to choose a service provider with the ability to scale its VPN to accommodate unplanned growth and changes within its operational periphery. In a VPN, the way network resources, mainly bandwidth will be allocated presents an interesting research issue. The recently proposed models like Hose VPNs are purely based on the idea of optimally assigning bandwidth to the hose pipes for effective utilization of network resources while at the same time maximizing network performance. This thesis proposes dynamic bandwidth allocation and management algorithms for VPNs using an agent based approach. These algorithms partition the link bandwidth among different VPNs and also partition a single VPN bandwidth among several users of the tunnel. This partitioning is dynamic. The bandwidth allocated to VPN users are either incremented or decremented on the basis of the traffic flow within a tunnel and also within inter tunnels. This is done to meet the user satisfaction level and also to increase revenue generation of service providers.

An IPVPN needs to predictably deliver high service availability for enterprise users, partners and customers—a capability that requires high network reliability. Some service providers are able to offer service level agreements (SLAs), usually for an additional fee. Service provider SLAs define the specific terms or metrics regarding availability of networking resources, and offer the VPN subscriber a contractual guarantee for network uptime and performance. An SLA can optionally define multiple levels of service for different types of traffic, such as voice and data, with lower-cost alternatives for less critical traffic. Towards this objective, this thesis proposes new algorithms for restoration strategies that provision bandwidth guaranteed recovery for unicast and multicast connections. The primary focus is on online restoration strategies that sequentially do pre-planning of resources for each request using the current network resource state. These algorithms compute active as well as backup paths for source destination pairs with quality of service guarantees. These algorithms evaluate the performance using several metrics like: total bandwidth requirement, the number of requests accepted in the network, the funds, and the computational time. Each algorithm makes trade-off between computational complexity, bandwidth efficiency, and number of accepted requests. The restoration strategies used for unicast and multicast connections in this thesis are compatible and thus it is possible to integrate the restoration strategies into a single system where they share the same backup resources. This thesis also models the multicast VPN using Integer Linear Programming formulations to get an optimal solution to minimal cost multicast VPN problem.

This thesis also proposes a layered VPN planning framework for business houses. It describes security issues corresponding to several layers of analysis like VPN topology, firewalls, survivability, quality of service, scalability etc. It also describes a case study based on actual implementation experience with a testbed. It also proposes a framework for supporting mobile VPN users in an IPVPN environment.

Table of Contents

Title Page	I
Certificate	II
Acknowledgements	III
Abstract	IV
Table of Contents	VI
List of Tables	XI
List of Figures	XII
List of Mathematical Notations	XV
List of Abbreviations	XVII
1 Introduction	1
1.1 Overview.....	1
1.2 Contribution of the Thesis.....	5
1.3 Structure of the Thesis.....	8
2 Virtual Private Networks – An Overview	10
2.1 What is a Virtual Private Network?.....	10
2.2 A Security Services Taxonomy.....	13
2.3 VPNs Based on IP Tunnels.....	15
2.4 Merits and Demerits of IPVPNs.....	16
2.5 Features of a VPN.....	17
2.6 VPN Protocols.....	20
2.7 Summary.....	23
3 Quality of Service in IP Virtual Private Networks	25
3.1 Internet Architecture.....	25
3.1.1 Introduction.....	25
3.1.2 Internet Protocol – IP.....	25
3.1.3 IP Addresses and Routing.....	28
3.1.4 An Overview of ISP Networks.....	29

3.2 Quality of Service in IP Virtual Private Networks.....	30
3.2.1 QoS from End-user and Provider's Perspective.....	30
3.2.2 Differentiated Services (DiffServ).....	32
3.2.3 Traffic Engineering.....	35
3.2.3.1 IP-ATM Overlay Networks.....	36
3.2.3.2 Multiprotocol Label Switching (MPLS).....	37
3.2.3.3 Constraint-Based Routing (CBR).....	40
3.3 Policy-Based QoS Provisioning in IPVPNs.....	41
3.4 Summary.....	43
4 Optimal Tunnel Path Finding in IP Virtual Private Networks	44
4.1 Requirements of IPVPN Tunnels.....	44
4.2 Approaches for Establishing IPVPN Tunnels.....	46
4.3 Minimal Cost (MC) VPN Problem.....	50
4.4 Literature Survey.....	55
4.4.1 Cohen's Optimal Cost VPN.....	55
4.4.2 Mitra's Joint Resource Allocation & Routing in VPNs.....	55
4.4.3 Anupam's VPN Design for Multicommodity Flow.....	56
4.4.4 Amit's VPN Provisioning in the Hose Model.....	56
4.4.5 Tuna's Stochastic Approach for Tunnel Path Computation.....	57
4.4.6 Erlebach's Multi-path Routing in Hose VPNs.....	57
4.5 Heuristic Algorithms for Minimal Cost Tunnel Path in IP Virtual Private Networks	57
4.5.1 Path Finding for MC VPN	59
4.5.2 Bit wise Positive Consideration Heuristic (BPCH).....	60
4.5.3 Implementation of BPCH	61
4.5.4 Activating Core Routers	65
4.6 Simulation Results and Analysis of Tunnel Path Finding Algorithms	68
4.7 Summary.....	70
5 Dynamic Bandwidth Management in IP Virtual Private Networks	71
5.1 Bandwidth Management and Logical Networks	71
5.2 Bandwidth Allocation Mechanisms	73
5.2.1 Static Link Partitioning Scheme	74
5.2.2 Dynamic Bandwidth Partitioning Scheme	75
5.2.3 Other Partitioning Schemes.....	75

5.3 Utility-based Bandwidth Allocation in IPVPNs	78
5.4 Dynamic Bandwidth Management in IPVPN Tunnels	80
5.5 Literature Survey on Dynamic Bandwidth Management in IPVPNs	86
5.5.1 Duffield's Hose Model VPN	86
5.5.2 Goyal's Distributed Open Signaling Architecture	87
5.5.3 Garg's Stochastic Fair Sharing Approach	87
5.5.4 Other Resource Allocation Approaches	89
5.6 Algorithms for Dynamic Bandwidth Management	89
5.6.1 Distributed Message Communication	93
5.6.2 Proposed Algorithms	94
5.6.3 Implementation	100
5.7 Simulation Results and Analysis of Dynamic Bandwidth Management Algorithms	102
5.8 Summary	104
6 Restoration in IP Virtual Private Networks	106
6.1 Protection at Different Layers of OSI Stack	106
6.2 Link Layer Restoration Strategies	109
6.3 Desirable Characteristics of Joint Dynamic Bandwidth Management and Restoration of IPVPNs.....	114
6.4 Literature Survey on Restoration Strategies in IPVPNs.....	115
6.4.1 Kar's Minimum Interference Routing	115
6.4.2 Kodialam's Minimal Information & Complete Information Scenario.....	116
6.4.3 Li's Full Information Restoration	117
6.4.4 Liu's Successive Survivable Routing	117
6.4.5 Iraschko's Algorithms	117
6.4.6 Fei's Algorithm	118
6.4.7 Singhal's Algorithms	119
6.4.8 Italiano's Approximation Algorithm.....	119
6.5 Algorithms for Survivable IP Virtual Private Networks.....	120
6.5.1 Minimal Cost (MC) Unicast Restorable IPVPN Problem.....	122
6.5.2 Proposed APH and BPH Algorithms for Unicast Restoration.....	124
6.5.3 Multicast IPVPN Restoration.....	129
6.5.4 Simulation Results of Unicast Restoration.....	143
6.5.5 Simulation Results of Multicast Restoration.....	147
6.6 Summary.....	149

7 Layered Framework for IPVPN Deployment	151
7.1 VPN Deployment Challenges.....	151
7.2 Challenges with Current Legacy Technologies.....	153
7.3 VPN Planning Issues.....	154
7.4 Comparison of VPN Solutions with Traditional Solutions.....	156
7.5 IPVPNs Supporting Mobile Users.....	159
7.6 Place Oriented VPNs (PO-VPNs).....	159
7.7 Active Networks and Software Agents.....	161
7.8 Design Challenges: A Case Study.....	162
7.8.1 VPN Gateways.....	164
7.8.1.1 VPN Gateway Functions.....	164
7.8.2 Interaction with Firewalls.....	165
7.8.2.1 Gateway and Firewall in Parallel.....	166
7.8.2.2 Gateway and Firewall in Series.....	166
7.8.3 VPN Topology.....	167
7.8.4 Addressing & Routing.....	168
7.8.5 Quality of Service.....	169
7.8.6 Scalability.....	170
7.8.7 Resiliency.....	171
7.8.8 VPN Solution Scenario.....	172
7.9 Mobile VPN User Support using Active Networks and Intelligent Agents.....	178
7.9.1 Mobile VPN User Support.....	180
7.9.1.1 Initial VPN Provisioning.....	181
7.9.1.2 Dynamic VPN Adoption.....	182
7.9.2 Active Quality of Service Routing.....	184
7.9.2.1 Active Packet Format.....	187
7.9.2.2 Server & Client Implementation.....	188
7.9.2.3 Route Optimization & Routing Agent.....	189
7.9.3 Analysis of Proposed Layered Framework.....	190
7.9.4 Analysis of Proposed Framework for Mobile VPN User Support.....	190
7.10 Summary.....	191
8 Conclusions	193
8.1 Optimal Tunnel Path Finding.....	194
8.2 Dynamic Bandwidth Management.....	195
8.3 Unicast IPVPN Restoration.....	196

8.4 Multicast IPVPN Restoration.....	196
8.5 Layered Framework for IPVPN Deployment.....	197
8.6 Mobile User Support in IPVPNs.....	197
8.7 Future Work.....	198
References	200
Appendix A IPsec VPN Implementation	212
Appendix B Traffic Engineering in MPLS VPN	238
Appendix C Simulation of IPsec VPN in Tunnel and Transport Modes	253
List of Publications	269
Biography	271

List of Tables

Table	Caption	Section/ Page No.
2.1	Different Protocols with VPN types	2.6/22
3.1	Thresholds for acceptable QoS [Bouch00]	3.2/30
3.2	Bandwidth and Delay requirements of typical Internet applications	3.2/32
3.3	Impact of Design Issues on the QoS parameter	3.2/33
3.4	LSP Attributes	3.2/39
4.1	Parameters defining Cost of a Link	4.3/51
4.2	QoS features with binary weight	4.5/61
4.3	Priorities for Different QoS parameters	4.5/62
7.1	Advantages and disadvantages of current legacy solutions	7.2/153
7.2	Traditional Solutions versus IPVPN solutions	7.4/156
7.3	Search Matrix	7.9/186
7.4	Different Kinds of Cost for Different Traffic	7.9/186

List of Figures

Figure	Caption	Section/ Page No.
2.1	A Virtual Private Network	2.1/10
2.2	An Intranet VPN	2.1/11
2.3	A Remote Access VPN	2.1/12
2.4	Six Tunneling models for VPNs [Yuan01]	2.2/14
2.5	Tunneling	2.3/16
2.6	Different VPN Types with Protocols	2.6/23
3.1	IPv4 Header	3.1/26
3.2	A sample part of an ISP network	3.1/29
3.3	A Comparison of Application Delay Sensitivity and Criticality in an Enterprise	3.2/31
3.4	Differentiated Services Architecture	3.2/33
3.5	IP-ATM Core Physical Topology	3.2/37
3.6	MPLS Traffic Engineering	3.2/38
3.7	Constraint-Based Routing	3.2/40
3.8	IETF Policy-Based Management Architecture	3.3/42
4.1	Network Example	4.2/47
4.2	CPE based approach	4.2/48
4.3	Network based Approach	4.2/48
4.4	Another Network based approach	4.2/49
4.5	An Intranet/Extranet based VPN Scenario	4.5/58
4.6	Example Network Graph (a) Step 1: Only one neighbor (b) Step 2: P4 is selected (c) Step 3: PE2 is selected (d) Step 4: Path from CE1 to CE2	4.5/63
4.7	Joining of both the VPN trees	4.5/65
4.8	The core router PE1 is activated giving rise to three tunnels	4.5/67
4.9	Increasing Funds also increases the Heuristic function estimation of total path	4.6/69
4.10	Increasing Border routers also incurs overhead on VPN cost for a fixed fund	4.6/69
5.1	VPN Bandwidth Management with other Network Management Functions	5.1/72
5.2	Types of IPVPN traffic and their utility functions	5.3/79
5.3	Initial Tunnel Setups and Link Capacities	5.4/85
5.4	Bandwidth Reallocation when T1 needs 2 mbps additional capacity	5.4/85

Figure	Caption	Section/ Page No.
5.5	Bandwidth Reallocation when T1 needs 4mbps additional capacity	5.4/86
5.6	Bandwidth Path Rerouting when T1 needs 8mbps more	5.4/86
5.7	Relationship between Agents and Bandwidth Managers	5.6/91
5.8	Distributed Message Communication	5.6/94
5.9	Intranet VPN Scenario with Bandwidth Manager	5.6/101
5.10	Utilization of a link with number of users	5.7/103
5.11	Utilization of a link with Average Traffic Rate	5.7/103
5.12	Request Blocking Rate with Average Traffic Rate	5.7/104
6.1	Active and Backup VPN Paths	6.1/109
6.2	Link/Path Restoration Strategies (a) Active Path 1-2-3-4 (b) Backup Path 1-2-6-5-3-4 (c) Backup Path 1-6-5-4	6.2/112
6.3	Intranet/Extranet VPN scenario	6.5/121
6.4	VPN Tree with source and destinations	6.5/127
6.5	Optimal VPN Tree with Funds = 1	6.5/127
6.6	Optimal VPN Tree with Funds = 2	6.5/127
6.7	Backup Tree for Failure Scenario f(S-C1, C3-C5)	6.5/128
6.8	Optimal Backup Tree for Failure Scenario f with Funds =1	6.5/128
6.9	An IPVPN based on the customer-pipe model	6.5/130
6.10	A VPN based on the hose model	6.5/131
6.11	Active and Backup Multicast VPN tree (Dotted Lines show edges of Active and dashed lines show edges of backup VPN)	6.5/134
6.12	VPN Multicast tree after step 2	6.5/139
6.13	Multicast active tree with CPS and PS	6.5/139
6.14	Backups for FS1 of CPS	6.5/140
6.15	Backups for FS2 of CPS	6.5/140
6.16	Backups for FS of PS	6.5/140
6.17	A Metropolitan Network	6.5/144
6.18	Active and Backup Path Cost with Funds = 2	6.5/145
6.19	Active and Backup Path cost with Funds =13	6.5/145
6.20	Active and Backup Path Cost with varied funds for group size = 15	6.5/146
6.21	Average number of Participating Core Routers with Group Size	6.5/146
6.22	A Random Sparse Network	6.5/148
6.23	Average Cost with Single Link Failure	6.5/148

Figure	Caption	Section/ Page No.
6.24	Average Cost with Multiple Link Failures	6.5/148
6.25	Funds Vs. Active Path and Backup Path Cost	6.5/149
7.1	Data Traffic Mix for North American Multinational Corporations [Source: the Yankee Group, 2002]	7.1/152
7.2	IDC WAN Manager Survey	7.1/152
7.3	Place Oriented Virtual Private Network	7.6/160
7.4	Inbound and Outbound Processing in a VPN Gateway	7.8/164
7.5	VPN Gateway and Firewall in Parallel	7.8/166
7.6	Gateway first, Firewall next	7.8/167
7.7	Firewall first, Gateway next	7.8/167
7.8	VPN Topologies	7.8/168
7.9	IPVPN Resiliency	7.8/172
7.10	Proposed IPVPN Design	7.8/177
7.11	Relationship Between Different Processes	7.9/183
7.12	Negotiation and Utilization Messages	7.9/184
7.13	Active Packet Format	7.9/187
7.14	Structure of Execution Environment	7.9/189
A.1	Generic Illustration of the Configured IPSec Tunnel	Appendix A/221
B.1	VPN Connectivity Using LSP Tunnels	Appendix B/239
B.2	Nested LSPs Providing VPN Connectivity	Appendix B/241
C.1	Testbed for IPSec VPN in Tunnel and Transport Mode	Appendix C/254

List of Mathematical Notations

Notation	Details
$G (V, E)$	Directed Graph G with vertices V and edges E
$c(e)$	Edge weight function
$w(v)$	Vertex weight function
F	A bound on available funds
M	Set of border nodes
S	Headquarter node
C_i	Capacity of i^{th} link
α_i	The weight associated with the i^{th} link L_i
L	Link
ϕ_i	The traffic intensity of i^{th} link (L_i)
C_k	Bandwidth of K^{th} traffic class
n_i	Number of users in i^{th} tunnel.
r_k	Trunk reservation for class k
$U_i()$	Utility function for traffic class i
R	Traffic Rate
δ	Additional bandwidth required
b_i	Bandwidth allocated to i^{th} user in a tunnel
SC	Spare Capacity of a link
$\text{Max}_U_i(b)$	Maximum bandwidth utilization of a tunnel
δ_{victim}	δ amount of bandwidth released from victim process
TS_{ij}	The set of tunnels over a link
T_i	Shortest tunnel path between source and destination
U_u	User Utility
BM	Bandwidth Manager
η	A small increment in the bandwidth
η_p	Slope of incoming traffic in previous interval
η_c	Slope of incoming traffic in current interval
$\text{Cost}_{i,j}$	Cost of links from i to j
No_Succ	Number of successors of a node
Profit	Cost savings

Notation	Details
ActiveList	A list structure to hold the active core nodes
A	A set of active nodes
T	Active VPN tree
BT	Backup VPN tree
FS	Failure scenario set
f	A failure scenario
X_{in}	Hose ingress bandwidth
X_{out}	Hose egress bandwidth
X	A vector representing flow on the active path
Y	A vector representing flow on backup path
$D_p(i)$	The physical degree of node i
CPS	Critical Primary Segment
PS	Primary Segment
AP	Active Path
BP	Backup Path

List of Abbreviations

Abbreviation	Details or Expanded Form
AA	Active Application
ADTH	Active Double Tree Heuristic
AF	Assured Forwarding
AP	Agent Process
APH	Active Path Heuristic
API	Application Programming Interface
AQR	Active Quality of Service Routing
ARP	Address Resolution Protocol
ASPH	Active Shortest Path First
ATM	Asynchronous Transfer Mode
B2B	Business to Business
BB	Bandwidth Broker
BGP	Border Gateway Protocol
BPCH	Bit-wise Positive Consideration Heuristic
BPH	Backup Path Heuristic
BR	Border Router
CA	Certificate Authority
CBP	Connection Blocking Probability
CBQ	Class Based Queuing
CBR	Constraint Based Routing
CE	Customer Edge
CED	Customer Edge Device
CoS	Class of Service
CPE	Customer Premises Equipment
CPS	Critical Primary Segment
DARPA	Defense Advanced Research Projects Agency
DES	Data Encryption Standard
DHCP	Dynamic Host Configuration Protocol
DiffServ	Differentiated Services
DNS	Domain Name System
DS	Differentiated Services

Abbreviation	Details or Expanded Form
DSCP	Differentiated Services Code Point
ECN	Explicit Congestion Notification
EE	Execution Environment
EF	Expedited Forwarding
EGP	Exterior Gateway Protocol
FEC	Forwarding Equivalence Class
GA	Group Agent
GRE	Generic Routing Encapsulation
HR	Hosting Router
HVP	Hierarchical Virtual Partitioning
ICMP	Internet Control Message Protocol
ICSA	International Computer Society Association
IETF	Internet Engineering Task Force
IGMP	Internet Group Management Protocol
IGP	Internet Gateway Protocol
IHL	Internet Header Length
IKE	Internet Key Exchange
ILP	Integer Linear Program
IntServ	Integrated Services
IP	Internet Protocol
IPSec	Internet Protocol Security Standard
IPVPN	Internet Protocol Virtual Private Network
IPX	Internet Packet Exchange
ISO	International Standard Organization
ISP	Internet Service Provider
ITU	International Telecommunications Union
L2TP	Layer 2 Tunneling Protocol
LAC	L2TP Access Concentrator
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
LDP	Label Distribution Protocol
LER	Label Edge Router
LNS	L2TP Network Server
LSP	Label Switched Path

Abbreviation	Details or Expanded Form
LSR	Label Switching Router
MC-VPN	Minimal Cost – Virtual Private Network
MPLS	Multiprotocol label switching
NAT	Network Address Translation
NPA	Network Provider Agent
OSI	Open System Interconnect
OSPF	Open Shortest Path First
PDP	Policy Decision Point
PE	Provider Edge
PEP	Policy Enforcement Point
PHB	Per-hop Forwarding Behavior
POP	Point of Presence
PO-VPN	Place Oriented Virtual Private Network
PPTP	Point-to-Point Tunneling Protocol
PS	Primary Segment
PVC	Permanent Virtual Circuits
QoS	Quality of Service
RADIUS	Remote Authentication Dial-In User Service
RARP	Reverse Address Resolution Protocol
RAS	Remote Access Services
RIP	Routing Information Protocol
RSVP	Resource Reservation Protocol
SLA	Service Level Agreement
SLS	Service Level Specification
SNA	Systems Network Architecture
SP	Service Provider
SPA	Service Provider Agent
SPH	Shortest Path Heuristic
SPI	Security Parameter Index
SSL	Secure Socket Layer
STP	Steiner Tree Problem
TCP	Transmission Control Protocol
TOS	Type of Service
TTL	Time to Live

Abbreviation	Details or Expanded Form
UA	User Agent
UDP	User Datagram Protocol
VBR	Variable Bit Rate
VC	Virtual Circuit
VCI	Virtual Channel Identifier
VLL	Virtual Leased Line
VOIP	Voice Over Internet Protocol
VPDN	Virtual Private Dial-up Network
VPI	Virtual Path Identifier
VPLS	Virtual Private LAN Services
VPN	Virtual Private Network
VPRN	Virtual Private Routed Network
WAN	Wide Area Network
WINS	Windows Internet Naming System

Chapter 1

Introduction

1.1 Overview

The Internet has a very interesting and chequered past. Born out of the cold war, it assumed an extraordinary prominence in the early 1990s, and became a buzzword in the late 1990s, as the century drew to a close. It has had such a profound impact on the way people communicate that it ranks among the greatest hallmarks in the evolution of communication. The Internet has become a popular low-cost backbone infrastructure. Its universal reach has led many companies to consider constructing a secure virtual private network (VPN) over the public Internet. Corporations with branch offices and facilities across the globe wish to connect their subnetworks hundreds or even thousands of mile away. Traditional solutions based on dedicated leased lines are gradually being replaced by VPNs. *A Virtual Private Network (VPN) is a logical network that is established on top of a physical network like the Internet in order to provide the behavior of a dedicated network with private lines to the users of the VPN* [Brown00]. Essentially, a VPN is a private data network that uses a non-private data network to carry its traffic. The most ubiquitous, least expensive non-private data network is the Internet, which is the perfect foundation for a VPN. The challenge in designing a VPN is often to provide the security of the traditional private self-administered corporate network over the non-private backbone. *Traffic Engineering* is the process of controlling how traffic flows through one's network so as to optimize resource utilization and network performance [Awduche99]. Traffic Engineering is needed in IPVPNs mainly because current Internet Gateway Protocols (IGPs) always use the shortest path to forward traffic. Using shortest paths, network resources are conserved but it could cause problems like congestion at some links because of overlapping of shortest paths from different sources or it could so happen

that the shortest path's capacity is not sufficient to serve the traffic from source to destination, while a longer path exists. Currently all IPVPN service providers have these problems. So, by performing traffic engineering in their networks, service providers can greatly optimize resource utilization and their network performance. They can even increase their revenue without large investment in upgrading network infrastructure. As per the recommendations of Internet Engineering Task Force (IETF), Traffic Engineering can be done using multiprotocol label switching (MPLS), constraint based routing (CBR) and enhanced link state IGP. So far the Internet has offered only best effort service. All traffic is processed as quickly as possible and no preferences are given to any type of traffic. Today there are more and more applications that need service guarantees in order to function properly and as we will see later service guarantees are also an important requirement of VPNs. Thus Traffic Engineering and Quality of Service guarantee have become major research areas in IPVPNs in providing better VPN services to the end users over an IP backbone.

A VPN tunnel encapsulates certain data packets (the original, or inner packet) into another data packet (the encapsulating, or outer packet) so that the inner packet is opaque to the network over which the outer packet is routed [Yuan01a]. In a VPN there exists a variety of ways that tunnels can be formed [Perlmutter00]. Those are IP tunnels (IP over IP, IPSec, GRE), ATM VCs, and MPLS. The peer VPN model is one where paths are computed on a hop-by-hop basis, but in overlay VPN model the network layer-forwarding path is not done on a hop-by-hop basis, but rather the intermediate link layer is used as a cut-through to another edge node on the other side of the public network cloud. The overlay model introduces serious scalability problems as network management to maintain routing information increases in direct proportion to the number of connected sites. A subtype of this overlay model is tunneling that allows tunnels between source and destination router, router-to-router, or host-to-host. Tunnels encapsulate source packets with a new header, and forward them into a tunnel with a destination address of the endpoint. Quality of Service in VPNs can be assured by employing either Pipe model or Hose model. In Pipe model, customer specifies the load between every pair of endpoints. In Hose model [Duffield99], customer specifies the aggregate traffic requirements. There are two

that the shortest path's capacity is not sufficient to serve the traffic from source to destination, while a longer path exists. Currently all IPVPN service providers have these problems. So, by performing traffic engineering in their networks, service providers can greatly optimize resource utilization and their network performance. They can even increase their revenue without large investment in upgrading network infrastructure. As per the recommendations of Internet Engineering Task Force (IETF), Traffic Engineering can be done using multiprotocol label switching (MPLS), constraint based routing (CBR) and enhanced link state IGP. So far the Internet has offered only best effort service. All traffic is processed as quickly as possible and no preferences are given to any type of traffic. Today there are more and more applications that need service guarantees in order to function properly and as we will see later service guarantees are also an important requirement of VPNs. Thus Traffic Engineering and Quality of Service guarantee have become major research areas in IPVPNs in providing better VPN services to the end users over an IP backbone.

A VPN tunnel encapsulates certain data packets (the original, or inner packet) into another data packet (the encapsulating, or outer packet) so that the inner packet is opaque to the network over which the outer packet is routed [Yuan01a]. In a VPN there exists a variety of ways that tunnels can be formed [Perlmutter00]. Those are IP tunnels (IP over IP, IPSec, GRE), ATM VCs, and MPLS. The peer VPN model is one where paths are computed on a hop-by-hop basis, but in overlay VPN model the network layer-forwarding path is not done on a hop-by-hop basis, but rather the intermediate link layer is used as a cut-through to another edge node on the other side of the public network cloud. The overlay model introduces serious scalability problems as network management to maintain routing information increases in direct proportion to the number of connected sites. A subtype of this overlay model is tunneling that allows tunnels between source and destination router, router-to-router, or host-to-host. Tunnels encapsulate source packets with a new header, and forward them into a tunnel with a destination address of the endpoint. Quality of Service in VPNs can be assured by employing either Pipe model or Hose model. In Pipe model, customer specifies the load between every pair of endpoints. In Hose model [Duffield99], customer specifies the aggregate traffic requirements. There are two

approaches for establishing VPN tunnels in Pipe model: customer premises equipment (CPE) based and Network based [Cohen00]. In CPE based approach, tunnels are established only between the CPE devices (mainly border routers); where as in the network-based approach, tunnels are established between the routers of core network. In case of CPE based IPVPN, security may be applied from end to end. As the traffic is IP traffic, the enterprise has the flexibility of adding and removing tunnels dynamically, instead of requiring the service provider to configure permanent virtual circuits (PVCs) within the network. This allows the enterprise to have the flexibility of a fully meshed network without having to pay for the numerous PVCs needed in frame relay or ATMs. As a result, the enterprise would bring up tunnels for video calls, overnight backups, file transfers, etc. and take them down when not needed. In Network based approach, we have the tunnel endpoints within the ISP's core network, i.e. the routers in the core network can be strategically activated to make the path optimal between any two border routers or Customer edge routers, passing through core routers or provider routers. The migration to the integrated voice, data and multimedia applications over the VPN introduces new challenges in supporting predictable communication performance. Multimedia applications have an urgent need of Quality of Service metric that should include delay, delay jitter, cost, and other parameters. QoS routing identifies paths that meet the QoS requirements and selects one that leads to high overall resource efficiency. The QoS requirements of a point-to-point connection can be specified using link constraints like bandwidth requirement or path constraint like delay for the entire path. The QoS enabled tunnel path finding in a VPN is *NP-hard*. That is there is no known polynomial time algorithm for solving this problem.

As private networks built on using dedicated leased lines offer guaranteed bandwidth, security, and latency, users demand similar guarantees from the IPVPNs. The second generation IPVPNs focused on security ignoring QoS guarantees. However, the recent IP-technologies like MPLS, Diffserv, Resource Reservation Protocol (RSVP) deal with QoS issues. Quality of Service issue is becoming a significant challenge for the VPN service providers as VPN users want to have real time applications such as IP telephony, interactive games, teleconferencing, videos,

and audios etc., over their VPN connections. The better VPN will be the one that provides quality of service guarantees to highest number of end users, and not the one that provides equal allocation of its resources to all active traffic connections. In order to achieve statistical multiplexing in the providers' network, and thus increase the utility of the underlying network and give the VPN users some economical benefits, the bandwidth allocated to each tunnel should be dynamically adjusted [Garg00]. The user may not be able to specify traffic accurately in the service level agreement (SLA). Hence, the admission control and tunnel setup must be supplemented with finer levels of dynamic control over the bandwidth allocation to maximize the network usage, and user satisfaction. Two major problems exist in this scenario, i.e. how to allocate the bandwidth within a single VPN and how to allocate the bandwidth in the 'core' network, which is simultaneously used by a number of VPNs and by the public Internet. So, the problem of optimal allocation of available bandwidth resources of the network in order to achieve maximum performance from the virtual private networks – both in terms of the user satisfaction and in terms of the network utilization has become an active area of research.

High availability of network connections is a key challenge to service providers to increase their revenue. When a path fails, the service provider must quickly re-establish another path so that the user can continue its VPN connectivity without interruption. Restoration mechanisms help in this regard. In a VPN tree, if any edge fails, this would definitely disrupt the service unless a backup path was established to reconnect the tree. A restoration algorithm selects a backup path so that the traffic disrupted by failure of a tunnel can be rerouted via backup paths [Kodialam03].

There exist different architectures for different VPN deployment scenarios. Deploying VPNs, especially in a large-scale environment, needs a large amount of planning. Corporations want to control and manage their VPN according to their own control objectives and management strategies. Sometimes they also want provider-independent control, say for example, to implement sophisticated access control mechanisms that can prevent unauthorized access to certain partitions of the network.

Furthermore, VPN traffic management responsibility could be moved from provider to the customer edge that enhances the chances of broadband VPN services.

Little has been done to provide QoS support in a VPN environment that can support roaming users [Karnouskos00a]. These users are those who often move from one network to another. The main problems that the VPN designers face in supporting mobile users are the technology integration. These traveling VPN users must be supported in small groups or separately. Also the tunnel that is to be formed between source and destination should guarantee maximum quality of service requirements. VPNs are designed mostly with static users in mind and little has been done to integrate mobile users or to provide mobile user support after their deployment. The problem here is that the availability of information and communication services varies from place to place and from time to time. The major requirement here is that of dynamic reconfiguration of the new environment and customization. IP being a best effort network, there may not exist one solution to all the problems of QoS support, because the application-specific state of QoS should be frequently changeable or easily deployable when needed, while the conventional network is simple and stateless. Now, recent research has shown that programmable *Active Networks* may be a better solution to guarantee QoS in the mobile VPNs [Karnouskos00a].

1.2 Contribution of the Thesis ^{Full}

The major contributions of this thesis are on Traffic Engineering and QoS issues in IPVPNs. The first part of the work is on optimal tunnel path finding. Nowadays open shortest path first (OSPF), and border gateway protocol (BGP) are being considered for providing some sort of QoS guarantees like performing load balancing, and automatically recovering from failed access links etc. Even, multi-constrained routing algorithms or heuristics can be used to satisfy multiple independent QoS constraints. These routing algorithms have not been tried out with IPVPNs for reducing VPN tunnel maintenance cost. Also these are complex and time consuming routing algorithms. Our approach for handling optimal VPN tunnel path

finding is based on an intelligent search technique and an optimization technique. Here, we have addressed the problem of determining a layout of VPN tunnels for an Intranet VPN taking into consideration the cost of links, which is expressed in the form of QoS parameters and the nodes have heuristic values depending on their suitability to provide QoS. Our approach will not activate any core routers as tunnel endpoints in a CPE based approach. A heuristic function and search algorithm are used to find the optimal route. It solves the problem of NP-hardness of minimal cost VPNs (MC-VPNs). This solution ignores the use of core routers as active nodes assuming that these are ISP routers and therefore are not available to the end user for configuration. In the later part, as an extension, Optimization was used to find out which of the core routers can be activated taking into consideration the maximum allowed number of activating core routers. Our approach gives better services at the expense of computational cost involved in computing the heuristic estimation. Also the results showed improved performance over other well-known heuristics like active shortest path heuristic (ASPH) and active double tree heuristic (ADTH) [Cohen00] available in the literature.

96 yes, what to use in VPN?

cost/performance?

The second part of the work is on dynamic bandwidth management in IPVPNs. Here, bandwidth-resizing algorithms for Intranet Virtual Private Network scenario using software agents have been proposed. These algorithms are designed keeping in view of efficient network (ISP's) bandwidth utilization and the different degrees of urgency between the users. These algorithms are user oriented i.e. the way the link bandwidth or the bandwidth allocated to tunnels over the link is partitioned depends on the user's utility. Each sudden change in the users requirement is addressed by a linear change in the partitioning parameter (α). Bandwidth resizing algorithms are developed in a distributed platform using Unix threads and communication APIs. The synchronization between processes and agents are implemented by using Unix conditional variables and Mutexes. Agents were tied to the concerned node in the topology with TCP/IP sockets. These algorithms are completely new and the results obtained from the simulation run of these show that VPN service providers can satisfy more number of users with quality of service guarantees and also at the same time they can improve upon their revenue.

non-linear

J
The third important contribution is on survivability of VPN connections. The restoration techniques can be categorized into different types like *link-based*, and *path-based*. Link restoration finds alternate paths between the nodes of the failed link. The optimal link restoration approach may involve circuitous restoration routes. Path restoration overcomes this problem by rerouting flow between the source and destination pairs affected by a link failure. The incorporation of restoration leads to new QoS enabled *traffic-engineering* problem. New active path heuristic (*APH*) and backup path heuristic (*BPH*) for finding restorable paths for IPVPNs with QoS guarantees have been proposed. The *APH* and *BPH* produced a better solution to MC Restorable VPN problem. The restoration algorithms are path based. These algorithms are compared with well-known approximation algorithms for STP, and it is found that the results are close to optimal performance. In this part, the work also → *How much close?* highlights restorability in multicast VPNs. Here, the aim is to minimize the network resources used for establishing the multicast VPN connections, and hence to minimize the blocking probability in case of failures. The outcome of this work is an Integer Linear Programming ^(ILP) formulation and its' equivalent heuristic. The ILP ← formulation is solved using CPLEX. The heuristics are simulated and the results are compared with the well-known SPH algorithms available in literature to solve Steiner tree problems (STP).

The fourth contribution is on development of a layered framework for IPVPN deployment. The problems in providing an IPVPN solution could be seen at different levels like Planning, Implementation, and Management. The work in this part analyzes a viable solution addressing the above issues and challenges meeting the objectives from a customer managed VPN solution. A VPN testbed using the available VPN hardware and software was built. IPSec, GRE tunneling and the CA authentication mechanisms were implemented. The layered framework was tested over client server architecture and suitable recommendations are made for the VPN implementers.

The fifth significant contribution of this work addresses building mobile VPNs with QoS guarantees. The Internet as of today does not support secure

communication between groups of mobile VPN users with minimal effort. The notion of place oriented VPNs (PO-VPNs) [Karnouskos00b], an alternative form of VPNs whose aim is to provide an infrastructure for groups with special characteristics like short time to live, low population etc. have been considered. These PO-VPNs can be deployed on top of existing VPNs and offer customized services in a flexible, interoperable and cost effective way. New algorithms for active QoS routing have been proposed for these users. These algorithms are implemented using Unix socket API. The path thus computed for a VPN tunnel between a mobile user and corporate office will satisfy the QoS requirements as demanded by the user's SLA. To fulfill the requirements of mobile user group, a virtual environment with the help of *Active Networks* and *Intelligent Agents* have also been proposed. An alternative to active networks has also been suggested using distributed resource management architecture.

1.3 Structure of the Thesis

This thesis is organized into 8 chapters with a reference section at the end. Chapter 2 gives an overview of Virtual Private Networks. It describes different VPN architectures available over the Internet backbone, pros and cons of IPVPNs, different VPN requirements, and the various VPN protocols. This chapter concludes with a summary of suggestions on the suitability of these protocols for different VPN architectures.

Chapter 3 first gives an overview of Internet architecture, and ISP networks that are the backbone for deploying IPVPNs. Next, Quality of Service in IP Virtual Private Networks from both end user and service provider's perspective are discussed. Various methods of guaranteeing QoS in IPVPNs like Differentiated Services; Traffic Engineering schemes for IPVPNs like ATM, MPLS, and Constraint Based Routing are explained. Finally, the Policy based Quality of Service guarantee approach for IPVPNs is discussed.

Chapter 4 describes a review of the work done till date for computing optimal tunnel path, and defines the Minimal Cost VPN problem. Heuristic algorithms are also proposed in this chapter to compute optimal tunnel path that is hard to approximate in polynomial time. First a CPE based solution is presented for the MC-VPN problem, and then it is optimized using another heuristic. It also includes the analysis and evaluation of tunnel path finding algorithms.

In Chapter 5, dynamic bandwidth management in IPVPNs, more specifically the bandwidth reallocation and path rerouting schemes are discussed. It also presents the research that has been done so far in optimizing the bandwidth allocation. Novel algorithms for dynamic bandwidth management are proposed using software agent concepts considering the user utilization and user satisfaction. It also includes the analysis and evaluation of bandwidth management algorithms. Several results are presented to demonstrate the achievement of objectives. Simulation output are analyzed to show how the system will perform in different scenarios.

Chapter 6 discusses IPVPN unicast restoration mechanisms based on backup path sharing. This chapter also includes the definition of Multicast VPN restoration problem (Multicast MC-VPN). It models the Multicast MC-VPN problem using Integer Linear Programming, and another alternative heuristics. It also analyzes the simulation results of proposed algorithms.

In Chapter 7, the challenges behind deployment of VPNs are discussed and the need for a layered framework is proposed. Also, it discusses active networks and agent technology and their use in building mobile VPNs with QoS guarantees. It also proposes algorithms for building an IPVPN to support mobile users in a group with QoS guarantees.

Finally, in Chapter 8, we conclude the thesis and summarize the main contributions and list possible extensions of the work.

Chapter 2

Virtual Private Networks-An Overview

2.1 What is a Virtual Private Network?

The broadest definition of a VPN is “any network built upon a public network and partitioned for use by individual customers” [Fox99]. This results in public frame relay, X.25, and ATM networks being considered as VPNs. These types of VPNs are generically referred to as Layer 2 VPN. The emerging forms of VPNs are networks constructed across shared IP backbones, referred to as “IPVPNs”. An IPVPN [Gleeson00] connects the components and resources of one network over another IP network. VPNs accomplish this by allowing the user to tunnel through the Internet in a manner that lets the tunnel participants enjoy the same security and features formerly available only in private networks. This is depicted in figure 2.1.

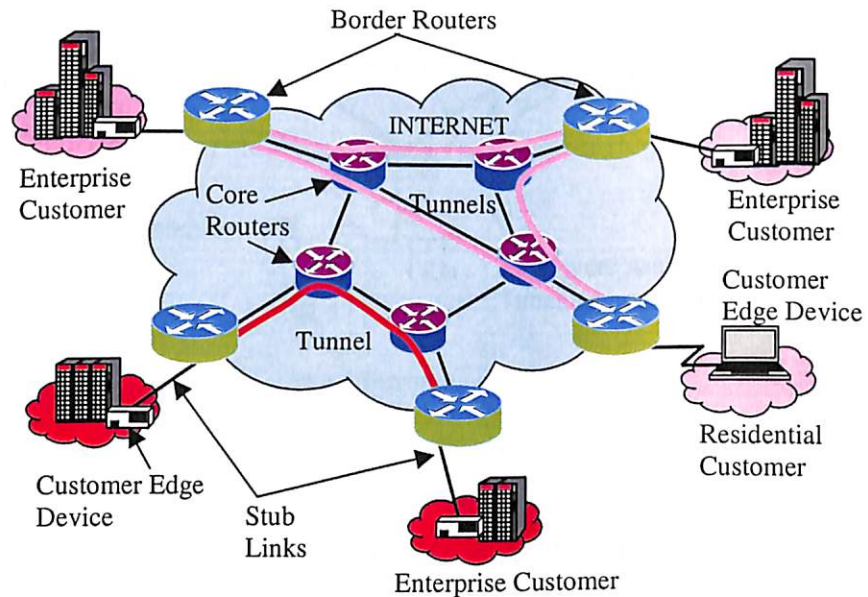


Figure 2.1 A Virtual Private Network

VPNs were first introduced to a wide customer base in the early 1980s, typically offering wide area connectivity to corporations. Prior to this, corporations were using dedicated leased line services for their business. These required either over-provisioning of resources or suffered from insufficient bandwidth availability to meet peak demands. Early VPN services were deployed on broadband networks using permanent virtual circuits (PVCs) over ATM or Frame Relay.

In an IPVPN, dial-up connections to remote users and leased line or Frame Relay connections to remote sites are replaced by local connections to an Internet service provider (ISP) or other service provider's point of presence (POP). A VPN allows a private intranet to be securely extended across the Internet or other network service, facilitating secure e-commerce and extranet connections with business partners, suppliers and customers. There are three main types of VPNs:

- **Intranet VPNs** allow private networks to be extended across the Internet or other public network service in a secure way [Quiggle01]. Intranet VPNs are sometimes referred to as site-to-site or LAN-to-LAN VPNs. This is shown in figure 2.2.

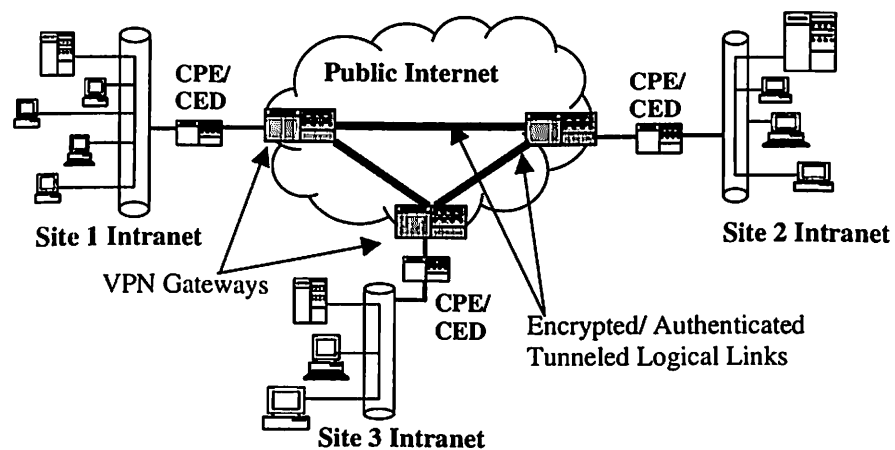


Figure 2.2 An Intranet VPN

- **Remote access VPNs** allow individual dial-up users to connect to a central site across the Internet or other public network service in a secure way (figure 2.3) [Davis01]. These VPNs are also sometimes referred to as dial up VPNs.

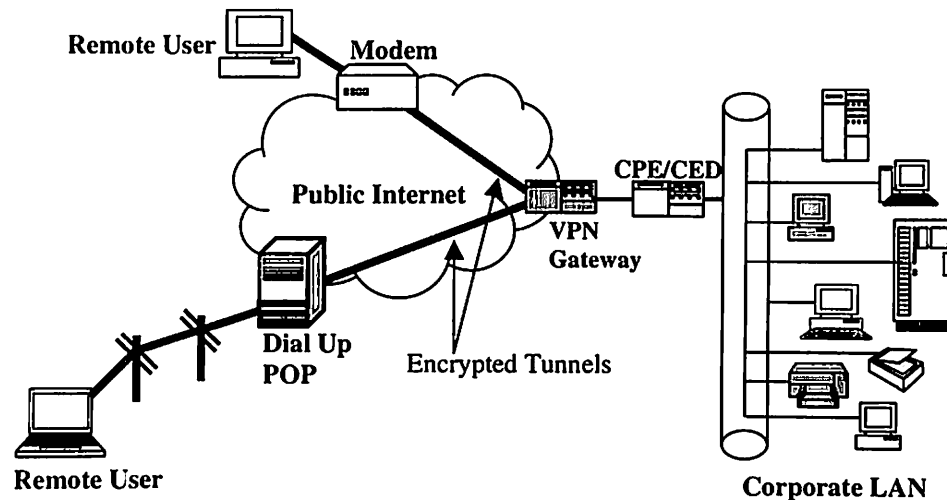


Figure 2.3 A Remote Access VPN

- **Extranet VPNs** allow secure connections with business partners, suppliers and customers for the purpose of e-commerce [Quiggle01]. Extranet VPNs are an extension of intranet VPNs with the addition of firewalls to protect the internal network.

All of these VPNs aim to provide the reliability, performance, quality of service, and security of traditional WAN environments at a low cost and more flexible service provider connections. VPN technology can also be used within an intranet to provide security or to control access to sensitive information, systems or resources. For example, VPN technology may be used to limit access to financial systems to certain users, or to ensure sensitive or confidential information is sent in a secure way. The different VPN types may also be described as follows:

- IP tunnels between a remote user and a corporate firewall with tunnel creation and deletion controlled by the user's computer and the firewall.
- IP tunnels between an Internet service provider and a corporate firewall with tunnel creation and deletion controlled by the ISP.
- IP tunnels between sites over the public Internet, or over a service provider's IP network that is separate from the public Internet.

2.2 A Security Services Taxonomy

For VPNs, the tunnel endpoints are where authentication and access control decisions are made, and where security services are negotiated and rendered [Yuan01a]. The selection of these tunnel endpoints and the rationale behind them are thus critical in the design of a VPN solution. In reality there are three possible kinds of security service endpoint locations. First, the endpoint can be at the end host itself, where the data originates or terminates. Second, the endpoint can be at the corporate LAN gateway device, where traffic has been aggregated. Third, the endpoint can be located outside the corporate network, within the ISP's infrastructure (POP), sometimes referred to as "in the cloud".

A VPN tunnel has two endpoints. Six types of security models can be derived from the possible combinations of the different locations. These are depicted in figure 2.4.

In the End-to-End model, the tunnel goes from one end system to the other. Hence, security service is negotiated and rendered at the source and destination of the communication. This scenario presents the highest level of security because data always travels securely in any segment of the network, be it public or private. However, as the total number of end systems rise, it becomes more difficult to manage the larger number of security services required by these end systems, unless the security service has only local significance and each end host is independent of the others. This security model is most often seen in higher layer implementations, as is the case with, for example, the Secure Socket Layer (SSL). Such higher layer implementations are generally not considered tunneling.

In the End-to-LAN model, the tunnel starts from an end system and terminates at the perimeter of the LAN on which the destination host resides. A VPN device located at the network perimeter is responsible for negotiating and rendering the security service on behalf of the other end system. Here, the security of a large number of devices on the corporate network can be managed at a single point, making it much easier to scale. Most remote access VPNs are implemented in this model.

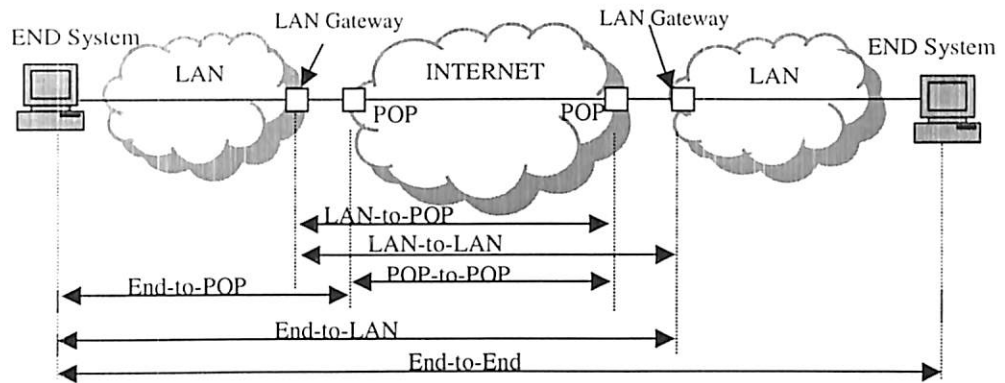


Figure 2.4 Six Tunneling models for VPNs [Yuan01a]

In the End-to-POP model, the tunnel starts from a host and ends at the service provider's POP. A VPN device, or VPN functions available in an ISP's POP device, is responsible for negotiating and rendering the security service on behalf of one of the destination end systems because the POP is the means by which the destination end system accesses the Internet. Data delivery from the POP to other end system must be secured either through its physical infrastructure or through a separate secure tunnel. This approach reduces the burden of deploying VPN devices or functionalities on the customer premises, i.e. the corporate network. Remote access VPNs can also use this model.

In the LAN-to-LAN model, both hosts use VPN devices situated at the corporate network perimeter. Here, end systems do not need to have VPN functionalities. Intranet VPNs use this model where the complexity for managing VPNs is drastically reduced.

In the LAN-to-POP model, tunnels are formed between a VPN device at customer side to a VPN device or a POP device with VPN functions at the ISP side. Currently, none of the VPN architectures use this model.

Finally, in POP-to-POP, both VPN devices are situated in ISP's network. Here, the security service is entirely transparent to the customers. This model is also

known as Network based VPN model. The major advantage in this model is the ability of the service provider to provide value-added services to customers without altering the customers' network infrastructure.

Out of all the above tunneling models, End-to-LAN and LAN-to-LAN models are being used the most to provide customer premises equipment (CPE) based VPN solution. However, the POP-to-POP or network based VPN model is attracting increasing attention as in this ISPs can provide value added services to customers. Of course, this will be at an extra cost.

2.3 VPNs Based on IP Tunnels

VPNs based on IP tunnels encapsulate a data packet within a normal IP packet for forwarding over an IP-based network [Brown00]. This is depicted in figure 2.5. The encapsulated packet does not need to be IP, and could in fact be any protocol such as IPX, AppleTalk, SNA or DECnet. The encapsulated packet does not need to be encrypted and authenticated; however, with most IP based VPNs, especially those running over the public Internet, encryption is used to ensure privacy and authentication to ensure validity of the sender responsible for sending the data. VPNs based on IP tunnels are mainly self-deployed; users buy connections from an ISP and install VPN equipment that they configure and manage them, ^{self-dep} relying on the ISP only ^{themselves} for the physical connections. ISPs also provide VPN services based on IP tunnels. These are usually fully managed services with options such as Service Level Agreements (SLAs) to ensure Quality of Service (QoS) guarantees. VPNs based on IP tunnels provide the following benefits:

- Reduced telecom costs, as dedicated and long-distance connections are replaced with local connections to ISPs.
- Greater flexibility in deploying mobile computing, telecommuting and branch office networking.
- Easier e-commerce and extranet connections with business partners, suppliers and customers.
- External Internet access, and internal intranet and extranet access can be provided using a single secure connection.

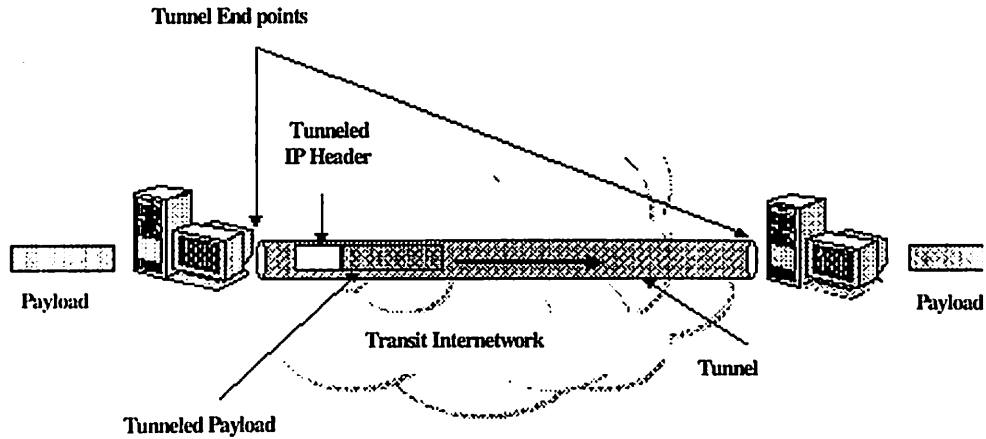


Figure 2.5 Tunneling

The main disadvantage of VPNs based on IP tunnels is that QoS levels may be erratic and are not yet as high as alternative solutions [Gleeson99]. Also, for VPNs based on the public Internet, higher levels of security such as authentication and data encryption are essential to ensure integrity and security of data. The ISP connections used for VPNs need not be protected by a firewall as data is protected through tunneling, encryption, etc. Also, one can use separate ISP connections for general Internet access and VPN access, or one can use a single connection with a common router with a VPN device and firewall in parallel behind it. In some cases, devices that integrate one or more of these functions can also be used.

How tunneling is secured

2.4 Merits and Demerits of IPVPNs

VPNs as well as the traditional telecommunication approaches are aimed at boosting productivity by extending the reach of the company's resources. These resources include hardware (computers, LANs), electronic services (file service, database access) as well as human resources (voice mail).

In order to extend the reach of a company's Intranet(s), a VPN over the Internet promises two benefits: *cost efficiency* and *global reachability*. However there

are three major concerns about VPN technology: security, manageability and performance.

Security: In order for Virtual Private Networks to be private, the transmitted data must be encrypted before entering the Internet, since the Internet is considered an untrusted network. Everybody can connect to the Internet and there is no guaranty that participants stick to any policy or rule. However, protecting the traveling data will not protect the information inside the Intranet from unauthorized access. Therefore, more elaborate VPN architectures must include additional protection such as firewalls and user authentication mechanisms.

Management: A company's telecommunication requirements and equipments evolve at a high speed. The VPN management must be able to cope with these changes avoiding high expenses. VPNs are connected to many different entities. These entities include the company's physical network (eventually featuring unregistered IP addresses), the company's security policy, the company's electronic services, the company's ISP(s) to mention a few. These entities evolve constantly and are hard to manage.

Performance: Since ISPs deliver IP packets still on a "best effort" basis, the transport performance of a VPN over the Internet cannot be predicted, it is variable. Current research is focusing on easing this situation. Furthermore, security measures (encryption and authentication) can decrease transport performance significantly. This also points out two management problems: The clients must be enabled to (1) measure the performance and (2) customize the VPN (e.g. the security options) in order to optimize it.

2.5 Features of a VPN

The right VPN solution must allow freedom for authorized remote clients to easily connect to corporate local area network (LAN) resources and must also allow remote offices to connect to each other to share resources and information. It must

also ensure the privacy and integrity of data as it traverses the public Internet. Therefore, at a minimum, a VPN solution should provide the following:

Availability: Availability applies to both uptime and access time. It does a user no good if he or she has authorization to access the corporate servers 24 hours a day, 7 days a week but cannot get them due to network problems. Unfortunately, many problems related to networks are beyond the ISP's control. A frame relay or ATM VPN guarantees availability from ISP but it is hard to achieve in Internet VPN.

Control: It is believed that some executives fear about leaving the control of their corporate VPN to their ISP that may lead to possible security breaches. But, in practice, managed VPN services (Network based VPN) can be of great value to a company due to the training, expertise, careful monitoring, and altering utilities or services provided by the service provider.

Compatibility: To use VPN technology and Internet as the transport medium, a company's internal network protocol architecture must be compatible to the Internet's native IP. Hence, if a company's internal network is running say, IPX or SNA protocols as network layer protocols, then it must have a gateway that can convert these into IP standard. But, definitely it adds a layer of complexity to the company's network.

Security: As the VPN is not the company's private network; others can intercept, collect and analyze data. Security encompasses everything with a VPN, from encryption using DES, RSA, etc. to authentication services using digital signatures and certificate authorities.

Interoperability: It is extremely difficult to choose a VPN solution including hardware and software, as whole suits of vendor's products are available that offer VPN hardware, software, encryption, and authentication schemes. The deciding factor is where the VPN fits in the organization like end user-to-end user interoperability, or LAN-to-LAN VPN connectivity. This will help decide on vendor

manufactures, and software suppliers. The International Computer Society Association (ICSA), a security assurance firm defines Internet Protocol Security (IPSec) standards that will most likely be well-accepted standard for VPN products applicable to different platforms.

Reliability: When a company's VPN is outsourced to an ISP, it will have to rely on the ISP, and even will have to live really at the mercy of that ISP. If the company's VPN goes down, they have no control to monitor or fix the situation other than waiting for something to be done by the ISP.

Data and User Authentication: VPN authentication consists of data and user authentication. Data authentication ensures that the message has been sent in its entirety and has not been altered in any form. User authentication is the process of allowing user access to the network. Before external users enter company's internal network, a VPN requires secure authentication and user verification.

Traffic Overhead: In all kinds of technologies there are trade-offs: speed versus performance, security versus flexibility. VPNs are no exception from this. If the VPN device encrypts every single packet that goes out, one can imagine the kind of CPU processing power required. If it encapsulates every packet, it may be increasing the packet size, hence consuming more bandwidth over the outgoing link. Hence, a bigger pipe may be required, as the ISP link will be overutilized. To minimize this, the VPN should give a choice of deciding what kind of data needs to be encrypted, what kind of data needs to be authenticated, and what kind of data can freely flow untouched. For example, general broadcasts, multicasts and similar traffic does not need to be encrypted; however it needs to be authenticated.

Maintenance: Based on whether the company's VPN is CPE based or Network based, one will have to decide upon who will maintain the VPN. If the VPN is CPE based, the corporate needs trained security IT staff to maintain it. If the VPN is Network based, now the maintenance is the responsibility of the ISP.

Nonrepudiation: It is the process of positively identifying the sender in such a way that the sender cannot deny it. VPN must guarantee this feature. This has broad implications for suppliers, retailers, vendors, and key trading partners. Nonrepudiation must exist in order for electronic commerce to become a viable option in VPNs. The use of certificate authorities (CA) and digital signatures in VPNs ensure this.

2.6 VPN Protocols

The term "private" in "Virtual Private Network" means encryption and authentication. The term "virtual" means that the protocol makes a physically discontinuous system appear as one network to the higher protocols. It seems to be the first choice to place such a VPN protocol at the lower layers in the ISO-OSI reference model, namely at the data link layer and the network layer. However, higher protocol layers can be an option, too.

VPN Protocols in the OSI Reference Model:

A list of the common VPN protocols starting from OSI layer 2 (data link) along with the protocols' key features is given below:

Point-to-Point Tunneling Protocol (PPTP)[Hamzeh99]:

Microsoft extended the ubiquitous Point-to-Point Protocol (PPP) by running it as the inner protocol with the Generic Routing Encapsulation (GRE) protocol. PPTP emerged out of Microsoft's client/server-LAN experience. Thus, it uses the Remote Access Services (RAS) of the client for user authentication. PPTP is limited in usage. It offers remote connections to a single point. It does not support multiple connections nor does it easily support network-to-network connections. PPTP's security is also limited. It does not offer protection from substitution or playback attacks, nor does it provide perfect forward secrecy. PPTP has no clear mechanism for renegotiation if connectivity to the server is lost.

Layer 2 Tunneling Protocol (L2TP)[Townesley99]:

This protocol also guides PPP over an IP network, but is a simpler version of GRE. L2TP came out of the router-ISP-engineering community. Thus it uses the Remote Authentication Dial-In User Service (RADIUS) at the ISP. L2TP is similar to PPTP and they both target the remote access scenario. L2TP delegates security features toward IP Security (IPSec) that is described next. Besides that it suffers from the same drawbacks as PPTP.

Internet Protocol Security (IPSec)[Kent98]:

IPSec evolved from the IPv6 movement and is promoted as a standard by the IETF. It is located in OSI-layer 3. IPSec is a broad-based open solution for encryption and authentication on a per-packet basis. IPSec can securely encapsulate IPv4 packets and tunnel them from one firewall to another. Thus it is an optimum solution for trusted LAN-to-LAN VPNs (the branch office connection network scenario presented in figure 2.5). IPSec can ensure authentication, privacy and data integrity. It is open to a wide variety of encryption mechanisms. IPSec is application transparent and a natural IP extension, thus ensuring interoperability among VPNs over the Internet. Most of the router vendors and VPN hardware vendors support IPSec. Nevertheless, there are disadvantages of IPSec. IPSec is bound to the TCP/IP stack. IP addressing is part of IPSec's authentication algorithm. This is less secure than higher layered approaches and it is a problem in dynamic address environments, which are common to ISPs. IPSec requires a public key infrastructure, which is still subject to current research. IPSec does not specify a methodology for access control beyond simple packet filtering. Furthermore, IPSec's development is delayed by ongoing IETF committee infighting and is not explicitly supported by Microsoft. Appendix A describes an implementation of IPSec VPN using CISCO 7100 series routers over a testbed.

SOCKS v5 and SSL:

SOCKS v5 was originally approved by the IETF as a standard protocol for authenticated firewall traversal. When combined with the Secure Socket Layer (SSL) it provides the foundation for building highly secure VPNs that are compatible with any firewall. SOCKS v5 strength is access control. SOCKS v5 controls the flow of data at the session layer (OSI- layer 5). It establishes a circuit between a client and a host on a session-by-session basis. Thus it can provide more detailed access control than protocols in the lower layers without the need to reconfigure each application. SOCKS v5 and SSL can interoperate on top of IPv4, IPSec, PPTP, L2TP or any other lower level VPN protocol. A session layer solution does not have to interfere with the networking transport components, thus the clients are non-intrusive. SOCKS v5 provides plug-and-play capabilities including access control, protocol filtering, content filtering, traffic monitoring, and reporting and administration applications. On the minus side, SOCKS v5 decreases performance. Also, client software is required to build a connection through the firewall to transmit all TCP/IP data through the proxy server

The following table indicates what VPN purposes are best served by the different protocols.

Table 2.1 Different Protocols with VPN types

VPN Type	Protocol	OSI Layer
Branch Office Connection Network	IPSec	Network
Basic Remote Access	PPTP, L2TP	DataLink
Secure Remote Access	SOCKS v5, SSL	Session
Business Partner Networks	SOCKS v5, SSL	Session

Its' architecture is sketched in figure 2.6.

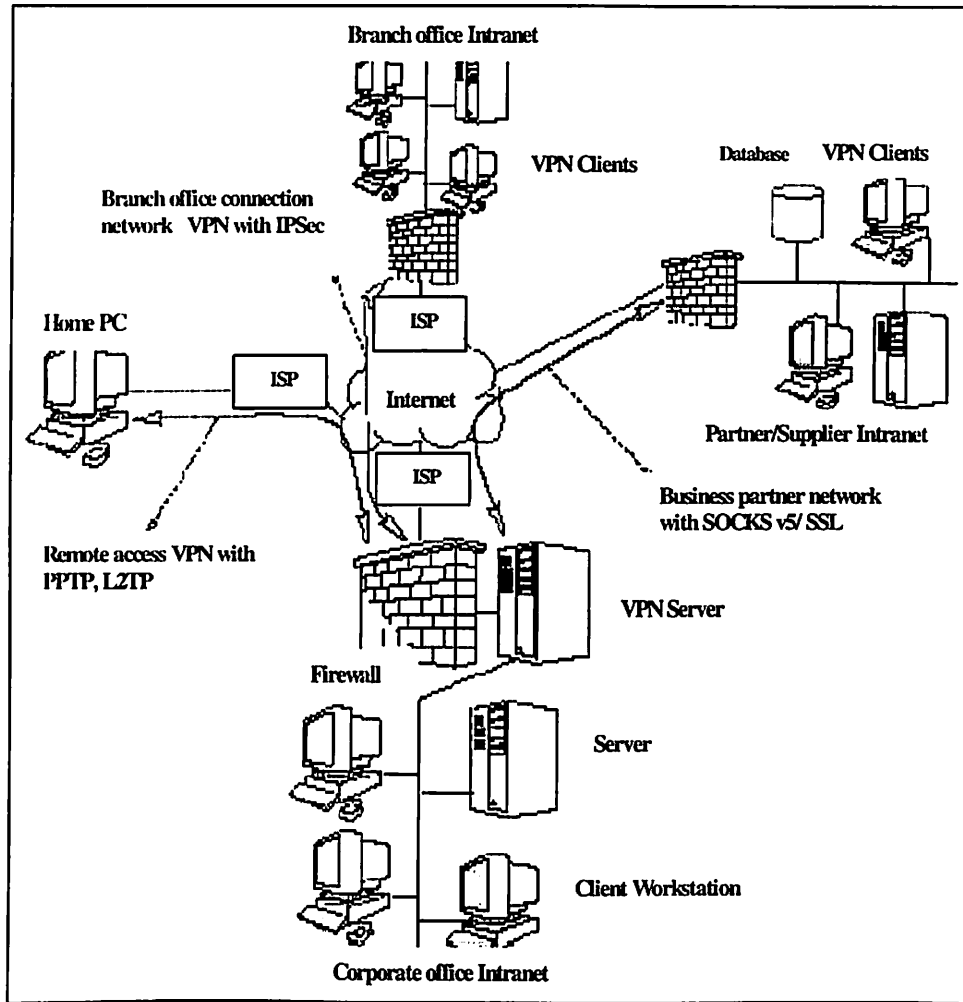


Figure 2.6 Different VPN Types with Protocols

2.7 Summary

In today's global market the availability of the company's electronic resources to the authorized personnel, customers and partners is believed to be a major competitive factor. However, remote dial-up and leased lines are expensive. Many companies therefore evaluate the deployment of virtual private networks over the Internet. Such solutions are price-effective and have a wide reach. Problems of such

IPVPNs are security, manageability and performance. For the simple VPN scenarios (remote access and trusted LAN-to-LAN networks) good protocols and solution exists and are in use. Nevertheless, there are many unresolved topics that continue to engage researchers in this area.

- The IETF working group is still arguing over group keys that are securely shared among the multicast user group for the multicast session in a multicast VPN, even though users may enter and leave the group at will [Perlmutter00]. Extensions to IGMP, and multicast routing protocol like core based trees are underway to support secure joins.
- Managing methodologies and tools for more complicated VPNs (Extranet Scenario) with a company wide security policy and fine-grained resource access control are still incomplete or missing [Yuan01a].
- Developing new models for improved VPN performance like Hose [Duffield99], optimal bandwidth reservations with multipath routing in Hose model [Erlebach04] are still under active research.
- Traffic Engineering and QoS considerations for IP traffic over arbitrary link layer technologies in MPLS VPNs are under IETF working group considerations. There are different ways to separate the routes between the two distinct MPLS-based VPNs. One such way is to maintain separate routing tables within a single routing process [Rosen00]. Another one is to create separate routing processes on the same physical device, each instance having its own routing table [Brahim00].
- New encryption methods for VPNs like Elliptic Curve Cryptography- Researchers have been studying these elliptic curve functions as replacements to standard exponential algorithms used in modern public-key cryptography [Brown00]. These elliptic curve cryptosystems can use smaller keys and are less computationally intensive. These can factor large numbers and can be used in Diffie-Hellman protocol, digital signatures, key exchange and confidentiality. These cryptosystems are also suited for low-resource available situations.

Chapter 3

Quality of Service in IP Virtual Private Networks

3.1 Internet Architecture

3.1.1 Introduction

The Internet is a collection of dissimilar networks over wide geographical boundaries that use certain common protocols and provide certain common services. TCP/IP protocol family is used to facilitate communication over the Internet. Although the protocol suit is TCP/IP, there are more members of this family than just Transmission Control Protocol (TCP) and Internet Protocol (IP) like Internet Control Message Protocol (ICMP), Internet Group Management Protocol (IGMP), Address Resolution Protocol (ARP), Reverse Address Resolution Protocol (RARP), and User Datagram Protocol (UDP). TCP and UDP are transport layer protocols responsible for end-to-end communication. IP and supporting protocols like ICMP, IGMP, ARP, etc. are used for connecting networks and gateways into a single virtual network called the Internet.

3.1.2 Internet Protocol - IP

IP - the fundamental protocol of the Internet is a layer 3 protocol. Its job is to permit hosts to inject packets into any network and have them travel independently to the destination (potentially on a different network). The IP layer provides a connectionless and unreliable datagram delivery service. IP makes its best effort to deliver an IP datagram to the specified destination, but there is no guarantee that the

datagram arrives. Each IP datagram is independently routed through the network possibly to avoid congestion. This is because IP is connectionless. These datagrams may arrive in a different order than they were sent, in which case IP layer at the receiving host should rearrange them before delivering to the destination application. Hence, packet routing is clearly the major issue at IP layer. If the application demands reliability, then it must be added by the upper layers of TCP/IP stack. If TCP is used as the transport protocol, then this reliability feature is provided by itself. In case of UDP, the application has to do this. The protocol between IP entities is best described with reference to the IP datagram format. Figure 3.1 describes the structure of IPv4 datagram that is the current IP standard.

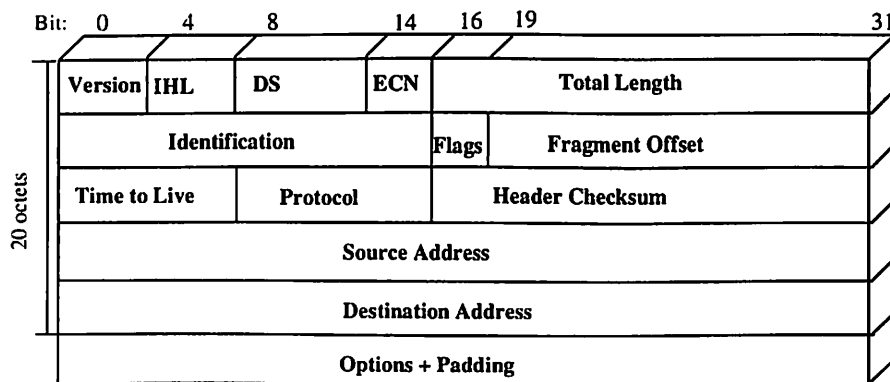


Figure 3.1 IPv4 Header

The fields are as follows:

Version: It indicates the version number, to permit evolution of the protocol; the value is 4. Version 6 is expected to be the successor to version 4, but it has yet to see widespread deployment.

Internet Header Length (IHL): This gives the length of IP header in 32-bit words. The minimum value is 5, for a minimum header length of 20 octets without options present.

Differentiated Services (DS)/Explicit Congestion Notification (ECN): Prior to the introduction of differentiated services, this field was referred to as the Type of Service field and specified reliability, precedence, delay, and throughput parameters. These parameters define the way for the IP packet to carry a notion of the quality of service desired. These values are used as a guide for the selection of the actual service parameters when the packet is transmitted through a particular network. This interpretation has now been superseded. The first 6 bits of the TOS field are now referred to as the DS field and the remaining 2 bits are reserved for ECN.

Total Length: It indicates the total length of the IP packet, including the header, measured in octets. An IP packet can have a maximum length of 65,535 octets, as there are a total of 16 bits in this field.

Identification: It is a sequence number assigned by the sender that, together with the source address, and user protocol, is intended to identify a datagram uniquely. It is used for assembling fragments of an IP packet. If the IP packet is too large for a particular router to handle, it may be divided into several smaller IP packets called fragments. The identification field helps to identify the various fragments.

Flags: It indicates whether this IP packet is allowed to be fragmented. If yes, there are more fragments to follow. If don't fragment flag is set, the router will drop the IP packet before fragmentation can occur. This bit may be useful if it is known that the destination does not have the capability to reassemble fragments.

Fragment Offset: It indicates where in the original datagram this fragment belongs, measured in 64-bit units. This implies that fragments other than the last fragment must contain a data field that is multiple of 64-bit length.

Time to Live: It specifies how long, in seconds, a datagram is allowed to remain in the network. It is similar to hop count. Each router is required to decrement this value as it routes the packet. The packet is dropped if this value ever reaches 0.

Protocol: It indicates what the payload is. Typically, the payload of an IP datagram is a packet for a higher layer protocol such as TCP or UDP. It is this layer that is to receive the data field at the destination. Hence, it identifies the type of the next header in the datagram after the IP header.

Header Checksum: It is a value computed by running a checksum function over the IP header only. Each router in the middle must check the integrity of the IP packet by verifying the checksum. Some fields in the IP header such as TTL may change during transit, so this checksum is reverified and computed again at each router.

Source Address: This contains a 32-bit global Internet address of the sender, generally consisting of a network identifier and a host identifier.

Destination Address: This contains a 32-bit global Internet address of the receiver, generally consisting of a network identifier and a host identifier.

Options and Padding: Options field carries any other information relating to how this IP packet is processed. If the options field does not end on a 32-bit boundary, the padding field fills it out. Hence, padding field is used to ensure that the datagram header is a multiple of 32-bits in length.

3.1.3 IP Addresses and Routing

IPv4 IP address is a 32-bit logical address used to identify a host over the Internet. It begins with a network number used for routing, followed by a local, network internal address. This encoding provides flexibility in assigning addresses to hosts and allows a mix of network sizes on an Internet. The different network classes supported by IPv4 are Class A, Class B, Class C, Class D (Multicast), Class E, and Broadcast. Class A can give few networks, each with many hosts. Class B can give medium number of networks, each with a medium number of hosts. Class C can give many networks, each with a few hosts. Class D is for multicasting and Class E is reserved for future use.

There are two types of devices at the IP level, hosts and routers. Any end user computer that connects to the network is called as a host. A host may have several local addresses and a single network address. An IP router is a dedicated computer that attaches two or more networks. It forwards the incoming packets to the next hop on the basis of network portion of the destination IP address. IP routers exchange network addresses as reachability information between them using different routing protocols, like Open Shortest Path First (OSPF), or Exterior Gateway Protocol (EGP), based on where in the network hierarchy the routers are located.

3.1.4 An Overview of ISP Networks

Internet Service Provider (ISP) networks consist of points-of-presence (POPs) and links interconnecting the POPs. Figure 3.2 shows a sample part of an ISP network. A POP consists of a combination of one or more access routers (ARs) that are connected to customers; border routers (BRs) are connected to other routers, hosting routers (HRs) are connected to web servers, and core routers (CRs) are connected to other POPs. Packets from an AR or BR or HR to other POPs must first be sent to a CR. The CR will then relay these packets to the CRs in other POPs. The network architecture of POPs is usually highly symmetric, as shown in the figure below. A large ISP may have around 30 POPs. Different POPs are normally connected in a ring topology to increase reliability.

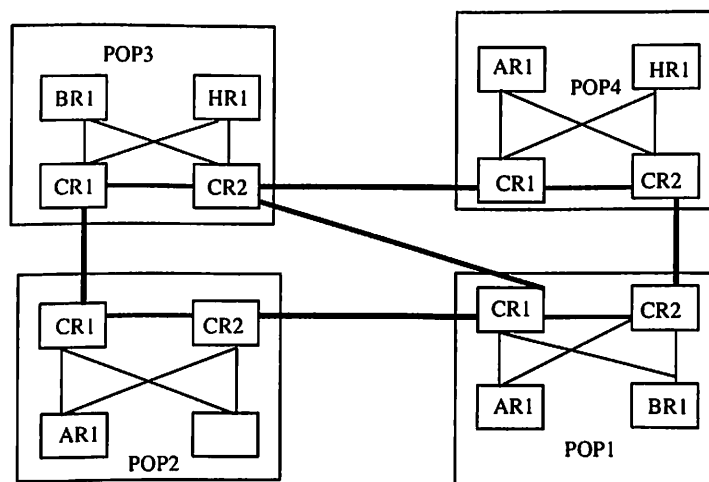


Figure 3.2 A sample part of an ISP network

3.2 Quality of Service in IP Virtual Private Networks

3.2.1 QoS from End-user and Provider's Perspective

Momentous changes in the way corporations do business and process information have been driven by changes in networking technology and at the same time have driven those changes. It is difficult to separate the chicken from the egg in this field. In the same way, the use of Internet has resulted in an increase in the total number of users and the traffic volume generated by each user. This, in turn has resulted in a need to increase the speed and efficiency of the Internet.

The Internet and the Internet Protocol (IP) were designed to provide a best effort, fair delivery service. It treats all packets equally under a best-effort scheme. As the level of traffic grows, and congestion occurs, packets are dropped by intermediate nodes more or less at random, and packet delivery is slowed down. In this scenario, no distinction is made in terms of the relative importance of any traffic or of the timeliness requirements of any of the traffic.

Nowadays the needs of user have changed. With the tremendous increase in traffic volume and the introduction of new real time, multimedia, and multicasting applications, the traditional Internet protocols are not sufficient. From users' perspective, traffic on IPVPN can be divided into two broad categories: inelastic and elastic. Typical thresholds for acceptable QoS for major Internet applications are listed in Table 3.1.

Table 3.1 Thresholds for acceptable QoS [Bouch00]

	Application	Minimum user requirement
Inelastic Traffic	Video	5 frames per second
	Audio	< 30% packet loss Latency < 400ms
	Interactive real-time multimedia	Delay < 200ms Jitter < 200ms
Elastic Traffic	Web page access	Latency < 11 seconds

Elastic traffic can adjust, over wide ranges, to changes in delay and throughput across an Internet and still meet the needs of its applications. Elastic applications include common Internet based applications, like e-mail, remote login, network management and web access. These applications do differ in the QoS requirements. For example, e-mail is insensitive to changes in delay, ftp is sensitive to changes in throughput and hence delay is proportional to file size, interactive applications such as remote login and web access are quite sensitive to delay.

Inelastic traffic does not adapt to changes in delay and throughput across the Internet. Voice, audio, and real-time multimedia are prime examples of this type. The requirements for these may include throughput, delay, delay variation, and packet loss. Meeting these requirements over Internet is difficult. A simpler way of providing these could be devising means to give preferential treatment to applications with more demanding requirements. To fulfill this, applications should state their requirements well in advance or on the fly.

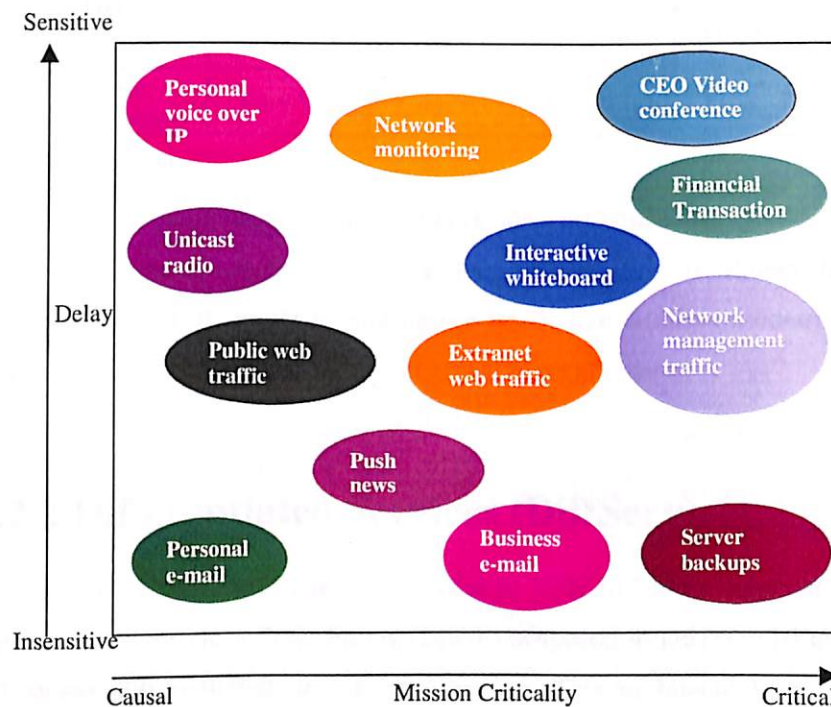


Figure 3.3 A Comparison of Application Delay Sensitivity and Criticality in an Enterprise

An alternative way to look at the traffic requirements of a corporation is shown in figure 3.3. Applications can mainly be classified into two types. One, that is delay sensitive, and the other that is mission critical. The former can be satisfied with timely delivery of packets, and the later can be satisfied by ensuring reliability.

From the ISP's point of view, guaranteeing quality of service in a multiservice environment requires different levels of their network performance. Table 3.2 gives an overview of typical application's requirements over the Internet from service provider.

Table 3.2 Bandwidth and Delay requirements of typical Internet applications

	Voice	File Transfer	Video Conference	Video Broadcast
Average bandwidth	Very low	High	Very high	Very high
Peak bandwidth	Low	High	Very high	Very high
Delay	Very high	Low	Very high	High
Delay variation	Very high	Low	Very high	Low

Design parameters of ISP network also influence the QoS parameters like delay, delay variation, loss, and random errors. This is shown in Table 3.3 [McDysan00b]. ISP should handle design issues like admission control, congestion control, packet scheduling, routing, and bandwidth allocation.

3.2.2 Differentiated Services (DiffServ)

Quality of Service can be provided to VPN traffic by aggregating traffic on Differentiated Services Code Points (DSCP) proposed in DiffServ [Blake98]. Figure 3.4 shows differentiated service architecture. This architecture was designed in response to the perceived lack of scalability of IntServ. In DiffServ, DSCP comprises of first six bits of the 8-bit Type of Service (TOS, or DS) field in the IPv4 header.

Instead of giving per-flow QoS, DiffServ assigns a behavior aggregate to each packet, as per its service requirement. Each behavior aggregate is identified by a single DSCP. Throughout a DiffServ domain, routers apply different per-hop forwarding behavior (PHB) to different behavior aggregates, i.e. classes of traffic. PHB is either queuing or scheduling behavior. So, in the interior of the network, with the help of DS code point [Nichols98], [Black01], the VPN traffic can be allocated a certain amount of node resources.

Table 3.3 Impact of Design Issues on the QoS parameter

Feature	Delay	Delay variation	Loss	Random errors
Propagation Delay	●			
Router Queue Architecture	●	●	●	
Link Rate	●	●		
Packet Size	●	●	●	
Buffer Capacity	●	●	●	
Resource Allocation	●	●	●	
Variations in Traffic Load	●	●	●	
Router and Link Failures			●	
Bit and Burst errors			●	●
Number of Routers Traversed	●	●	●	●

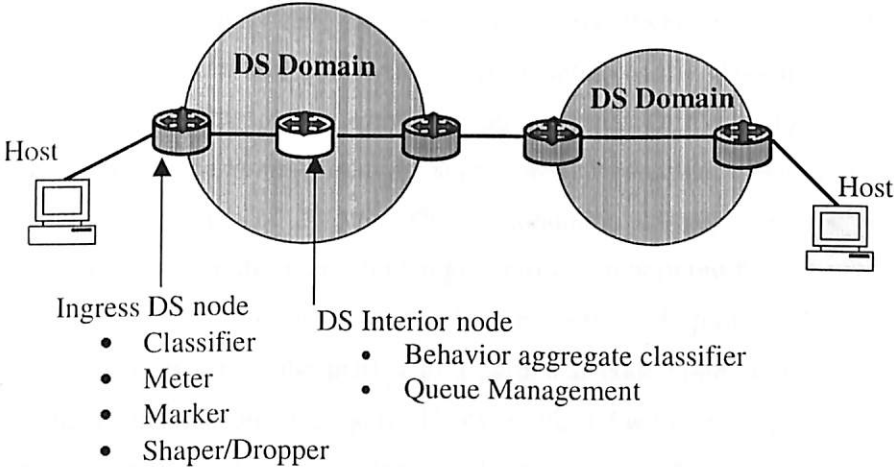


Figure 3.4 Differentiated Services Architecture

Buffer management and Packet scheduling mechanisms are employed to implement PHBs. The allocation of buffer and bandwidth resources to a behavior aggregate defines the forwarding treatment a PHB receives. The two forwarding types are Expedited Forwarding (EF) [Jacobson99] and Assured Forwarding (AF) [Heinanen99]. EF is a form of “premium service” that assures minimal delay, jitter, and packet loss, and assured bandwidth. This is achieved by keeping the arrival rate of packets at a node lesser than the output rate at that node. This is achieved by eliminating queuing for packets that violate this traffic profile. They are either dropped or delivered out of sequence. It is believed [Gleeson99] that in case of IPVPN, users may want Virtual Leased Line (VLL) type low-loss point-to-point connection for which EF seems to be an appropriate choice. To implement EF PHB, scheduling mechanisms like Class Based Queuing (CBQ) [Floyd95] or its variants [McKenny90], [Stoica97], [Stiliadis97] are all strong candidates.

Assured Forwarding (AF) offers different levels of forwarding assurances by having multiple classes and multiple drop precedences for each class. Each AF class receives a certain amount of bandwidth and buffer space at each intermediate node. Each class can have three drop precedences. These drop precedences indicate relative importance of the packet within an AF class. Packets with higher drop precedence values are discarded first when congestion occurs.

A VPN customer establishes Service Level Specifications (SLSs) with the ISP and also ISPs between themselves. These SLSs define traffic class and conditioning actions when a VPN tunnel spans through several DiffServ domains. Traffic classification can be done based on source and destination address, source and destination port, protocol ID etc. The conditioning actions could be metering, marking, shaping, and dropping (policing). Traffic is monitored by a meter and also checked for whether the classified traffic is meeting the right profile. The DS field of the IPv4 header is set by the marker to a particular code point, adding the marked packet to a DS behavior aggregate. Using a token bucket a shaper delays some packets until they can be safely released into the network. This is to ensure that the

traffic is in accordance to the traffic specification. If the flow violates the traffic specification, a policer drops excess packets from the flow.

Guaranteeing QoS to VPN tunnels using DiffServ requires special attention while mapping DSCP at the tunnel starting point. In Network based VPNs, the ingress router of an ISP might perform DiffServ classification and traffic conditioning as well as tunnel creation. If the tunnel is formed first, the ingress router can select the appropriate DSCP for the corresponding tunnel. If DiffServ processing is done first and then encapsulation, the DSCP of the inner IP header must be copied to the DSCP field of the outer IP header.

The DiffServ domains can be managed by agents known as Bandwidth Brokers (BB) [Nichols99]. A BB maps SLSs to concrete configurations of DiffServ routers, in particular to edge routers of a DiffServ domain. It keeps track of current allocations with the DiffServ domain, handles requests for new allocations, and interacts with peers to configure service classes that cross domain boundaries.

3.2.3 Traffic Engineering

Traffic Engineering is the process of controlling how traffic flows through one's network so as to optimize resource utilization and network performance [Awduche99]. Traffic Engineering is needed in IPVPNs because current ISP's Interior Gateway Protocol always use the shortest paths to forward traffic. The basic problem with this is that the shortest paths between different VPN tunnel ingress and egress pairs may overlap at some links, causing congestion on those links while links on the other possible alternate paths remain unutilized. It could so happen that the traffic from a source to a destination exceeds the capacity of the shortest path, while a longer path between these two routers is underutilized. Traffic Engineering provides the ability to move traffic flows away from the shortest path selected by the IGP and onto potentially a less congested physical path across the service provider's network. By performing traffic engineering on their networks, ISPs can greatly optimize

resource utilization and network performance. They can increase their revenue without putting up large investment in upgrading network infrastructure.

In the past, Traffic Engineering has been done on overlay models, i.e. running IP over an underlying connection-oriented network topology such as ATM or Frame Relay. In order to do Traffic Engineering effectively, the Internet Engineering Task Force (IETF) introduced MPLS [Rosen99], Constraint-based Routing [Crawley98], and an Enhanced Link State IGP [Li99]. These techniques exploit the economies of the bandwidth that has been provisioned across the entire network. They are briefly reviewed in this section.

3.2.3.1 IP-ATM Overlay Networks

Asynchronous Transfer Mode (ATM) is a connection oriented network topology that supports both constant and variable rate traffic. Figure 3.5 shows an IP over ATM overlay model. Here, ATM cloud is surrounded by IP routers. These routers communicate with each other by a set of Permanent Virtual Circuits (PVCs) configured across physical ATM network. It uses small, fixed size cells that can be switched at very high speeds. The cell header consists of Virtual Path Identifier (VPI) and Virtual Channel Identifier (VCI) used for switching purposes. These labels have only local significance and are remapped at each switch as required. The VPI and / or VCI refer to the Virtual Circuit (VC) over which the cell is being transmitted. To interconnect IP routers of particular VPNs, ISPs can establish meshes of PVCs that form a logical circuit. As this overlay model can explicitly route PVCs to precisely distribute traffic across all links, it supports Traffic Engineering. The wrong side of these overlay networks is that it requires management of two separate networks with different technologies that adds to increased operational complexity.

ATM was designed as a multiservice network that can support all types of service and hence be efficient, cost effective and flexible. It was standardized by the International Telecommunications Union (ITU). Due to this standardization effort, and also due to high cost and relative complexity, ATM is mostly deployed in

network backbones. It is not well suited as a public access network. This is exploited by Multiprotocol Label Switching (MPLS), described next, which adopts this model and associates resources with connections that are traversed through a connectionless network topology such as Ethernet.

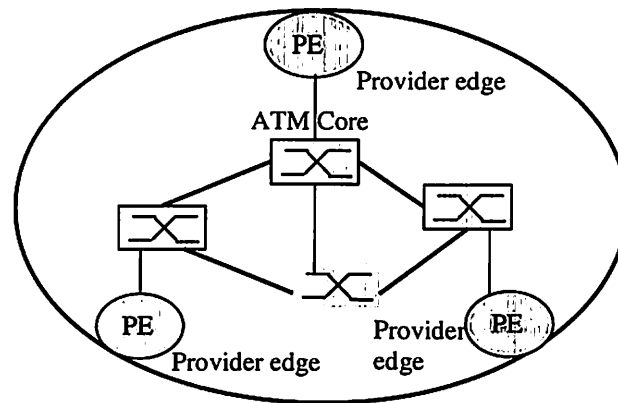


Figure 3.5 IP-ATM Core Physical Topology

3.2.3.2 Multiprotocol Label Switching (MPLS)

Multiprotocol Label Switching (MPLS) [Callon99, Rosen99] is an advanced method of packet forwarding that selects the next hop according to a short fixed length label in the packet header. It offers an efficient and robust solution to the problems of over-utilized paths in core ISP networks by establishing an explicitly routed label switched path (LSP) to handle a large volume of traffic that takes a particular route. This scenario is depicted in figure 3.6. MPLS operates over different link layer technologies. Instead of a forwarding mechanism to determine the route, it adopts Traffic Engineering and QoS based routing mechanisms.

At the ingress Label Switching Router (LSR) of an MPLS capable domain IP packets are classified into Forwarding Equivalence Classes (FECs) based on a combination of the information carried in the IP header of the packets and the local routing information maintained by the LSRs. An MPLS label of 4 bytes consisting of a 20-bit label field, a 3-bit experimental field (earlier known as Class of Service

field), a 1-bit label stack indicator and an 8-bit TTL field is then prepended to each packet as per their FEC. This label is inserted into a spare field in the network or data link header, or else the packet is encapsulated with a special purpose (shim) header. The packet is forwarded within the network core at the link label as per its' label and no further analysis of the packet header is required, allowing fast forwarding. Within an MPLS capable domain, an LSR will use the label as the index to look up the forwarding table of the LSR. The packet is processed as specified by the forwarding table entry. The incoming label is replaced by the outgoing label and the packet is forwarded to the next LSR or next hop as mentioned in the forwarding table entry. The path through which the packet moves from one LSR to another is called the Label Switched Path (LSP). Using constraint based routing with an utility tool, a path can be selected that offers the highest capacity with lowest congestion and meets VPN tunnel performance requirements. Once a path has been determined MPLS can be used to setup LSPs to carry VPN traffic.

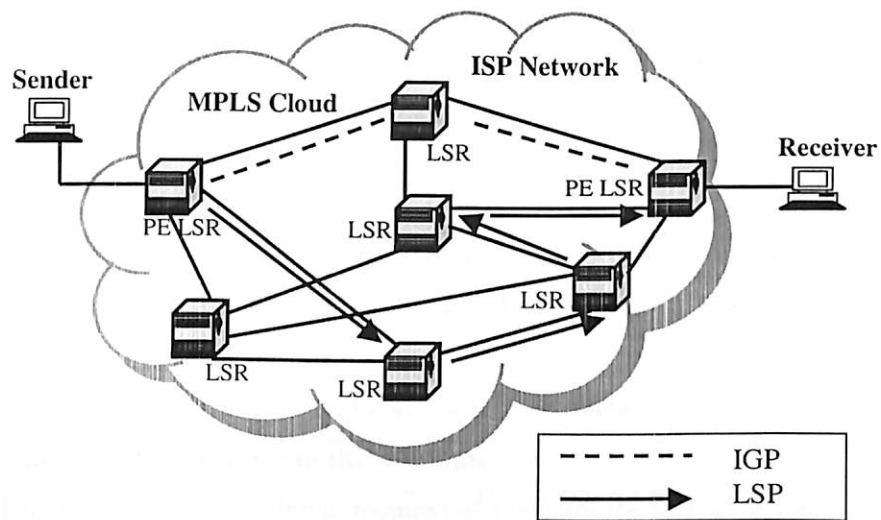


Figure 3.6 MPLS Traffic Engineering

MPLS uses some signaling protocol like Label Distribution Protocol (LDP) to set up LSPs. It can also use Resource Reservation Protocol (RSVP) for the same cause. A LDP is used to advertise assigned labels, in turn establishing LSPs throughout the network. There exists several variations of LDP [Callon99]. Topology

driven LDP [Andersson00], the conventional routing protocol does label distribution. Request driven LDPs allocate labels as per the Resource Reservation Protocol (RSVP) [Zhang93]. Traffic driven LDPs are triggered by arrival of data, like IP switching [Newman98] in which an IP flow prompts a connection establishment to carry the related traffic.

Again MPLS has QoS support. MPLS can be combined with DiffServ at the edge routers, while interior or core routers can process the packet on MPLS labels and DiffServ DSCPs.

MPLS provides an efficient mechanism for supporting VPNs. Resource assured VPNs can be built up using MPLS. MPLS can assure a VPN of the buffer space and bandwidth required at each intermediate router. However, the provisioning of resource assured VPN requires partitioning of label space. The lack of this feature restricts MPLS in guaranteeing resources to each VPN. It also does not support inter-VPN resource management. This is because, VPNs resource allocation can only be expressed as resource allocations of its LSPs, and resources cannot be guaranteed for the further, except by finding out a path and associating each label with a predetermined QoS.

To control the path of LSPs effectively, each LSP can be assigned one or more attributes as summarized below in Table 3.4.

Table 3.4 LSP Attributes

Attribute Name	Meaning of the Attribute
Bandwidth	The minimum required of a path for the LSP to be setup along it.
Path attribute	LSP path should be specified manually or dynamically by CBR.
Set-up priority	Which LSP will get the resource when multiple compete for it.
Holding priority	Should a new LSP preempt the resource that an older one holds?
Affinity	Administrative policy for desired LSP placement.
Adaptability	Switching a LSP to a more optimal path when one is available.
Resilience	When the current path is affected by failure, whether to reroute it.

The Internet Engineering Task Force (IETF) MPLS working group is presently in the process of standardizing MPLS in order to support its implementation over various link layer technologies. The motto of this process of standardization is to define MPLS packet header formats and LDPs, and mechanisms to provide support for QoS. MPLS definitely is more scalable and efficient than the IP-ATM overlay model. Even though MPLS is scalable and efficient in the sense that it provides Traffic Engineering capabilities with the help of its connection oriented nature, and also it supports DiffServ QoS, it does not have built-in security mechanisms. However, higher security level can be provided by integrating VPN with MPLS where a LSP can be viewed as a QoS pipe and it can be switched over IPsec based tunnels. Appendix B describes an implementation of Traffic Engineering in MPLS VPNs.

3.2.3.3 Constraint-Based Routing (CBR)

Constraint-based routing computes VPN routes that are subjected to constraints such as bandwidth, hop count, and delay etc. As Constraint-based routing considers more than network topology in computing routes, it may find a longer but lightly loaded path better than the heavily loaded shortest path as computed by IGP. Network traffic is hence distributed more evenly. The constraints may also be imposed by the network itself or by administrative policies. Suppose bandwidth is used as a constraint, and the shortest path does not meet that constraint, then CBR may select a path that has more hop counts but is lightly loaded. MPLS which is a path-oriented technology has made constraint based routing feasible over public IP networks. Figure 3.7 shows an example of CBR.

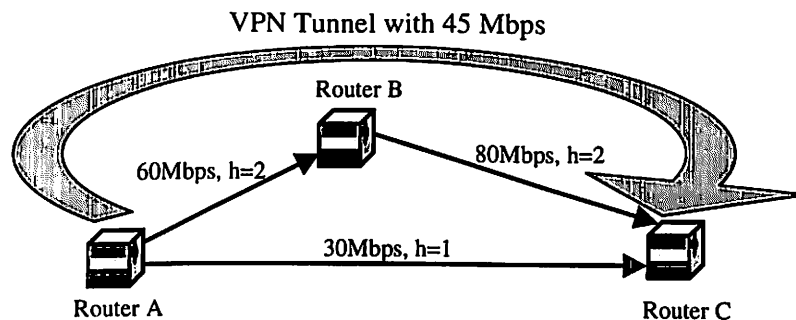


Figure 3.7 Constraint-Based Routing

In figure 3.7, the shortest path between router A and C is through link A-C with IGP metric $h=1$. If the requirement is a tunnel with 45 Mbps reserved bandwidth, one cannot form the tunnel over A-C. The Constraint Based Routing now finds out a tunnel path A-B-C instead, that provides 45 Mbps reserved bandwidth. This is because the shortest path does not meet the bandwidth requirement in this case. Here the bandwidth is assumed to be the available bandwidth over the links. Again, CBR does not compute a path based on instantaneous residual bandwidth of links as it could increase the probability of routing instability. CBR can be online or offline. With online, routers can compute VPN paths at any time, whereas with offline, a router computes VPN paths periodically, and VPN tunnels are then configured over the computed paths.

3.3 Policy-Based QoS Provisioning in IPVPNs

Many vendors are currently building up policy based QoS management architectures as defined by IETF, [Yavatkar00]. The various entities in this architecture are Policy Management Tool, Policy Repository, and Policy Decision and Enforcement Points. Using Management tool, network administrators can specify policies to be enforced in network devices. Also, it converts policy input from network administrator to the format compatible with policy repository. To store policies, a policy repository directory service like Lightweight Directory Access Protocol (LDAP) is used. Also, a database can be used for the same. A policy Decision Point (PDP) retrieves policies from the repository, makes a decision, translates them into device specific configurations, and then sends to network devices known as Policy Enforcement Points (PEPs). A protocol like COPS [Boyle00] can be used for communication between PDP and PEP. Figure 3.8 shows the IETF Policy-Based Management Architecture.

For managing DiffServ VPNs, bandwidth brokers have been proposed as Policy Decision Points (PDPs). A bandwidth broker incorporates policy server functions, but also deals with customer contact and network resource allocation. A bandwidth broker is a software agent that allows a customer of an ISP to buy

bandwidth, enforces the bilateral agreement between them by turning high level QoS policies into low level device configurations and pushing them to network devices to make available the required bandwidth service. The contract between ISP and the customer is called as customer-ISP SLA. An inter-ISP Service Level Agreement (SLA) can be created if this bandwidth service has to be extended to other ISP domains as well.

A Bandwidth Broker essentially is a central server managing and controlling network resources of DiffServ domain. It sells network resources to customers or peer ISPs and makes a difference in providing different categories of services. It should have a pricing and accounting mechanism to make profit out of the sales.

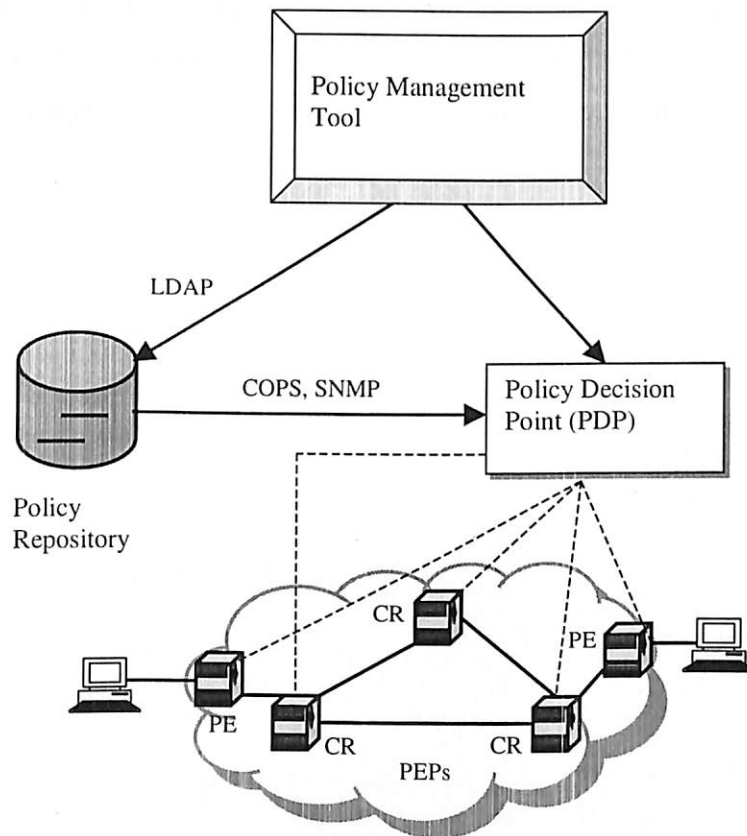


Figure 3.8 IETF Policy-Based Management Architecture

3.4 Summary

This chapter described Internet Architecture, IP Protocol, and ISP architecture that act as backbone for building up IP Virtual Private Networks. It also presented a review of quality of service problem and methodologies for IPVPN. The problem of quality of service discussed in this chapter is from both user's and network provider's perspective.

This chapter also reviewed the QoS support for IPVPNs like DiffServ, MPLS, Constraint-Based Routing, and Policy-Based QoS provisioning including their merits and demerits. Although these architectures are not widely accepted in the IPVPN community, and also none of them will serve as a unique solution to the quality of service problem, they are still significant. Further research is currently going on in these areas including the research presented in this thesis. All these proposals are of particular interest as they attempt to construct resource-assured VPNs, and therefore encounter similar global resource allocation issues as those tackled in this thesis.

Chapter 4

Optimal Tunnel Path Finding in IP Virtual Private Networks

4.1 Requirements of IPVPN Tunnels

In this part the mechanisms for creation of VPN tunnels are described first. Then the two approaches for building VPN tunnels, namely the Customer Premises Equipment (CPE) based approach and Network based approaches are reviewed. The Minimal Cost VPN problem is defined next and we expound on various issues involved in it. In the next section we present a literature survey on this research problem. Section 4.5 describes the algorithms and issues that are developed as part of this research work in finding an optimal tunnel path for IPVPNs.

IPVPNs must be implemented through some form of IP tunneling mechanism, where the packet formats and or the addressing used within the VPN can be unrelated to that used to route the tunneled packets across the IP backbone. Such tunnels, depending upon their form, can provide some level of intrinsic data security. This is referred as encapsulation.

Creation of tunnels is the main mechanism for establishing opaque network layer connectivity between various nodes in an IPVPN. There are numerous IP tunneling mechanisms, including IP to IP, Generic Routing Encapsulation (GRE) tunneling, Layer 2 Tunneling Protocol (L2TP), IPSec, and Multi-protocol Label Switching (MPLS). While some of these protocols are not often thought of as tunneling protocols, they do each allow for opaque transport of frames as packet payload across an IP network, with forwarding disjoint from the address fields of the encapsulated packets.

An IP tunnel connecting two VPN endpoints is a basic building block from which a variety of different VPN services can be constructed. An IP tunnel operates as an overlay across the IP backbone, and the traffic sent through the tunnel is opaque to the underlying IP backbone. In effect the IP backbone is being used as a link layer technology, and the tunnel forms a point-to-point link. There are a number of desirable requirements for a VPN tunneling mechanism, however, these are not all met by the existing tunneling mechanisms. These requirements include:

Multiplexing:

There are cases where multiple VPN tunnels may be needed between the same two IP endpoints. This may be needed, for instance, in cases where the VPNs are network based, and each endpoint supports multiple customers. Traffic for different customers, travels over separate tunnels between the same two physical devices. A multiplexing field is needed to identify which packets belong to which tunnel. Sharing a tunnel in this manner may also reduce the latency and processing burden of tunnel setup. Of the existing IP tunneling mechanisms, L2TP (via the tunnel-id and session-id fields), MPLS (via the label) and IPSec (via the Security Parameter Index (SPI) field) have a multiplexing mechanism.

The IETF and the ATM Forum have standardized on a single format for a globally unique identifier used to identify a VPN (a VPN-ID) [Fox99]. The tunneling mechanism to aggregate packets for different VPNs over a single tunnel can use a VPN encapsulation header. In this case an explicit indication of VPN-ID is included with every packet.

Tunnel Maintenance:

The VPN endpoints must monitor the operation of the VPN tunnels to ensure that connectivity has not been lost, and to take appropriate action (such as route recalculation) if there has been a failure. There are two approaches possible. One is for the tunneling protocol itself to periodically check in-band for loss of connectivity, and to give an explicit indication of failure. For example L2TP has an optional keep-alive mechanism to detect non-operational tunnels.

The other approach [Glesson00] does not require the tunneling protocol itself to perform this function, but relies on the operation of some out-of-band mechanism to determine loss of connectivity. For example if a routing protocol such as Routing Information Protocol (RIP) or Open Shortest Path First (OSPF) is run over a tunnel mesh, a failure to hear from a neighbor within a certain period of time will result in the routing protocol declaring the tunnel to be down.

Another out-of-band approach [Glesson00] is to perform regular ICMP (Internet Control Message Protocol) *pings*. This is generally sufficient assurance that the tunnel is operational, due to the fact that the tunnel also runs across the same IP backbone.

4.2 Approaches for Establishing IPVPN Tunnels

Generally, there are two approaches for establishing tunnels: the “*CPE-based approach*” and the “*Network-based approach*”. In a CPE-based approach, tunnels are established only between the CPE devices (mainly border routers). In a Network-based approach, tunnels are also established between the routers of the core network. Though the CPE-based approaches are simpler, mainly from the perspective of the ISP operating the core network, for scalability and economic reasons, network-based solutions for VPNs are preferred.

In case of a CPE based IPVPN, we can have security applied from end to end. As the traffic is IP traffic, the enterprise has the flexibility of adding and removing tunnels dynamically, instead of requiring the service provider to configure permanent virtual circuits (PVCs) within the network. This allows the enterprise to have the flexibility of a fully meshed network without having to pay for the numerous PVCs needed in frame relay or ATMs. As a result, the enterprise would bring up tunnels for video calls, overnight backups, file transfers, etc. and take them down when not needed.

In Network based approach, we can have the tunnel endpoints within the ISP's core network i.e. the routers in the core network can be strategically activated to make the path optimal between any two-border routers or Customer edge routers, passing through core routers or provider routers. To better understand the trade-off between the two approaches consider figure 4.1, where a network example and the associated graphs are depicted.

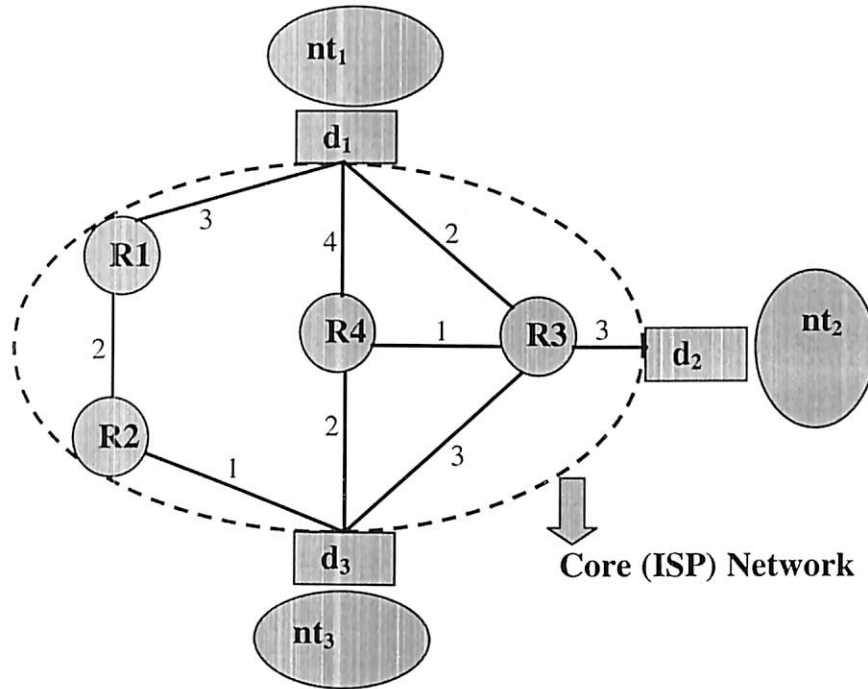


Figure 4.1 Network Example

In this example, there are three corporate networks, referred to as nt_1 , nt_2 , and nt_3 , that need to be connected by a VPN across a core network. The three networks are connected to the core network by means of three-border routers d_1 , d_2 , and d_3 respectively. In the associated graph, each of these networks is represented by its border router. The number associated with every network edge represents the cost of setting up a VPN tunnel across that link.

Three possible VPN configurations for the example in figure 4.1 are shown in figures 4.2, 4.3 and 4.4. Figure 4.2 demonstrates the CPE-based case, where the core

routers do not support VPN. In such a case, the tunnels are established between the border routers of the edge networks: d_1 , d_2 , and d_3 .

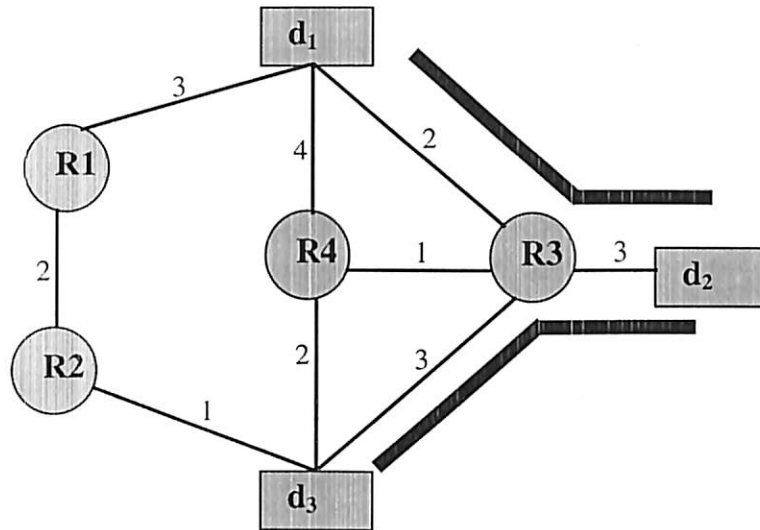


Figure 4.2 CPE based approach

The *cost of the VPN* in this example is $= 2 * 3 + 2 + 3 = 11$. Here, 3 for using the link between d_3 and R_3 and plus 2 for using the link between d_1 and R_3 and plus $2*3$ for using the link between R_3 and d_2 twice.

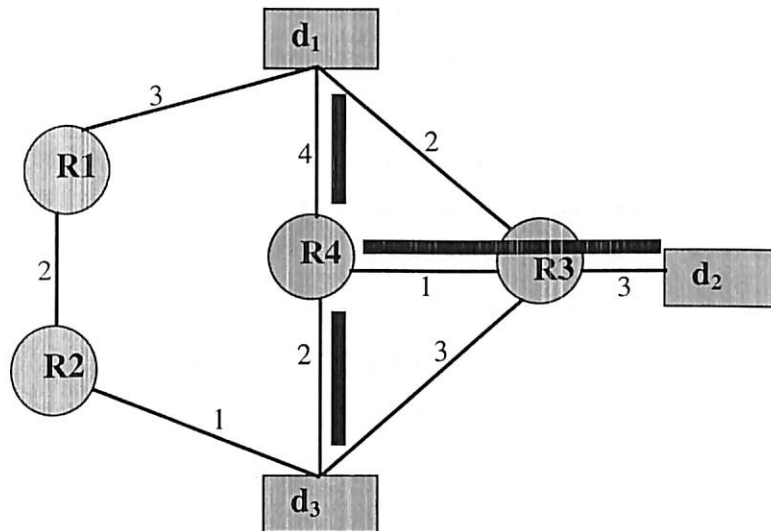


Figure 4.3 Network based Approach

In figure 4.3, the core network's routers exhibit VPN capabilities. Hence, these routers can serve as the endpoints of tunnels, and the *cost of the VPN is reduced to 10* ($2 + 4 + 1 + 3$). The tunnels formed are d_1 to R_4 , R_4 to d_3 and R_4 to d_2 using R_3 .

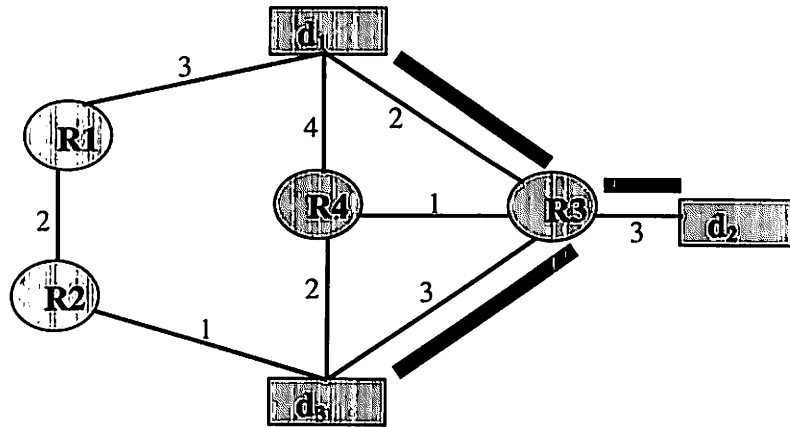


Figure 4.4 Another Network based approach

In figure 4.4, the network's routers also support VPN. However, here rather than R_4 , R_3 is employed as an endpoint of the VPN tunnels, and the *cost of the VPN is reduced to 8* ($2 + 3 + 3$). The tunnels formed are d_1 to R_3 , d_3 to R_3 and R_3 to d_2 .

From these examples, it is clear that when the core routers are capable of functioning as endpoints of VPN tunnels, the routing cost for the VPN can be reduced. However, this does not come *without any cost* because each VPN tunnel terminated at a core router adds to the *management and memory complexity* on that router. To understand the reason for this, consider the VPN in figure 4.3. Suppose that a local host served by d_1 sends a packet to a local host served by d_3 . Upon receiving the packet, border router d_1 sends it over the tunnel to R_4 . Core router R_4 might serve as a VPN router of several VPNs. Therefore, upon receiving the packet it needs to determine the VPN to which the packet belongs. This can be done based upon the identity of the tunnel (VPNID discussed in the previous section) over which the packet has been received, or the address of the source host. Then, R_4 needs to get the routing table associated with this specific VPN and to locate in this table, the entry associated with the destination host. This implies that it needs to maintain a routing table for every VPN for which it serves as a tunnel endpoint. *An immediate*

consequence is that this table needs to be updated, most likely by means of a routing protocol like RIP or OSPF. An independent instance of such a protocol has to be executed for every VPN by the core routers that function as endpoints of the tunnels forming the VPN. Therefore, R_4 also needs to participate in the routing protocol associated with every VPN it serves.

4.3 Minimal Cost (MC) VPN Problem

The layout of VPN tunnels can be optimized to satisfy different optimization parameters, like bandwidth, survivability, minimum hops between source and destination pairs, etc. However, one of the most important factors is usually minimum cost. We assume that when a tunnel is established over a link, it encounters a cost, which is associated with this link. Each link on a real network could have a different cost on every direction. Therefore, the core network is represented by a *directed* graph. The cost of each link on every direction is determined based on administrative considerations of the core network operator (ISP) and on the bandwidth allocated over that link, explicitly or implicitly, for the VPN tunnel. Generally, overloaded links are expected to be more costly than under loaded links. Since network links have different capacity on each direction, and different load on each direction, we believe that a directed graph reflects more accurately the behavior of a real network. We can associate several other parameters as defined in Table 4.1 with each link as various metrics for computing the cost of a link. These parameters can be defined as Quality of Service parameters.

The routers connecting the remote networks to the core network ($d1$, $d2$, and $d3$ in figure 4.1) are referred to as “the border routers” whereas the other routers (R_1 , R_2 , R_3 , R_4 in figure 4.1) are referred to as “the core routers.” The nodes that function as endpoints of tunnels, like R_4 in figure 4.3, are referred to as “active core nodes.” By definition, every border router is an active node. When the routers of the core network do not support VPN, the VPN tunnel path would have to start and end only at the border routers. In such a case, the group of active nodes is equal to the group of border nodes, as in figure 4.2.

Table 4.1 Parameters defining Cost of a Link

QoS Resource	Type	Description
Available Link Bandwidth	Concave (min /max)	This denotes that some percentage of bandwidth will be available (reserved) for QoS flows. This resource is min/max because the minimum (bottleneck) bandwidth on the path is the available end-to-end bandwidth.
Link Propagation Delay	Additive	This denotes the latency encountered on the network links. For delay-sensitive requests, some of the links can be pruned from the graph before selecting the path.
Delay Jitter	Additive	This denotes delay variation on the network path.
Hop Count	Additive	This denotes the number of hops. The minimum hop count path is used by most algorithms to designate the shortest path (least-cost path). Hop count is the only resource that is not typically included in SLAs.
Cost	Additive	This denotes an abstract measure of network resource usage. Cost can be defined in dollars, or as a function of the buffer or bandwidth utilization, for example.

The objective here is to find minimum cost VPN tunnels connecting all the border routers. The VPN scenario that is considered could be either an Intranet VPN or Extranet VPN scenario. An immediate consequence is that the VPN tunnels *form a tree*, because in any circle of VPN tunnels one tunnel can be removed in order to reduce the cost of the layout without affecting connectivity. Since the core network is represented by a directed graph, there are several options to set up a minimum-cost tree spanning a group of nodes. In order to define a unique solution, we assume that one of the branch networks connected by the VPN is *the corporate headquarter* where most of the corporate servers are located, and that the majority of traffic is sent from headquarter to other branches. Hence, we will seek for a directed tree of VPN tunnels rooted at a headquarter border node and spanning the rest of the border nodes. Assuming that using a core router as an active node incurs a price, our goal is to find a minimal cost set of tunnels forming a logical tree rooted at the headquarter border router and spanning the other border routers, such that the cost of the active nodes used is below a given bound.

A formal definition of this problem is as follows:

Given a directed graph $G(V, E)$, an edge weight function $c: E \rightarrow \mathbb{R}^+$, a vertex weight function $w: V \rightarrow \mathbb{R}^+$, a bound on the available “funds” for active core nodes, a group $M \subseteq V$ of border nodes, and a root (headquarter border node) $S \in M$, find a set of directed paths T and a set of active nodes A that minimize

$$\sum_{p \in T} \sum_{e \in p} c(e) \text{ Such that } \sum_{v \in A \setminus M} w(v) \leq F \text{ -----(1)}$$

where

- $\forall p \in T$ the endpoints of p are vertices in A .
- $\forall v \in M \setminus S$ there exists a sequence of one or more directed paths in T that leads from S to v .

The solution to MC-VPN induces a *logical* directed tree rooted at S and spanning $M \setminus S$. This tree is a real spanning tree in an induced graph $G'(A, E')$ where an edge between two vertices in A exists if and only if a path between these vertices exists in the original graph G . Here, we have to achieve a trade-off between the cost of routing, the cost of tunnel set-up and maintenance, and the cost of activating internal nodes as endpoints of VPN tunnels. The design of a VPN is relatively simple if the main target is to minimize the usage of bandwidth. In such a case the VPN should consist of the collection of “shortest paths” from the headquarter node to each of the other sites of the VPN. However, in many networks today, and in most of the future networks, the cost of operation and maintenance of virtual topologies is larger than the cost of the bandwidth. Our model in this chapter considers such a network. The weight associated with every link indicates the cost of building and maintaining a VPN tunnel over this link, regardless of the volume of traffic the link actually carries for each VPN. Such a cost includes the need to set up a tunnel, to maintain the tunnel (e.g., forwarding the “KEEP-ALIVE” messages exchanged between the two end points) and to hold tunnel information (e.g., the VCI/VPI value in the case of an ATM-based tunnel, or the LSP in the case of an MPLS-based tunnel). This cost is incurred for each tunnel, and therefore a link that participates in several tunnels of the same VPN encounters this overhead for each tunnel independently of the other

tunnels. This gives rise to our requirement that $\sum_{p \in T} \sum_{e \in p} c(e)$ has to be minimized.

The cost of tunnel maintenance incurred by the intermediate nodes of a tunnel is different than the cost incurred by the tunnel endpoints. This is due to the following two reasons.

1) These nodes need to do more extensive tunnel maintenance work. For example, they need to have a timer that dictates when a KEEP-ALIVE message has to be sent over the tunnel.

2) These nodes need to maintain not only the tunnel but also the VPN itself. For instance, node R4 in figure 4.3 needs to run a routing protocol and to maintain routing tables in order to determine how to route a packet received from d_1 whose destination is d_2 or d_3 . Node R4 also needs a special logic in order to determine the VPN to which a received packet belongs to (e.g., using a “VPN identifier” as explained before).

Following this assumption, where the cost of tunnel maintenance incurred by the intermediate nodes of a tunnel is different from the cost incurred by the tunnel endpoints, we distinguish in our model between $\sum_{p \in T} \sum_{e \in p} c(e)$ and $\sum_{v \in A \setminus M} w(v)$, and seek for a solution that takes into account both the cost of the links and the cost of activating internal routers as endpoints of a VPN tunnel.

Although MC-VPN assumes that every core router has VPN capability, this assumption may be easily bypassed by assigning infinite costs to nodes that do not have such capability. In order to seek a layout that uses no active core router, the fund can be set to 0. MC-VPN imposes no restriction on the active group, except that this group must include all border routers. Note that the cost of active nodes within the group of border routers is not considered as part of the VPN cost, since the border nodes must be active, and their cost is a constant factor in any feasible solution.

If the fund F is infinite, MC-VPN reduces to the classic Steiner tree problem (STP), since there is no motivation left for reusing edges [Cohen00]. In this case, a Steiner tree may be transformed into a valid solution by decomposing it into paths with endpoints either at the source, destinations, or fork nodes of the tree.

The Steiner tree problem, succinctly, is a minimum interconnection problem. The most basic version is defined on a graph: given a weighted graph in which a subset of vertices is identified as terminals, find a minimum-weight connected sub graph that includes all the terminals. If the edge weights are all positive, then the resulting sub graph is obviously a tree.

On the other hand, when the cost of activating some of the core routers is greater than 0, MC-VPN penalizes a solution that uses the same edge several times by multiplying the edge cost by the number of times it is used. This we have already seen in figure 4.2.

The balance between the node and edge cost function may depend heavily on the details of the core network implementation, such as the processing power of core routers, their memory resources, the cost of setting up and taking down VPN tunnels between core routers, the cost of executing a routing protocol for every VPN, and many other factors.

Minimal Cost VPN is NP – hard and also hard to approximate. In [Ramanathan96b] it is shown that the version of STP on directed graphs is NP-hard as well as hard to approximate. The approximation algorithm achieving an approximation ratio lower than $O(\log n)$ does not exist unless $DTIME(n^{\text{poly } \log(n)}) \supseteq NP$. When the Funds F available are infinite, Minimal Cost VPN reduces to the directed STP. So, the same hardness results apply to MC VPN as well. Hence, a polynomial algorithm that finds an optimal path for MC VPN problem is unlikely to exist. In section 4.5 we concentrate on heuristics that approximate the optimal solution. We first construct a CPE – based solution that ignores the ability of core routers as endpoints of VPN tunnels. We then improve the solution by spending the

funds F on the active core routers in strategic points. The presented algorithms achieve no strict theoretical approximation ratios but in practice they perform respectably well.

4.4 Literature Survey

4.4.1 Cohen's Optimal Cost VPN

[Cohen00] et al. have proposed heuristic algorithms (ASPH, and ADTH) to provide an approximate solution to the optimal MC-VPN problem. They have proposed a layout for Intranet VPN tunnel path considering cost of the links over which VPN is constructed and cost of activating core routers as VPN endpoints. The cost of each link is considered as the administrative cost that is incurred in establishing a tunnel between the endpoints of the link. They have proved that MC-VPN is a NP-hard problem. This is also hard to approximate. They have also proved that there does not exist any approximation algorithm achieving an approximation ratio lower than $O(\log n)$. The general approach used in their work is first to construct a CPE based solution that ignores the ability to use core routers as active nodes. Then, the solution is improved by spending the funds F on activating strategically the core routers. They have compared the results with the well-known STP, which does not find a valid solution but serves as a benchmark.

4.4.2 Mitra's Joint Resource Allocation & Routing in VPNs

[Mitra99] et al. have proposed a joint resource allocation and optimal traffic routing problem in Intranet IPVPNs. Their design considers service level agreements that are the contracts between customers and service providers to carry the customer's traffic. Routing and bandwidth assignment is done jointly. They compute an optimal route over which the VPN traffic is routed through. When the traffic is routed through an optimal path, the bandwidth assignment is made to links in such a way that a weighted aggregate measure of carried bandwidth over the service provider's infrastructure is maximized, subject to the constraint that each VPN carries a

specified minimum. Their design employs multiplexing across services and routes within each VPN, but not across VPNs. This is not across VPNs as it enforces SLA protection strictly. The network is modeled as a multirate loss network. The approach that they have used is first to carry out an optimal routing and then in the next step capacity allocations. The optimum routing problem for each VPN is solved independently.

4.4.3 Anupam's VPN Design for Multicommodity Flow

[Anupam01] et al. have proposed a solution to VPN design problem for multicommodity flow. As there exists bounds on cumulative amount of traffic each node can send and receive; they select a VPN tunnel path for each pair of terminals, and also a bandwidth allocation for each edge of the path, so that any traffic matrix consistent with the upper bounds can be feasibly routed. They have proposed optimal and approximate algorithms for several variants of this VPN tunnel path finding problem based on symmetric traffic matrix and whether the designed network is required to be a tree.

4.4.4 Amit's VPN Provisioning in the Hose Model

[Amit01] et al. have proposed novel algorithms for provisioning VPNs in the recently proposed hose model [Duffield02]. The hose model for IPVPNs allows for greater flexibility as customers need not specify the traffic matrix and it permits traffic to and from a hose endpoint to be arbitrarily distributed to other endpoints. VPN endpoints are connected using a tree structure and their algorithms produce an optimized VPN tree. They have shown that when the links are assumed to have infinite capacity, the general problem of computing optimal VPN tunnel path or tree is NP-hard. For the special case, when ingress and egress bandwidth are same, they have devised an algorithm for computing optimal VPN tree. The time complexity of their algorithm is $O(mn)$, where m is the number of links and n is the number of nodes in the network respectively. Also, they have given an Integer programming formulation for the case where ingress and egress bandwidths are arbitrary. Finally, they have extended their algorithms to handle optimal VPN tree computation when

network links have capacity constraints. They have shown that computing optimal VPN tree is NP hard with capacity constraints even if the ingress and egress bandwidths are equal.

4.4.5 Tuna's Stochastic Approach for Tunnel Path Computation

[Tuna04] et al. have proposed algorithms for computing multiple tunnel paths between source-destination pairs without having to modify the underlying routing protocol. Their algorithms are based on simultaneous perturbation stochastic approximation. Their algorithms do not rely on analytical cost functions, but rather on noisy estimates from measurements. They have proved that their algorithm converges to the solution of optimal tunnel path-finding problem.

4.4.6 Erlebach's Multi-path Routing in Hose VPNs

[Erlebach04] et al. have proposed multi path routing techniques between VPN endpoints in the hose model VPN. They have proved that their multipath VPN routing algorithms are better over earlier VPN tree routing and single path routing where the problem is computationally hard. Their optimal polynomial-time algorithm computes a minimum cost VPN tree between the VPN endpoints of a hose VPN model. The cost over each link is defined in terms of bandwidth. Their algorithm applies to both symmetric and asymmetric bandwidth requirements and also can handle the general constraint that the capacities of the network links are finite.

4.5 Heuristic Algorithms for Minimal Cost Tunnel Path in IP Virtual Private Networks

Here, the VPN tunnel layout is minimized to satisfy bandwidth requirement, packet loss, delay, cost of link, hop count, and resource reservation. We have assumed same heuristic evaluation on both the directions of the link, hence it can be called as a symmetric model. Generally, overloaded links will give rise to bad evaluation function value as compared to under loaded or lightly loaded links. In

figure 4.5, there are 3 border routers (CE 1, CE 2, and CE 3), and 4 core routers (PE1, PE2, PE3, and P4). Depending on the funds available, we can make any number of core routers as active routers. Active means it is participating in VPN maintenance i.e. it should be a VPN capable router. In CPE based approach, by default, all the CE's are active routers. In this scenario, we consider the headquarter router as the root, where all the corporate resources are available. So, an optimal path for VPN tunnels is defined as the problem of finding a minimal cost set of tunnels or paths forming a logical tree routed at the headquarter border router and spanning the other border routers, such that the cost of active nodes is within the reach. We have considered the Intranet/Extranet VPN scenarios.

The problem of Minimal Cost (MC) VPN is mathematically defined as:

Let $G(V, E)$ be a directed graph, with an edge weight function $C: E \rightarrow R^+$, a vertex weight function $w: V \rightarrow R^+$, a bound on available funds F for active core nodes, a group $M \subseteq V$ of border nodes, and a root (headquarter node) $s \in M$. Our problem is to find a set of directed paths T and a set of active nodes A that

minimize

$$\sum_{p \in T} \sum_{e \in p} C(e) \quad \text{such that} \quad \sum_{v \in A \setminus M} w(v) \leq F$$

where, $\forall p \in T$ the endpoints of p are vertices in A , and $\forall v \in M$ there exists a sequence of one or more directed paths in T that leads from s to v .

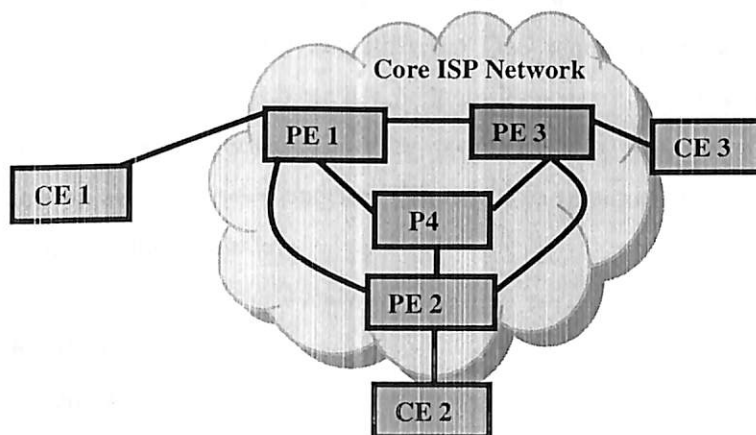


Figure 4.5 An Intranet/Extranet based VPN Scenario

Our work is motivated by the work done in [Cohen00][Fox99]. In these two, the authors have described several possible VPN topologies along with the trade off between cost of routing, cost of VPN tunnel set up and maintenance, and the cost of activating the core routers. We are not limiting to only bandwidth usage, rather we have considered the cost of maintaining the virtual topologies. Our approach is solving the MC-VPN problem from a QoS perspective, where the path that we shall be getting after applying our heuristic evaluation function to all the nodes will be a maximal QoS guaranteed path, satisfying the SLA needs. This problem is NP hard, so a polynomial algorithm for finding the optimal solution does not exist. So, we have proposed a search algorithm with an appropriate heuristic to solve it.

4.5.1 Path Finding for MC VPN

To find the best path, a simple network is formed with all the intermediate VPN routers and the hosts as shown in figure 4.5. An algorithm for deciding the heuristic values is discussed below. The best node next to the source VPN router that is selected is the router with the highest heuristic value. Then that node becomes the source, and among all its children, the best one is chosen. The motivation behind using a binary-formed heuristic for each node in the graph is to meet the needs of selecting a path that can provide better quality of service guarantees. [Cohen00] used a shortest path heuristic where the next node in the path selected was the one with least administrative cost that might not give always a tunnel path with satisfactory performance. We cannot use again any other heuristic function that can give an integer number, where for two different QoS requirements we may get the same heuristic value for two or more descendents. Hence, we have here considered a binary formed heuristic to solve MC-VPN problem. This heuristic will give definitely distinct values for the descendents. The heuristic values are based on the performance of the node which include:

- a. Maximum allowed bandwidth
- b. Packet loss
- c. Jitter
- d. Delay

- e. Hop count
- f. Resource reservation
- g. Cost of the link

4.5.2 Bit wise Positive Consideration Heuristic (BPCH)

In this algorithm we consider only the positive effects of all the QoS features listed above. To explain this, let us consider a node that supports a bandwidth of 1Mbps but it is 20 hops away from the destination. In this case, the hop count of the node does not have any negative effect in the value of the heuristic. The path we computed using our BPCH heuristic targets core ISP network only. This is because the company's headquarter and branch offices in an IPVPN are tunneled over this core ISP network.

The heuristic value of the node is in binary format in which each bit corresponds to one of the QoS features. For example, let us assume that the minimum requirements of the connection are:

- | | |
|----------------------------|-----------------------|
| 1. Should support capacity | $\geq 128\text{Kbps}$ |
| 2. Cost of the link | ≤ 10 |
| 3. Delay | $\leq 500\text{ms}$ |
| 4. Jitter | $\leq 4\%$ |
| 5. Packet loss | $\leq 4\%$ |
| 6. Hop count | ≤ 14 |
| 7. Resource reservation | = yes |

Now these values can be split into various classes e.g. bandwidth can be split into 128 Kbps – 1 Mbps and 1Mbps - *. This helps in giving more preference to 1Mbps connection when requested for video conferencing or other tasks that require a capacity of more than 1 Mbps. The values of the bits can be as given in Table 4.2.

The heuristic values for all the nodes are computed from the above table. These values are converted to decimal for comparison. If there is a variation in the performance of the node at different times then the values of the heuristics is computed at that time and the best path be found once again.

Table 4.2 QoS features with binary weight

Bit Position	QoS Feature	Corresponding Decimal Value
0	Delay of 50 – 500 ms	500
1	Loss of packet 2 – 4 %	4
2	Resource reservation = No	0
3	Resource reservation = Yes	1
4	Jitter 2 – 4 %	4
5	Hop count 5 – 14	14
6	Cost 4 – 10	10
7	Supports 128 Kbps - 1 Mbps	128
8	Jitter 0 – 1 %	2
9	Loss of packet 0 – 1 %	2
10	Delay of 0 – 49 ms	50
11	Cost 0 – 3	3
12	Hop count 0 – 4	4
13	Supports 1 Mbps +	1024 (1M)

4.5.3 Implementation of BPCH

The implementation of BPCH shall be explained with the example considered in Table 4.2. To compute the heuristic value we need one more table, which is given below. Following table is needed to find out which bit number represents the bandwidth, delay, loss etc.

In Table 4.3 each performance number corresponds to a particular QoS feature, for example number 0 indicates bandwidth (bit numbers 7, 13 in Table 4.2) with a corresponding value of 8320.

Table 4.3 Priorities for Different QoS parameters

Performance No.	Corresponding Value (In Binary and Decimal)
0	(10000010000000) = 8320
1	(00001000000010) = 514
2	(00000100010000) = 272
3	(00010000000001) = 1025
4	(01000000100000) = 4128
5	(00000000001100) = 12
6	(00100001000000) = 2112

Third column of Table 4.2 can be used with reference to Table 4.3 to find various classes for the QoS features. For example, to find the classes for bandwidth, we first refer to Table 4.3, which gives the bit numbers 7 and 13. These bit numbers when referred to column 3 of Table 4.2 give the values 128 and 1024. This indicates that bit 7 corresponds to a bandwidth of 128kbps – 1024 kbps where as bit 13 corresponds to 1024 and greater. The word greater cannot be used always because when we consider delay, bit 10 in Table 4.2 (third column) indicates 50ms and smaller. Deciding greater or smaller is based on whether the sequence of the values corresponding to the bit numbers for a particular feature is in ascending order (greater) or descending order (smaller). Hence for bandwidth, bit 13 corresponds to 1024 kbps and greater.

Now the value of the heuristic is computed by setting the bits to 1 in a 14-bit (in our example) value if the performance of the node falls within the classes specified in Table 4.2.

The heuristics for finding the QoS tunnel path is given in algorithm 1. The space complexity of this algorithm is $O(1)$. We cannot guarantee that this solution is optimal. However, it will give us a minimal VPN tree with headquarter as the root node and leaves as the destination nodes. We need additional mechanisms for escaping from local minima. The input to this algorithm is source destination pairs, the core ISP network with the link QoS parameters as defined in the above section.

An application of the BPCH algorithm on figure 4.5 will yield the following minimal VPN tree which is a solution to MC-VPN problem with active routers as border routers only, hence it represents a CPE based solution to the said problem. This is achieved by keeping the funds for core routers infinite.

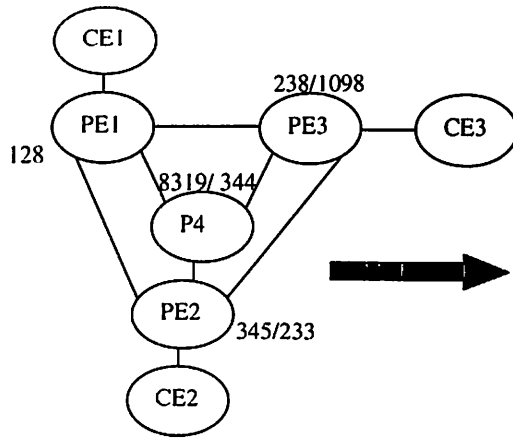
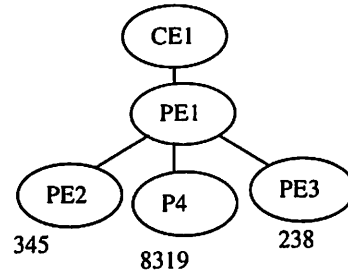
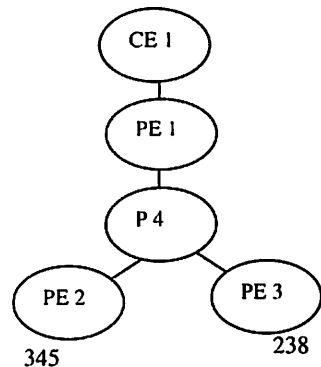
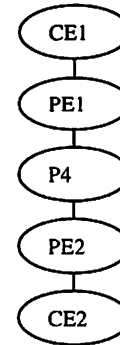


Figure 4.6 Example Network Graph

(a) Step 1: Only one neighbor



(b) Step 2: P4 is selected



(c) Step 3: PE 2 is selected

(d) Step 4: Path from CE1 to CE2

Algorithm 1: Finding best QoS Tunnel Path using BPCH

Begin {QoS path computation}

- i. Path $p \leftarrow \{\phi\}$, and VPN tree $T \leftarrow \{\phi\}$
- ii. Start node $s \leftarrow$ the source router
- iii. **For** each pair of source and border router

Do

Begin

1. Root = the source router s .
2. Path $p \leftarrow \{\text{Root}\}$
3. **If** p includes a border router then return VPN tree and path p otherwise continue with Root.
4. **Loop** until p includes a border router or Root is unchanged
 - a. Let **SUCC** be the node such that any possible next node will be better than **SUCC**
 - b. **For** each neighbor router that lies in the path to border router
 - i. Evaluate the node using BPCH
 - ii. If it is the border router, then return else compare it with **SUCC**
 - iii. If better then set **SUCC** = this node
 - c. **If** **SUCC** is better than Root then Root = **SUCC**
 - d. Add the selected one as a node to T , i.e., $T \leftarrow T \cup \{\text{node}\}$
 - e. Update path, i.e., $p \leftarrow p \cup \{\text{node}\}$

End;

- iv. Join all the trees with only one source (the headquarter router) as the root and leaves as the border routers giving a **minimal cost VPN tree**.

End {QoS path};

In the similar manner we get the VPN tree for source CE1 and destination CE3. Then we join both the trees as per the last step of the algorithm. The resultant minimal VPN tree is as shown in figure 4.7.

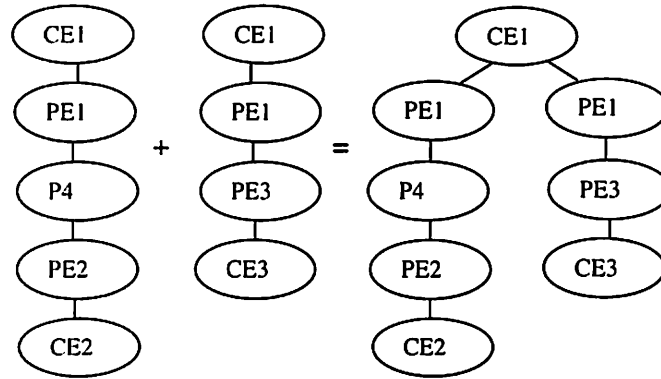


Figure 4.7 Joining of both the VPN trees

Heuristic computation is dynamic, and the best descendent is chosen that satisfy the SLA requirements. Figure 4.6 shows two values of heuristic for PE2, PE3, and P4 nodes. These values correspond to different destinations like one for CE 2 and one for CE 3.

4.5.4 Activating Core Routers

If we examine figure 4.7 we can find the edge PE1 being used twice i.e. one for connecting to CE2 and the other for connecting to CE3 at a later point of time. This incurs additional overhead or QoS cost. But, this is inevitable, if we cannot activate any core router. Otherwise it is better to make PE 1 a VPN capable node so that it can work as the tunnel endpoint between CE 1 and itself. It should now maintain the tunnel between itself and CE 1, hence finding out whether the packet is destined to the tunnel PE 1 and CE 2 or PE 1 and CE 3. With this extra overhead, we can achieve better QoS path between source and destinations. This is called as a Network based approach where we activate a core router. The extra work PE1 has to do is to maintain a routing table based on VPN identifiers as described in section 4.2. But, at the expense of this extra overhead or routing complexity, we can minimize the VPN tunnel path hence providing a required level of quality of service to various customers of the VPN. Also the functions available at PE1 can be used by the sources. Surely, activating PE1 will ensure a QoS path in formation of VPN tunnel.

Another advantage we can achieve here is that any other source can also use the rest part of tunnel path (PE1 to CE2 or PE1 to CE3) without recomputing again a shortest path. Of course, it has to find out a QoS path from itself to PE1 first.

Algorithm 2: An optimization algorithm for activating selective core routers

- i. Let $d(v)$ = Profit of a vertex v in T
 - ii. Let $w(v)$ = Heuristic value for the node v
 - iii. Let funds (F) = Initial value
 - iv. Let $C = \{\text{set of candidate nodes for which } d(v) \geq 2 \text{ and } w(v) \leq F\}$
 - v. Initialize the Active set (A) to NULL
 - vi. **Until** the funds are exhausted or C is empty
 - Do**
 - Begin**
 - a. **For** every node v in C
 - Calculate the benefit $b(v) = w(v) / d(v)$
 - b. Arrange all the vertices according to the ascending order on $b(v)$.
 - c. Select the vertex with lowest $b(v)$ first.
 - d. **If** the remaining incoming path still connects the headquarter node s to the node selected in step c then
 - i. Funds (F) = Funds (F) – $w(v)$
 - ii. Activate this node, i.e. make this as part of a tunnel ($A \leftarrow A \cup v$)
 - iii. Remove this core router from the candidate list
 - iv. Update the candidate list again, i.e. Remove nodes with $w(v) > F$ and Add any node whose $w(v) < \text{new } F$
 - Else**
 - Consider the next core router with second lowest $b(v)$ for activation.
 - e. **If** any active node is left out with one outgoing edge, deactivate it, as it becomes an intermediate node in some path.
- End;**

Input to our optimization algorithm 2 is the VPN tree produced from algorithm 1, and the metric funds available. Funds available is a variable that is used as a metric for deciding on number of core routers that can be activated. Any node whose heuristic value is more than funds cannot be chosen as the activating router. Also, any node whose in degree is less than 2 is to be left from the candidate set of activating core routers. This is because that is the only router in the path with one incoming link. The logic of the algorithm is arranging every possible node (the node that can be selected) in the core network as per their benefit value. Select the node with lowest benefit first and then decrease the funds by the heuristic evaluation function value of the selected node. Repeat this procedure with a new list of nodes without this one. Repeat the selection till the funds are exhausted.

Applying the optimization algorithm on figure 4.7, we have only one candidate node i.e. PE1 whose benefit value is it's heuristic value divided by two (in degree). Hence, we activate this router i.e. we make this router VPN enable that can participate in creation, and maintenance of VPN tunnels. In a more complex network, we can get the real advantage of this optimization algorithm in computing the optimal QoS tunnel path. The minimal VPN tunnel path is as shown in figure 4.8. The tunnels that are to be formed are CE1 to PE1, PE1 to CE2, and PE1 to CE3.

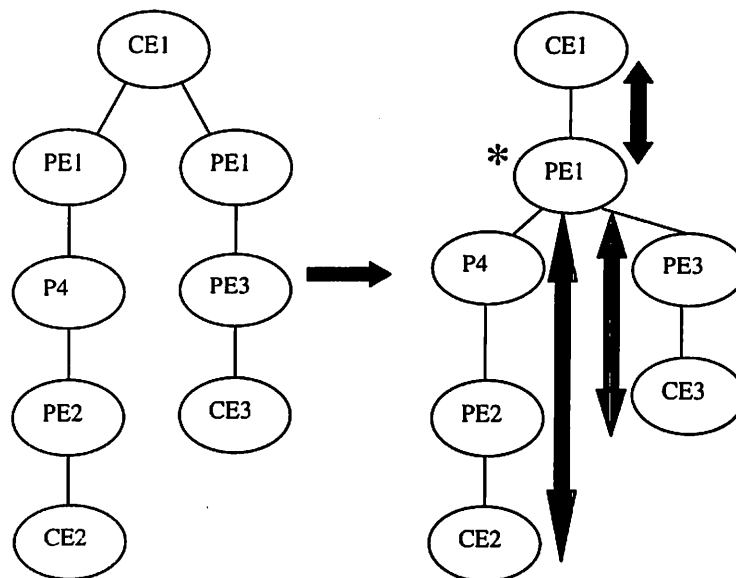


Figure 4.8 The core router PE1 is activated giving rise to three tunnels

4.6 Simulation Results and Analysis of Tunnel Path Finding Algorithms

The route finding algorithm with the heuristic evaluation function described in this chapter (section 4.5) is implemented and tested with random graphs. Though the real world networks are not completely random in their structure, we believe the simulation results predict accurately the behavior of proposed algorithms in a realistic set up. The algorithm was run with different networks. Each link was assigned randomly all the QoS parameters listed, and the BPCH was used to compute the best QoS path. For every run, different values of funds were taken. It was concluded that as we go on increasing funds, the VPN cost reduces i.e. the best QoS path is computed with more number of activating or participating core routers. Also the number of branch office routers i.e. border routers is varied. VPN cost also was increased when we increased number of branch office routers. The performance graph of all these simulation results matched the performance of the Active Double Tree Heuristic (ADTH), Active Shortest Path (ASPH) algorithms presented in [Cohen00]. The results obtained through this algorithm are also compared with Shortest Path Heuristic (SPH) algorithm [Takahashi80] of Steiner Tree Problem (STP), which do not provide any valid solution for optimal QoS path for a VPN. It performed better than Shortest Path Heuristic (SPH), when funds available were more. As ADTH or ASPH [Cohen00] are not considering any QoS requirement of the traffic class, our algorithm has an advantage over these two in this aspect. Also these two are considering the shortest path heuristic, which may not always give optimal results. But, our BPCH heuristic will give optimal results. This advantage will overpower the demerit i.e. the time taken to compute a heuristic value at a node. The results are plotted in figure 4.9 and figure 4.10.

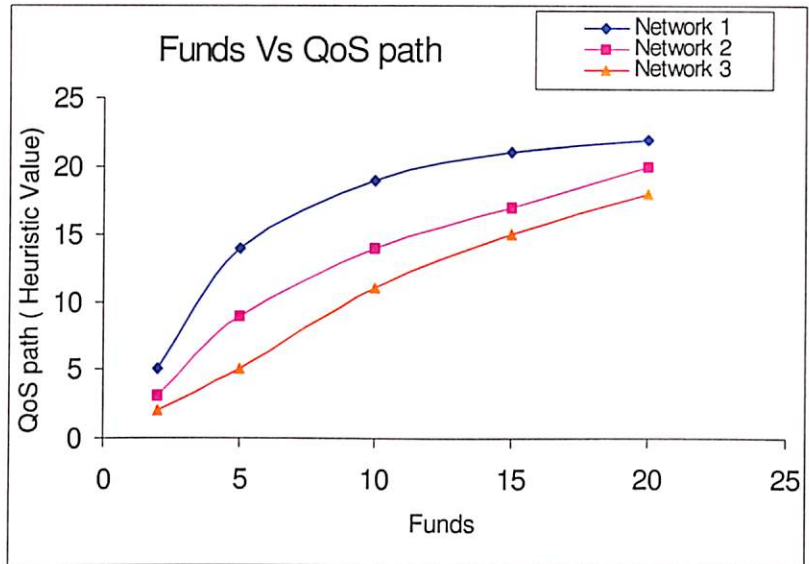


Figure 4.9 Increasing Funds also increases the Heuristic function estimation of total path

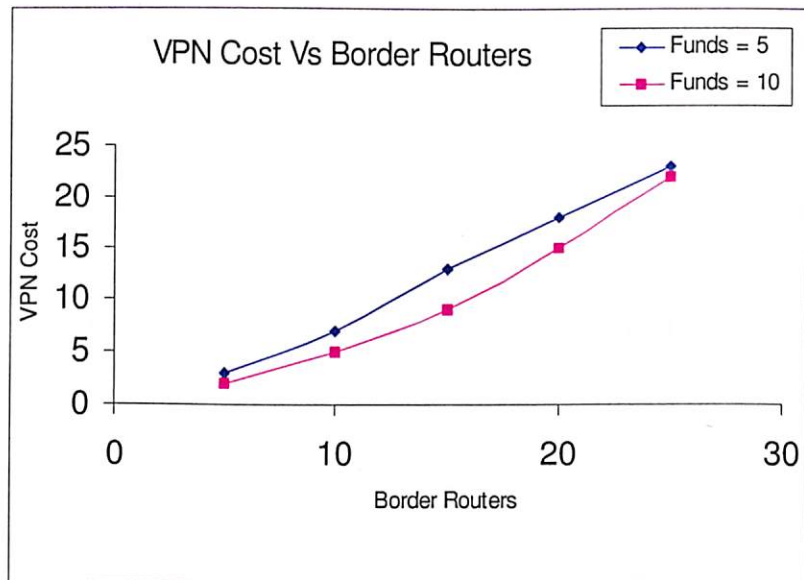


Figure 4.10 Increasing Border routers also incurs overhead on VPN cost for a fixed fund

4.7 Summary

In section 4.1 we presented the need for optimal tunnel path finding in an IPVPN. This is basically to allow the service provider to provide QoS guarantees to the customers. There are two types of IPVPNs we discuss in this section i.e. CPE based and Network based. Finding an optimal tunnel path in a CPE based approach can be carried out either by employing a shortest path algorithm or by carrying out constraint based routing. But, for Network based IPVPNs, an optimization phase is additional that strategically selects the core routers that are to be activated giving an optimal VPN tunnel path. This problem we called as Minimal Cost VPN problem which is NP-hard and hard to approximate. We also defined several factors that can be considered for modeling the cost of the link while solving the minimal cost VPN problem. We also included the work done till date in this area.

Section 4.5 presented a solution to minimal cost tunnel path finding problem in IPVPNs. We defined several QoS parameters for the sake of computing the link cost. Two heuristic algorithms were presented i.e. bit wise positive consideration heuristic (BPCH) and another optimization heuristic to strategically activate few core ISP routers keeping the utilization of funds under control.

Section 4.6 presents the results of our BPCH heuristics algorithms in terms of different network behavior for QoS path computation with varying funds, and also cost of VPNs for varied group of destination routers.

Chapter 5

Dynamic Bandwidth Management in IP Virtual Private Networks

5.1 Bandwidth Management and Logical Networks

An IPVPN is built up at network layer that is independent of the underlying physical network. Bandwidth can be allocated to each tunnel, although they can also be established without any resource reservation. The bandwidth assigned to each tunnel can be changed as required [Sato90] [Dziong96][Leon-Garcia03]. The major drawback here is, IPVPNs with a greater number of source destination pairs and/or fewer physical links with small capacity, there could be a significant number of tunnels configured over certain links that divide their capacity into smaller chunks.

Given a physical topology, the first task is to build the VPN design. This is as described in chapter 4. The objective of this is to optimize the topology and bandwidth allocation as per the predicted traffic demand and performance constraints. Once the tunnel paths are decided, the initial design may not be optimal, due to either discrepancies between the real and predicted traffic patterns or changes in them. Hence, an adaptive procedure is needed to adjust the bandwidth allocations of each tunnel, and if required the topology as well. Allocating bandwidth to a VPN tunnel is based on performance measures from the connection layer as shown in figure 5.1.

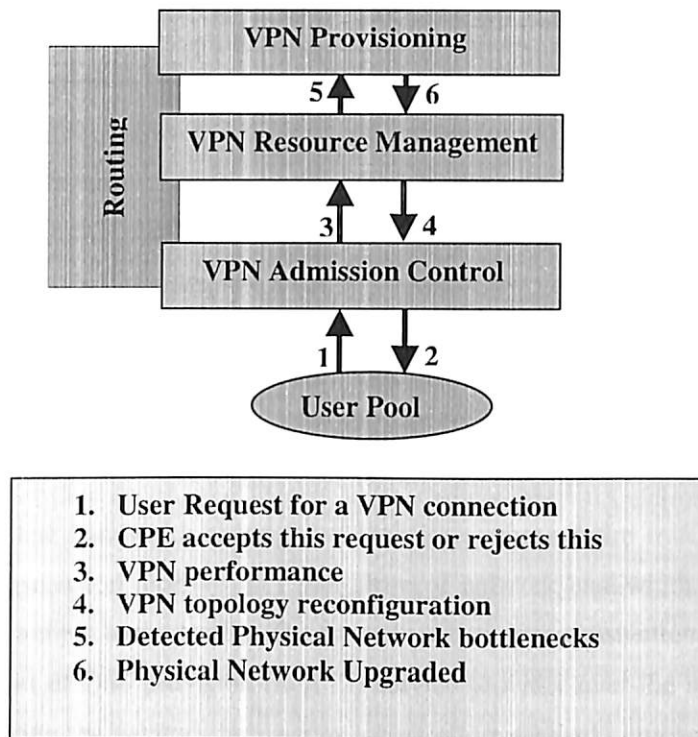


Figure 5.1 VPN Bandwidth Management with other Network Management Functions

The VPN Admission control layer establishes new user connections into the tunnel. The VPN resource management layer logically allocates bandwidth to these connections. This allocation is made in such a way that it separates traffic control algorithms and higher layers from the cell layer. However, the bandwidth allocation to user connections or flows makes the system similar to dedicated circuit switched connections. The VPN provisioning takes care of updating the VPN topology based on the performance measures of the VPN. Some of the functions of this layer could be detecting physical bottlenecks, such as link failures or node failures etc. These functions analyze the VPN and propose physical network updates and improvements to the service provider. All these layers closely interact with the routing mechanisms. User connections need to be routed over the VPN and the tunnels need to be routed over the physical network. The improvements in the physical network may affect significantly the routing of tunnels. So, routing has to be taken into consideration at every layer.

5.2 Bandwidth Allocation Mechanisms

Our objective in this part of work is to guarantee the user with the amount of bandwidth he or she needs based on online measurements. Also we should see that the network performance is within an acceptable range. Different methodologies that can be adopted in this scenario are discussed in this section. Available in the literature are several methods like complete sharing, complete partitioning, virtual partitioning and trunk reservation. The overview of these mechanisms is given in this section. Researchers have used these techniques to solve similar type of problem we are trying to solve in this thesis.

Complete bandwidth sharing is a technique where entire traffic is accepted and each transmission receives an equal share of network bandwidth. It provides a best effort treatment to the traffic. The scenario where user satisfaction is critical, we need some sort of QoS provision from the service provider over the network. As in complete sharing the network behaves as a best effort network, guaranteeing QoS is the responsibility of the service provider. This paradigm provides room for high network utilization, but does not guarantee the required performance levels. There are several schemes proposed in literature that try to provision QoS in best effort networks.

[Hurley99] et al. proposed a scheme where each packet is marked as blue or green. Green packets receive more losses during congestion than blue ones, hence they receive less delay jitter. Each packet is assigned a finishing service time deadline, and the packet currently having the lowest value is served first. Green and Blue packets are assigned with service time deadline differently such that green packets receive low delay jitter, but at the expense of lower throughput.

Complete bandwidth partitioning divides bandwidth between the services, allowing each service exclusive use of its' allocated capacity. Complete partitioning can improve the efficiency of the IPVPN, mainly in terms of user satisfaction and greater revenue for service providers. Our work in this thesis is based on this mechanism. Different types of bandwidth partitioning schemes are described next.

5.2.1 Static Link Partitioning Scheme

This is one of the primitive bandwidth management schemes. In this scheme, a physical link is partitioned into various virtual links or traffic classes. The total bandwidth of the physical link is *statically* partitioned among these virtual links. By saying statically we mean that the bandwidth for any virtual link does not change with time, no matter whatever may be the traffic in that virtual link. This concept is well illustrated in mathematical terms below.

Let total bandwidth of the physical link be C , and the number of virtual links be I (indexed from $i = 1, 2, \dots, I$). α_i is the weight associated with the i^{th} link L_i such that the total bandwidth allocated to L_i is given as $C_i = \alpha_i C$.

Now we have the inequality

$$\sum_{i=1}^I \alpha_i C \leq C$$

$$\text{i.e., } \sum_{i=1}^I \alpha_i \leq 1 \quad \text{where, all } \alpha_i \text{ s are constant.}$$

The strengths of static partitioning scheme are as blow:

It is the simplest partitioning scheme. It is easy to comprehend, implement and maintain. Also there is complete isolation among the virtual links, which ensures privacy. Lastly, there is complete eradication of exploitation of one traffic class over the others.

The weaknesses of this scheme are outlined below:

As quite evident, there are high probabilities for poor performance of the network. This is because, if a traffic class, say L_i , is almost empty, then the bandwidth allocated to it is unused. It may so happen that another traffic class, say L_j , may have a heavy traffic but cannot utilize the unused bandwidth of L_i . It does not support statistical multiplexing.

Hence, to avoid these limitations and increase the utilization of the network, one has to be in a position to redistribute the unused bandwidth of other virtual links among those that are needy. This gives rise to a new scheme of bandwidth partitioning namely the *Dynamic Bandwidth Partitioning* discussed next.

5.2.2 Dynamic Bandwidth Partitioning Scheme

In this scheme, we allow α_i s to vary with time depending on the traffic intensity of the virtual links. The unused bandwidth of a virtual link may be distributed among various other virtual links where the traffic intensity is high. This is illustrated with a mathematical model as given below:

Let the traffic intensity of i^{th} link $L_i = \phi_i$. So the α_i s are a function of ϕ_i i.e. $\alpha_i = \alpha_i(\phi_i)$. If we assume that the traffic intensity of a virtual link is a function of time, i.e. $\phi_i = \phi_i(t)$ then we have α_i s as a function of time i.e. $\alpha_i = \alpha_i(t)$. But always we have $\sum_{i=1 \dots n} \alpha_i(t) \leq 1$ valid.

The merits of this scheme are that the utilization of the network increases and the user satisfaction level is also high. The demerits are, this scheme may result in starvation of the underutilized links. Extra care has to be taken to avoid this. Secondly, we are not sure when a burst occurs on a virtual link, that is presently underutilized, say L. When a burst occurs on L, the redistributed bandwidth is restored to L, thus decreasing the throughput of a heavily used link to which the unused bandwidth of L was given.

5.2.3 Other Partitioning Schemes

There is yet another way of partitioning the bandwidth called the *Hierarchical Packet fair Queuing Algorithm* to efficiently implement link sharing. It is similar to the above-mentioned dynamic partitioning but the traffic classes are given priorities based on their importance and the type of applications that are present in it (i.e. based on their classification). This algorithm ensures that each traffic class gets its requisite share of the link capacity, and carry out fair redistribution of the residual unused bandwidth of a traffic class to other classes. The hierarchical packet fair queuing algorithm work at the scale of packet transmission time.

The demerits of this scheme are, although they redistribute the free capacity of inactive classes to active classes, they cannot ensure, on their own that the redistributed capacity will be available for a predictable period of time. This residual capacity goes back to its respective class as soon as the sending rate of the class is increased. As a result, real-time sessions that require a guaranteed Quality of Service (QoS) for their lifetime, cannot utilize this excess capacity. Also this scheme poses a problem for the traffic classes that need a minimum bandwidth for a guaranteed QoS, as they cannot get their minimum bandwidth guarantee from redistributed capacity.

Dynamic bandwidth partitioning can also be accomplished using *trunk reservation*. [Key90] et al. proposed trunk reservation based on limited capacity availability to low-priority traffic in the network whenever there is congestion. [Mitra96] et al. proposed trunk reservation based on physical partitioning. In this scheme, bandwidth is divided among competing services. If we have K classes of traffic, calls of class k , $k=1,2,\dots,K$ are assigned bandwidth C_k , where $\sum_{k=1}^K C_k = C$. Here, C is the link capacity. Hence, instead of just one trunk group, one can think of K trunk groups. The k^{th} trunk has capacity C_k . Each of these k trunk groups is assigned a trunk reservation parameter r_k . Calls of each class have different priorities at their allocated trunk group. If a trunk group has spare capacity on it, then an arriving call is accepted, otherwise, the call chooses, in accordance with some discipline, another trunk group, which has spare capacity more than the trunk reservation parameter r_k and is carried there. If no such group exists, the call is lost. The simplest discipline picks the overflow trunk group at random. However, this scheme which is based on physical partitioning leads to unfair usage distribution when traffic classes arrive in bursts. At the same time it gives us fairness in comparison to complete partitioning scheme. [Mitra96] et al. improved this approach by a newer approach where instead of having fixed priority for each class, they defined priority of each class based on the system state. This scheme they called as Virtual Partitioning. The k^{th} class of call is assigned a nominal capacity C_k , where $\sum_{k=1}^K C_k \geq C$ and C_k is sufficiently high to ensure the desired grade of service. Let

n_k denote the number of calls of class k in progress. An arriving call of class k is always accepted if

$$\sum_{r=1}^k n_r < C - r_k$$

where r_k is a trunk reservation parameter chosen to protect the remaining classes against overloads of class k . So, if $n_k < C_k$, priority status of class k calls are high and if $n_k \geq C_k$ then the priority status of class k calls are low.

Hierarchical Virtual Partitioning (HVP) is another approach [Mitra97] for providing sharing of a resource, such as a link, by multiple customers, each with multiple service classes over a shared infrastructure. This scheme can handle several customers, each having several service offerings. These services have different bandwidth requirements and statistical characteristics like call arrival rates and mean call holding times, call blocking probability in the form of quality of service requirement. The objective of HVP is to provide customers with the QoS that approximates as closely as desired that experienced on dedicated, private networks, while in fact, the aggregate traffic is multiplexed on a shared network. Of course, the objective of sharing is to realize multiplexing gains. On one hand, this scheme guarantees fairness and robustness. On the other hand it also guarantees efficiency, better resource utilization, and multiplexing gain that come from sharing resources. HVP is more closely attuned to the notion of Virtual Private Networks in which the notion of multiple customers, each with multiple service classes, is embedded. This scheme relies on nominal allocations of capacity to each customer and service, and priorities in the call admission process, which are implemented by dynamic trunk reservations. A reward/penalty paradigm is proposed to reflect the issues in Virtual Private Networking. It uses a hierarchy of state dependent priorities.

The methods that bring bandwidth allocation closer to complete partitioning are not a recommended approach. This is because of the static nature of link partitioning. As the ratio of QoS sessions over best effort sessions change over time, it is extremely difficult to find out what fraction of link capacity should be used for each traffic class. A dynamic link partitioning approach would be better in this scenario. Under this scheme, the measured link utilization for different traffic classes

can be used to periodically update what fraction of the link capacity is to be assigned to each traffic class. Often this may not help as link utilization may suddenly change or if we do not have the required bandwidth available. These issues will be further analyzed in section 5.6 and new algorithms to dynamically manage the bandwidth in IPVPNs will be proposed.

5.3 Utility-based Bandwidth Allocation in IPVPNs

The IPVPN traffic has become very heterogeneous since recent times. The Internet was originally designed for transporting only data traffic, but in recent years it has become obvious that the TCP/IP-based Internet has to change and become a global broadband communications network, capable of supporting diverse network applications. There are many differences between these applications. On one side we have video conferencing that demands a high network performance in terms of guaranteed bandwidth and guaranteed delay bound. We also have simple file transfer that does not impose any delay and bandwidth requirements. Hence, based on the importance of the data or application over the IPVPN, the entire network traffic can be classified into various classes. We can develop utility functions for these traffic classes based on the end user satisfaction.

We can broadly classify the VPN traffic into three major categories namely Brittle traffic, Stream traffic, and Elastic traffic. An example of brittle traffic could be video conferencing, stream traffic could be playing audio or video files over IPVPN, and elastic traffic could be simple file transfer. The brittle traffic requires strict end-to-end performance guarantees and does not show any adaptive properties. Stream traffic requires the network to provide a *minimum* level of bandwidth. The network stream traffic applications can adapt to the allocated bandwidth level. Elastic traffic flows have loose response time requirements. The utility functions that are used to model these traffic classes are presented in figure 5.2.

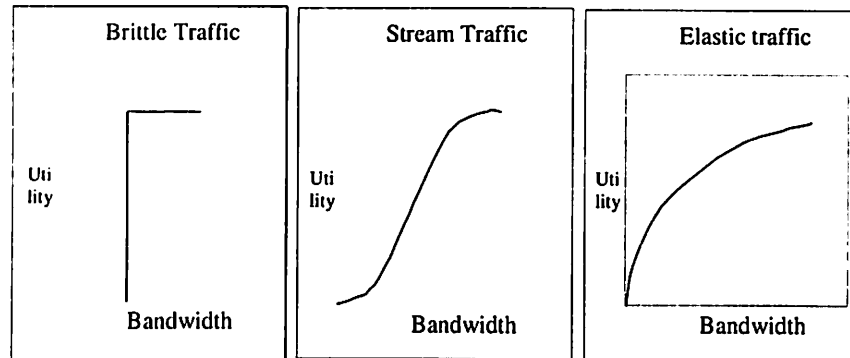


Figure 5.2 Types of IPVPN traffic and their utility functions

The chosen utility functions reflect the major properties of the traffic classes. The first utility function represents brittle traffic. Users of a brittle application, such as a highly sophisticated video telephony or a live TV broadcast do not tolerate any performance variations of the applications; they are either satisfied with the application's performance, or dissatisfied. On the other hand, the users of the stream application (streaming video/audio) can tolerate changes in the bandwidth level allocated to their application, as long as that level is higher than the minimum requested. Finally, users of elastic applications (file transfer) do not have any requirements on the network, not even minimal, and therefore utility function for this traffic class is strictly concave, as presented in figure 5.2.

The utility-based bandwidth allocation in VPNs is an interesting problem that has been addressed in this work. VPN user derives certain utility from the application he is using over his or her VPN. We treat here utility as the level of end user's satisfaction. This utility is subjective measure of application performance over the VPN. In other words utility depends on the quality of service guaranteed by the VPN service provider. Earlier approaches use different metrics like end-to-end delay, delay jitter, throughput, etc. to guarantee quality of service to IPVPN customers. So, use of this utility function gives a unique metric for QoS guarantee to VPN customers. Allocating bandwidth optimally on the basis of user utility means the mean utility of end user's must be maximized. [Kel97] argues that bandwidth should not be shared so

as to maximize the link utilization, but to maximize an objective function representing the overall utility of the end user.

The measurement of utility is a complex task. Utility levels are calculated using utility functions. These utility functions map network parameters like delay, throughput, packet drop, etc. into the performance of an application. Hence, there should be some scheme for allocating the available bandwidth in such a way that user utility should be improved. The analytical model that best suits to this task is described next.

Let us consider a VPN with link capacity as C . This VPN serves different class of traffic as described earlier. Let each VPN tunnel i is allocated with x_i amount of bandwidth. Let the utility function for traffic class i is $U_i(x_i)$. The objective of utility-based bandwidth allocation is to find the optimal bandwidth allocation x_i such that the overall utility $U = \sum_i U_i(x_i)$ is maximized. This utility-based bandwidth allocation problem is a multi-constrained optimization problem. Hence, over a VPN, the number of tunnels will define the number of utility functions required. Designing a bandwidth allocation scheme for this type is complicated, if not impossible to do. The scalable approach is to allocate bandwidth to different traffic classes within a tunnel. Assume that each traffic class has one utility function defined, $U_j(x_j)$. The utility-based bandwidth allocation problem now can be modeled as an optimization problem as given below:

It is defined as finding optimal bandwidth allocation x_{ij} for each VPN tunnel i belonging to traffic class j , so that the overall utility $U = \sum_j \sum_i U_j(x_{ij})$ is maximized.

5.4 Dynamic Bandwidth Management in IPVPN Tunnels

The VPNs have traditionally been deployed for reasons of economy of scale, but have either been statically defined, requiring manual configuration, or else unable

to offer any quality of service (QoS) guarantees. In order to achieve the statistical multiplexing in the providers' network, and thus increase the utility of the underlying network and give the VPN users some economical benefits, the bandwidth allocated to each tunnel should be dynamically adjusted [Juttner03][Hyong03]. The user may not be able to specify traffic accurately in the service level agreement (SLA). Hence, the admission control and tunnel setup must be supplemented with finer levels of dynamic control over the bandwidth allocation to maximize the network usage, and user satisfaction.

Link sharing [Floyd95] is an important concept using which service providers lease portions of their link to several VPN customers. To implement link sharing, the link capacity is statically partitioned into classes and assigned to the customers. Once a link is partitioned, the capacity allocations remain fixed irrespective of the actual usage by the organizations. This partition is made as per the service level agreement. Thus, if one of the organizations is not fully using its capacity, the free link capacity cannot be used by other organizations to accept new real-time sessions or even any other type of packet. For example, in telecommunication networks, once a service provider has leased a T1 line to an organization, it will not carry more than 24 calls of the organization through its line even if the trunk has required spare capacity. A better alternative is to fairly redistribute the free capacity by resizing (and possibly charge for it) depending upon the current usage of different organizations sharing the link.

In the natural extension of link sharing to virtual private networks [Gleeson00], instead of leasing bandwidth on a single link, the service provider leases bandwidth on a path to a corporate. The corporate tunnels its VPN traffic through this path as if it was a single *virtual link*. Such Virtual Leased Links (VLL) may be realized by IP tunnels [Simpson95] and label switched paths in IP networks, and virtual paths in ATM networks.

Use of hierarchical packet fair queuing algorithms [Zhang95] to efficiently implement link sharing has been proposed in literature. These algorithms ensure that each traffic class gets its requisite share of the link capacity, and carry out fair

redistribution of the residual unused bandwidth of a traffic class to other classes. The hierarchical packet fair queuing algorithms work at the scale of packet transmission time. Although they redistribute the free capacity of inactive classes to active classes, they cannot ensure, on their own that the redistributed capacity will be available for a predictable period of time. This residual capacity goes back to its respective class as soon as the sending rate of the class is increased. As a result, real-time sessions, which require a guaranteed Quality of Service (QoS) for their lifetime cannot utilize this excess capacity. Even the best effort sessions with minimum bandwidth requirement cannot get their minimum bandwidth guarantee from redistributed capacity. Therefore, capacity sharing using packet scheduling algorithms only, is limited to the *elastic* portion of the best effort traffic.

In order to support QoS in IPVPNs, the multihop virtual links will have to provide tight bandwidth and delay guarantees to their traffic. [Garg00] has shown that an inherent limitation of tunneling real-time traffic without explicit QoS marking is loss of statistical multiplexing over the link. The traffic cannot be sent on a virtual link at a rate higher than its allocated capacity, even if the underlying provider network has large spare capacity. Therefore, the resource sharing in service provider's network cannot be achieved without dynamically changing the capacity of virtual links. In [Duffield99] a dynamic resizing approach to resource management in VPNs has been proposed and is shown to result into significant (factor 1.5 to 3) statistical multiplexing gain. Similarly in [Goyal99] a pipe resizing approach for AT&T switched network is shown to result into a gain of 2. However the issue of fairness and protection has not been discussed in these approaches. Without fairness guarantees, a few IPVPN tunnels may take most of the bandwidth of the network resulting into starvation of other tunnels. Our work is based on fair bandwidth allocation to all the IPVPNs dynamically. We have also considered survivability of IPVPNs as part of QoS guarantee to IPVPNs.

In case of a multi-hop virtual links of IPVPN, the current capacity of a virtual link may be changed by sending a bandwidth increase or decrease request to all the participating intermediate devices in the service provider's network. The increase

request is admission controlled at each link of provider network. After processing these requests, the corresponding link capacities in the provider's network are adjusted.

Bandwidth management in IPVPNs tries to manage the bandwidth allocated to tunnels so that the VPN can be better adapted to the offered traffic. In other words, it is done to see that the users are satisfied or are getting an acceptable level of performance from the service provider. The ultimate aim is to maximize the network performance. When, due to unforeseen changes in the offered load and/or when the VPN design is not optimal, some tunnels can become under-utilized and others congested, i.e. all the tunnel bandwidth is already assigned to several connections or flows and new connections on this tunnel are being rejected. These rejected connections could be accepted if the traffic loads were better balanced or managed. One of the main objectives of bandwidth management is to minimize the Connection Blocking Probability (CBP), i.e. the probability that an offered user request is rejected due to insufficient capacity being available for the allocation of the new request. Minimizing CBP can also be seen as maximizing QoS for a particular service or maximizing the user satisfaction in the case where the tunnel carries only traffic of a single service.

Bandwidth allocation approaches can be divided into two categories: static and dynamic [Chan01]. In static approach, the allocated bandwidth remains fixed throughout, but in dynamic approach, it varies from time to time as per the user requirement leading to better network utilization. Static allocation is suitable for constant bit rate traffic (CBR) where dynamic scheme is suitable for variable bit rate traffic (VBR).

There are two different ways for bandwidth management: one is bandwidth reallocation and the other is path rerouting [Friesen96]. For example, figures 5.3 to 5.6 show both the ways of dynamically managing the bandwidth.

Figure 5.3 shows a network with four nodes and four links. The links L1, L2, L3 and L4 are provisioned with 12 Mbps, 30 Mbps, 50 Mbps, and 70 Mbps capacity each. Over link L1 we have two tunnels T1, and T2 provisioned with 5 Mbps capacity each. Thus, the spare capacity available over L1 is 2 Mbps. Over link L2, tunnel T3 is provisioned with 10 Mbps, and hence the spare capacity over L2 is 20 Mbps. There are no tunnels currently provisioned over links L3 and L4.

In bandwidth reallocation, if there are few tunnels that are congested and few are under utilized, then the bandwidth of the congested one's can be increased without the need for rerouting. The required bandwidth can come from two possible sources: the available bandwidth of the physical links, not assigned to any other purpose; and the underutilized bandwidth already assigned to other tunnels with a low load (pre-emption). Figure 5.4 and figure 5.5 depict two different situations of bandwidth reallocation. In figure 5.4, T1 needs an additional bandwidth of 2 Mbps. As the spare capacity is sufficient enough, this requirement is satisfied. In figure 5.5, T1 needs an additional amount of 4 Mbps that is more than the spare. Assuming that T2 is underutilized, 2 Mbps is released from it. Now, T1 is given with spare 2 Mbps plus 2 Mbps released from T2.

If bandwidth re-allocation cannot be applied because one or more physical links in the same physical path have no available bandwidth and bandwidth from other tunnels sharing the same physical path cannot be utilized either (there is not enough bandwidth or pre-emption is not allowed). In such a scenario, the only possibility is to re-route the tunnels through other physical paths in order to have enough available bandwidth and then increase bandwidth to the congested tunnels. Again, there are two ways of doing this: re-route the congested path through a physical path with enough available bandwidth to meet its needs; or re-route other VPN tunnels going through the same physical path as the congested tunnel in order to release bandwidth and make room to increase it to the congested VPN.

Figure 5.6 depicts this scenario. Let node 1 is the source and node 2 is the destination. Here, T1 needs 8 Mbps additional bandwidth in addition to its currently

allocated quota of 5 Mbps. This makes total of 13 Mbps. But, the capacity of link L1 is 12 Mbps only. Irrespective of link L2, link L3, and link L4's utilization level, the requirement of T1 cannot be satisfied by the link L1. So, T1 needs to be rerouted to a different path. Here, T1 is provisioned over L2, L3, and L4, as the spare capacity of L2 is 20 Mbps that is enough for a 13 Mbps requirement, and both L3 and L4 have sufficient capacity. The tunnels are formed over L2, L3, and L4 to provide connectivity between source node 1 and destination node 2. One of the main objectives of our work in this area is to minimize the Request Blocking Rate (RBR).

Bandwidth reallocation is preferable to re-routing because it is less traumatic for the already established connections going through the affected VPNs. Another possibility in re-routing is to have a transient situation where both the old and the new substitute tunnel coexist for some time. New connections are made in the new tunnel and the old one endures while it still has live connections. However, this option requires more bandwidth.

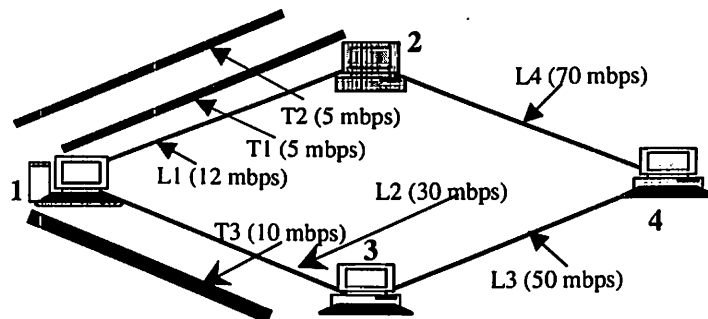


Figure 5.3 Initial Tunnel Setups and Link Capacities

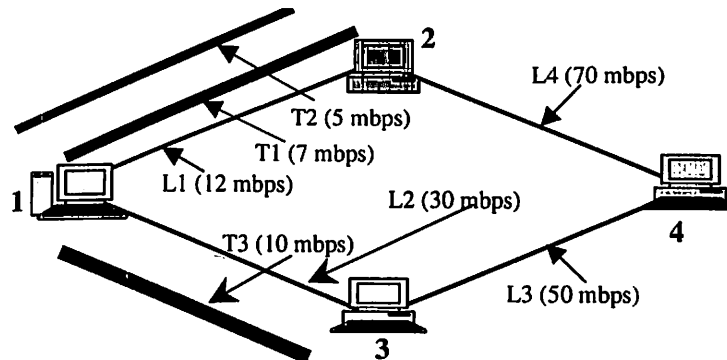


Figure 5.4 Bandwidth Reallocation when T1 needs 2 mbps additional capacity

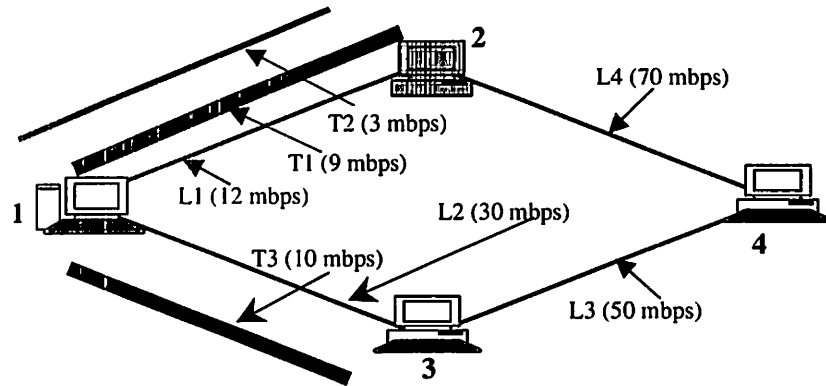


Figure 5.5 Bandwidth Reallocation when T1 needs 4mbps additional capacity.

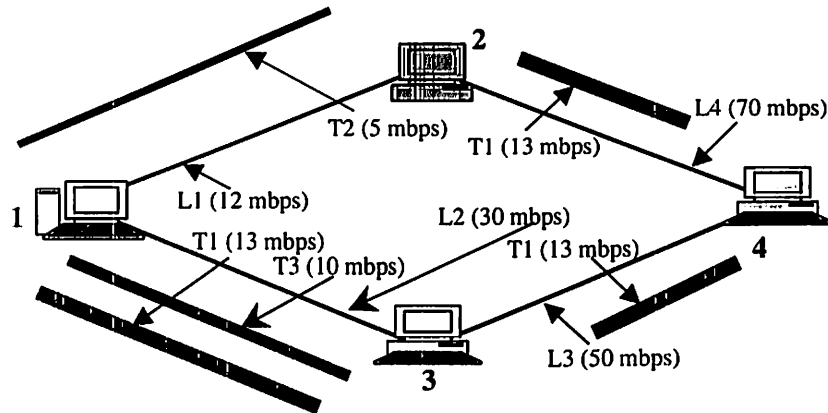


Figure 5.6 Bandwidth Path Rerouting when T1 needs 8mbps more

5.5 Literature Survey on Dynamic Bandwidth Management in IPVPNs

5.5.1 Duffield's Hose Model VPN

[Duffield02] et al. proposed a *hose model* that analyzes the need for aggregating the traffic that belongs to a single VPN. They analyzed heterogeneous traffic, consisting of voice and data calls, which is more similar to the traffic served in IPVPNs. They reserve capacity in the core network, where the capacity needs to be dynamically allocated, because of the uncertain profile and volume of the traffic within each VPN. To manage resources so as to deal with this increased uncertainty, they employed two techniques: statistical multiplexing and capacity resizing. They

used *statistical multiplexing* to decrease aggregate bandwidth requirements. Later they *resized* the initial allocation on the basis of online measurements.

The set of hoses making up a VPN have a common QoS guarantee, and hence the service provider can multiplex all the traffic of a given VPN together. These techniques can be applied on both access links and network internal links. Service providers can allocate statically capacity to all the traffic on a VPN. Alternatively, providers can allocate some initial capacity and then resize on the basis of online measurements. The simulation results presented in this work shows that the capacity requirement gain is around 2 over the customer pipe model VPN, where each flow reserves the required capacity.

5.5.2 Goyal's Distributed Open Signaling Architecture

[Goyal99] et al. described Distributed Open Signaling Architecture for resource management. They made their solution experimented over an IP telephony network. Our interest here is the resource management part. To support large scale services, admission control and scheduling must be used in the core network on an aggregate basis. For example, phone calls with the same source and destination IP address should be aggregated and a certain amount of bandwidth should be allocated to these aggregates. This allocated bandwidth is periodically resized. The amount by which this resizing should be made is derived from a prediction of aggregate voice flows that are expected in future. The simulation results show that this resizing on a large scale IP telephony network once every 5 minutes saves a factor of around 2 in capacity on all links. Although this model uses voice traffic only, it shows the benefit of the dynamic resizing on bandwidth partitions.

5.5.3 Garg's Stochastic Fair Sharing Approach

[Garg00] et al. proposed a stochastic fare sharing approach for bandwidth management. The capacity allocation is fair between a number of active VPNs in the network. They have considered normalized usage of a fraction of link bandwidth. The

capacities are dynamically modified, i.e. increased upon session arrival and decreased on completion. They claim that in place of static link sharing, it is better to have fair redistribution of the free capacity by resizing the capacity based on the current usage of different VPN customers sharing the link. Their SFS approach does not allow few virtual links to take up most of the bandwidth of the network leading to starvation of other virtual links using the same physical network. As a result of SFS, the redistributed capacities, which are available for session lifetimes, can be used by real time traffic as well as VPN traffic. Their approach ensures fairness and protection both. The admission control algorithm decides whether increase of logical link capacity will lead to any failure. If it is acceptable, then the corresponding weights in the underlying packet scheduler are updated to reflect the change in bandwidth allocation.

Their scheme assigns priorities to VPNs on the basis of their normalized usage. The normalized usage of a logical link i is given as $n_i = r_i / c_i$ where, r_i is the capacity already reserved by logical link i or VPN_i and c_i be the capacity allocated to link i . Logical links are assigned priorities on the basis of their normalized usage. This scheme gives higher priorities to sessions of logical links having low normalized usage and thus attempts to equalize the normalized usage of different logical links. If the normalized usage of a link is close to its' fair share, then it is not required to have a large value of trunk reservation for this link. In this scenario, trunk reservation is rather reduced. The fair share for each logical link is dynamically computed. If the difference between the fair share and the current reservation is less than its' static trunk reservation, then the trunk reservation of the link is set to the difference. Otherwise, the trunk reservation is set to static value. If the usage of logical link is less than its' fair share, then the free capacity is fairly redistributed among heavily loaded links after keeping aside a small trunk reservation. This scheme gives protection to classes with low session arrival rate against classes with high session arrival rates by ensuring them a low blocking probability. Their simulation results show that SFS gives improved throughput while achieving max-min fair sharing. 94 percentages of virtual links achieve an equivalent capacity within -10 to $+5$ percent of their max-min fair capacities.

5.5.4 Other Resource Allocation Approaches

[Lin96] et al. proposed virtual path bandwidth resizing, routing and rerouting algorithms for managing virtual paths in ATM networks. They have considered the bandwidth utilization and the different degrees of urgency between increase and decrease of the bandwidth. They have proposed several heuristics for this work.

[Cao99] et al. proposed utility max-min fairness. In IP networks, the traditional max-min bandwidth allocation results in disparity in the application layer performance, though a fair amount of bandwidth is allocated to the application. The bandwidth max-min flow control mechanism tries to maximize the bandwidth allocated to each application, without differentiating the applications. Their approach stressed on application-oriented approach to fairness. They concluded that to design a feasible utility-based bandwidth allocation scheme, one has to trade-off the generality and simplicity in the specification of utility functions. Each user could define his or her own utility for some application.

[Banchs00] et al. proposed a new architecture called as Scalable Share Differentiation (SSD) to allocate network resources on a user basis. It also provides proper isolation for elastic and real time traffic, based on the agreements between users and service providers. Every user i contracts a share S_i with the provider. When there is not much bandwidth for that traffic type is available, packets with lower effective share should be dropped with higher probability.

5.6 Algorithms for Dynamic Bandwidth Management

This part of our work proposes new dynamic bandwidth allocation algorithms for IPVPNs. These algorithms partition the available bandwidth between active traffic classes, so that it adapts to the state of the network links in order to produce maximal end user's utility. These algorithms include admission control, bandwidth allocation and bandwidth partitioning as described in section 5.1. Our algorithms use utility-

based dynamic bandwidth allocation approach. Utility is treated as a measurement of the level of end user's satisfaction. This utility is representative of the quality of service ensured by the IPVPN to its' customers. We have discussed utility-based resource allocation in previous chapter.

The design problem for bandwidth allocation in an IPVPN is how to allocate the bandwidth of a core ISP link that is used by a number of IPVPN tunnels and by the normal Internet best effort traffic. This allocation should generate more revenue for the ISP and also it should satisfy the performance requirements of all users. One of the issues here is how to share the bandwidth between different VPNs and the other one is sharing the bandwidth within each VPN. VPN customers will be completely satisfied if the link utilization is very low or if the traffic belonging to different VPNs is fully isolated i.e. if the network is fully partitioned. The partitioning of link bandwidth to various VPN customers can be virtual (at the admission point) or physical (by setting the scheduling parameters in priority queues at intermediate routers). Our bandwidth allocation algorithm in this section addresses these above issues in detail.

We have used an agent based resource management system for this part. The VPN tunnel path from source to destination is computed using the hill climbing shortest path heuristics defined in algorithm 4 (page 98). While constructing this path we consider the administrative cost of the link as the cost for deciding the suitability of the link. We use one agent process for each user at the Customer Premises Equipment (CPE) device and Bandwidth Managers (BMs) at all the ISP domains including the CPE. The agent process is created by the BM for each user of the tunnel that is responsible for monitoring the bandwidth usage of the corresponding user. It is dynamic in the sense that user traffic is monitored periodically. If it is necessary to resize the bandwidth currently allocated to a user in the VPN, then the corresponding agent process sends a request to the local BM. The BM in turn sends this to all the BMs in the path of the tunnel that was earlier computed. A distributed consensus protocol is run by all the BMs to reach an agreement. The interaction between a pair of nodes is as shown in figure 5.7.

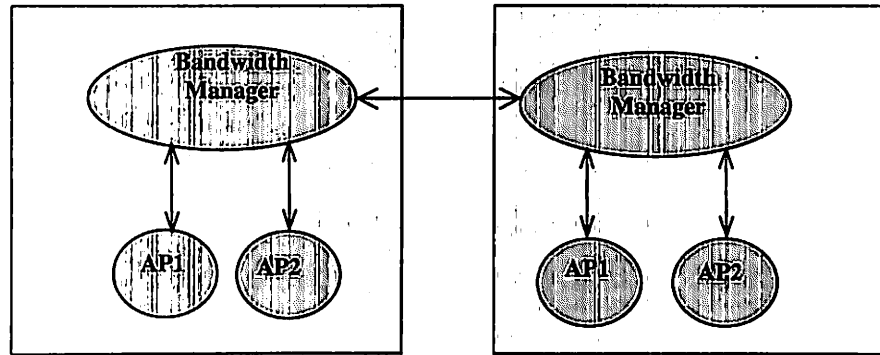


Figure 5.7 Relationship between Agents and Bandwidth Managers

For better understanding let us look at an example. Let the link capacity be C . Let the utility function for every user 'i' in the tunnel be $u_i(b)$ that is computed by the agent process using equation 1.

$$u_i(b) = R/b_i \quad (1)$$

Here, R is the traffic rate, and b_i is the allocated bandwidth.

Let the single link serve several VPN tunnels. Let there be a partitioning parameter α for each tunnel 'i'. The bandwidth manager computes the bandwidth that is to be allocated to each tunnel using the equation given below.

$$C_i = \alpha_i * C \text{ Where } 0 < \alpha_i \leq 1 \quad (2)$$

The spare capacity over a link with N number of tunnels is given as:

$$SC = C - \sum_{i=1}^N C_i \quad (3)$$

Let each tunnel i support n_i number of users. The bandwidth manager allocates initially b_i amount of bandwidth to the i^{th} user in tunnel 'i' as:

$$b_i = C_i / n_i \quad (4)$$

Using equations 1, and 4, the agent process computes the utility function for every user in the tunnel. Two thresholds (Low and High) are defined for the utility function of any user. The following heuristics are used by agent processes to send a request to the bandwidth managers:

- If utilization is greater than or equal to High, then an increase (for additional δ amount) request is sent.
- If utilization is lower than or equal to Low, a request is sent to decrease the bandwidth.

Upon the receipt of a request, the local bandwidth manager broadcasts a request message to all the local agent processes asking for their utility functions. It also sends this bandwidth increase or decrease message to all the bandwidth managers in the tunnel path. Bandwidth manager computes the total utility of the tunnel with ' n_i ' number of users by using:

$$U_i(b) = \sum_{i=1}^{n_i} u_i(b) \quad (5)$$

Every tunnel also has an upper threshold, i.e. $Max_U_i(b)$. Bandwidth manager uses the conditions given in the following equation for making a decision regarding bandwidth reallocation or path rerouting.

$$b_i = \left\{ \begin{array}{l} b_i + \delta, \text{ if } U_i(b) \leq Max_U_i(b) \ \& \ \delta < SC \\ b_i + \delta_{victim}, \text{ if } U_i(b) \leq Max_U_i(b), \delta > SC \ \& \ \delta \in V \\ (((\alpha_i + \sigma) * C) / n_i) + \delta_{victim}, \\ \text{if } Max_U_i(b) < U_i(b), \delta < SC, 0 < \sigma \leq 1 \ \& \ \delta \in V \\ \text{Re compute } TunnelPath, \text{ Otherwise} \end{array} \right\} \quad (6)$$

The equation number 6 depicts four different cases of bandwidth management. Here, δ is the additional amount requested. Requested amount is allocated from spare if we have spare capacity available in the link and current utilization is under the upper bound. This is characterized by the condition 1. If spare capacity is not available, and tunnel utilization is not exceeding the threshold, then choose a victim, release the required amount of bandwidth from it, and give this to the requesting process. This is formulated in condition 2. Third case shows that utilization has gone up, but the spare is sufficient to handle the request, then first resize the tunnel bandwidth by changing α and then add an additional δ requested. For this case as well choose a victim, and decrease its bandwidth by an appropriate amount. Fourth case shows that the spare is not available and also the utilization of the tunnel has gone up. This case is of bandwidth path rerouting. To handle this, find out another shortest path between source and destination of the tunnel and reallocate bandwidth to it.

5.6.1 Distributed Message Communication

We describe here, the different types of messages that are required for communication between BM and agent processes. Let there be three agent processes at one branch office intranet, a bandwidth manager, and the tunnel paths span over two domains ISP1 and ISP2 as shown in figure 5.9. The format of request messages is *(source, destination, originator, bandwidth, request_type)*. The Agreement messages are consensus i.e. OK messages. The format of reply messages is *(source, destination, utility)*. Figure 5.8 shows the message receive events between various agent processes and bandwidth manager. Let event 1 be a message receive event for bandwidth increase sent by Agent Process 1 (AP1) as user's utilization has increased. In response to this, BM1 sends request messages 2 and 3 to other bandwidth managers in the tunnel path for allocating same amount of additional bandwidth to Agent Process 1. It also sends request messages 4 and 5 to other agent processes to collect other users' utility in the same tunnel. Replies from AP2 and AP3 are message receive events 7 and 9. All the three BMs in the tunnel path run a consensus protocol to reach at agreement. This is because; the requirement should be met throughout the

complete path. Consensus messages are 6 and 8. After agreement, BM1 sends decision to AP1, AP2 (10,11) and Agreement message to BM2 and BM3 (12,13).

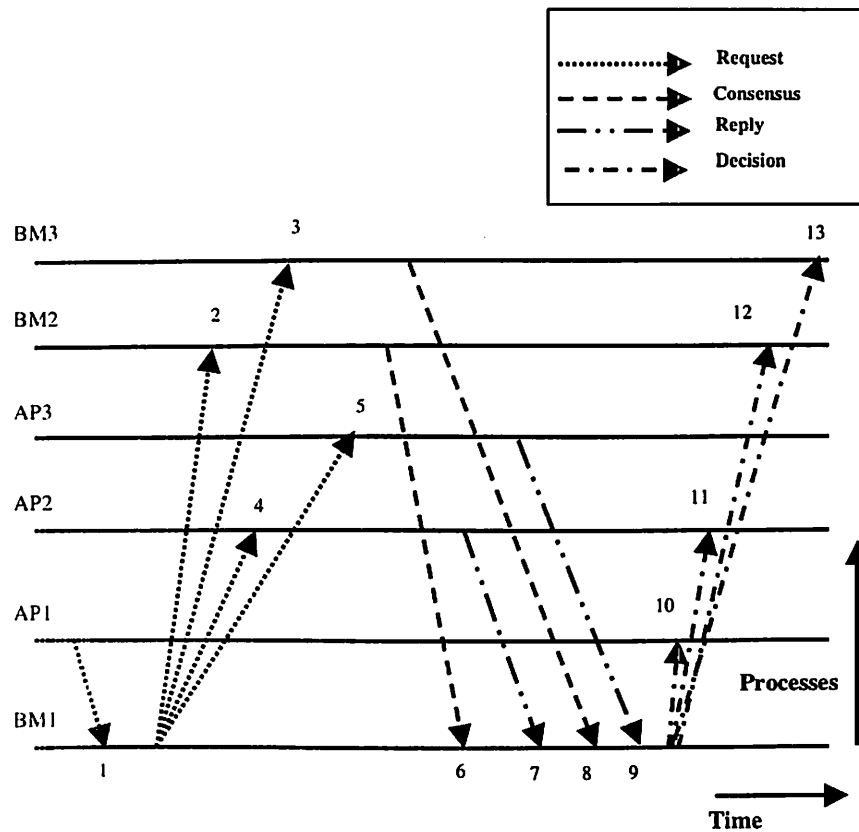


Figure 5.8 Distributed Message Communication

5.6.2 Proposed Algorithms

This section consists of five algorithms for dynamic bandwidth management. Algorithm 3 is the main part of our work in this section where the bandwidth manager creates an agent process for every user at the start of the connection request. It allocates an initial amount of bandwidth to every user. Algorithm 4 is used to

compute quickly a shortest path between every source destination pair of the Intranet VPN. The agent process uses algorithm 5 and algorithm 6 to compute the user utility. Algorithm 7 is used to reach at a consensus between all the bandwidth managers. The data structures that are used in algorithm 3 are described as below:

- DQ: A queue to hold the deferred tunnel paths for bandwidth path rerouting.
- ActiveList: A linked list that has an entry for every active user in the tunnel.
- SLA: The service level agreement to be supported for source destination pairs.
- TS_{ij}: A set that contains the tunnels that are formed over a single link l_{ij}.
- P: A set consisting of tunnels that form a path between a source destination pair.

An application of these algorithms is explained using figure 5.9. Let the shortest tunnel paths computed by step 2 of algorithm 1 for both the branches be S1 → C1 → C3 → C4 → C7 → D, and S2 → C2 → C1 → C3 → C4 → C7 → D. Step 3 computes number of tunnels over each link. DQ is a deferred queue to hold the tunnels which are reformed. Step 4 computes initial capacity for each tunnel in the link. Step 5 computes initial capacity of each user in the tunnel. Step 6 computes the spare capacity on the link. Step 7 is the main part of this algorithm where the BM interacts with agent processes and peer BMs. When the first packet comes from a user, the network element sends new user request to local BM that creates an agent process (AP), makes an entry of this active user in a Linked List called *ActiveList*. BM sends the initial bandwidth allocated to the agent process. If the message received is for bandwidth increase from any of the earlier existing agent processes then it collects the utilities of each user in the tunnel by using algorithms 5 and 6. BM then decides to either resize the capacity allocated to another user or the capacity allocated to the tunnel. The victim agent is selected by using a greedy approach for simplicity reasons. The victim agent process is the process that has lowest utilization among others. This victim process can only be selected for a maximum number of times as defined in the variable limit.

Algorithm 3 Dynamic Bandwidth Management

Input: Network Graph, Source Destination Pairs, and Maximum Utilization of each Tunnel (MAX_UT).

Output: Initial Bandwidth allocated to each user, and dynamic increase or decrease of the bandwidth allocated to active users.

Step 1: Initializations: $C \leftarrow$ Link Capacity, $U_T \leftarrow 0$, $MAX_UT \leftarrow$ Maximum Utilization, $TS_{ij} \leftarrow \phi$, $P \leftarrow \phi$, $DQ \leftarrow \phi$, Limit \leftarrow Max number of times a process can be selected as victim, Count [victim] $\leftarrow 0$.

Step 2: Compute Tunnel paths for all the SLAs' to be supported.

For every SLA_i to be supported do

$T_i = \text{ShortestPath}$ (source, destination); //Algorithm 4

$P = P \cup T_i$;

Endfor;

Step 3: Compute the set of tunnels that use a single link.

For every link l_{ij} on Service Provider's domain do

Select an element 's' from $P \cup DQ$;

If ($l_{ij} \in s$) then $TS_{ij} \leftarrow TS_{ij} \cup s$;

Endfor;

Step 4: Compute the initial bandwidth to be allocated to Tunnel $_i$ of link l_{ij} .

For every Tunnel $_i$ in TS_{ij} do

$C_i = \alpha_i * C$;

Endfor;

Step 5: Compute the initial bandwidth to be allocated to user $_i$ with n_i number of users in Tunnel $_i$

For every user $_i$ in Tunnel $_i$ do

$b_i = C_i / n_i$;

Endfor;

Step 6: Compute the spare capacity of the link l_{ij}

$$SC = C - \sum_{i=1}^{n_i} C_i$$

Step 7: When a new user packet arrives, Bandwidth Manager (BM) creates an agent process (AP), sends the initial bandwidth, and runs in an infinite loop.

While (True) do

ReceivePacket;

If (NewUser) then // Connection Establishment

$i = \text{QueueInWhichPacketArrives}$;

If (ExistsinActiveList (i) == false) then

AddtoActiveList (i);

Increment SizeofActiveList;

Fork AgentProcess (i);

Send BandwidthAllocated (b_i) to Agent Process;

Endif;

Else

// Dynamic Bandwidth Management

```

If (BandwidthIncreaseRequest) then
  Run ConsensusProtocol; // Algorithm 7
  While SizeofActiveList  $\neq$  NULL do
    Get Element (i) from ActiveList;
    Send RequestMsgforUtility to Agent Process;
     $U_i$  = Receive UserUtility; //Algorithm 5
     $U_T = U_T + U_i$ ;
  Endwhile;
  If ( $U_T \leq$  MAX_UT) then
    If ( $\delta \leq$  SC) then
      If (ConsensusOK) then
         $SC \leftarrow SC - \delta$ ;
         $b_i = b_i + \delta$ ;
        Send BandwidthAllocated ( $b_i$ ) to AP;
      Else
         $SC = SC + b_i$  ;
        Remove RequestfromActiveList;
        Decrement SizeofActiveList;
        Kill AgentProcess (i);
         $T_i \leftarrow$  Recompute the Path; // Algorithm 4
         $DQ \leftarrow DQ \cup T_i$ ; Go to step 3;
      Endif;
    Else
      Victim  $\leftarrow$  ProcessWithLowestUtilization;
      Count [victim]  $\leftarrow$  Count [victim] + 1;
      If (Count [victim]  $\geq$  Limit)
        Victim  $\leftarrow$  ProcessWithNextLowestUtilization;
       $b_{victim} = b_{victim} - \delta + SC$ ;
       $b_i = b_i + \delta$ ;
       $SC \leftarrow 0$ ;
      Send BandwidthAllocated ( $b_i$ ) to AP;
    Endif;
  Else
    If ( $\delta < SC$ ) then
      Victim  $\leftarrow$  TunnelWithLowestUtilization;
      Count [victim] = Count [victim] + 1;
      If (Count [victim]  $\geq$  Limit)
        Victim  $\leftarrow$  TunnelWithNextLowestUtilization;
       $b_{victim} = (((\alpha_{victim} - \sigma) * C) / n_i) - \delta$ ;
       $b_i = (((\alpha_i + \sigma) * C) / n_i) + \delta$ ;
      Send BandwidthAllocated ( $b_i$ ) to AP;
    Else
       $SC \leftarrow SC + b_i$  ;
      Remove RequestfromActiveList;
      Decrement SizeofActiveList;
      Kill AgentProcess(i);
       $T_i \leftarrow$  Recompute Tunnel Path // Algorithm 4
       $DQ \leftarrow DQ \cup T_i$  ; Go to step 3;
    Endif;
  Endif;
Else

```

```

If (BandwidthDecreaseRequest) then
  For each BM in the TunnelPath
    Send BandwidthDecreaseRequestMessage;
  SC = SC +  $\delta$ ;
Else // Termination of the connection
  Remove RequestfromActiveList;
  Decrement SizeofActiveList;
  Kill AgentProcess (i);
  SC = SC +  $b_i$ ;
Endif;
Endif;
Endif;
Endwhile;

```

Algorithm 4 Shortest Tunnel Path using Hill Climbing

Input: Network Graph, Source, and Destination Pairs.

Output: Tunnel Path (T) for every Source Destination Pair.

Step 1: Initializations: $T \leftarrow \phi$, $I \leftarrow$ Source (S), $D \leftarrow$ Destination.

Step 2:

While ($I \neq D$) do

i. Select neighbor (N) along the path with least cost;

ii. $T \leftarrow I \cup N$;

iii. $I \leftarrow N$;

Endwhile;

Algorithm 5 Algorithm for computing User Utility and the bandwidth increase or decrease (Agent Process)

Input: Maximum User Utility (MAX_UU), Minimum User Utility (MIN_UU), and Allocated Bandwidth (b_i).

Output: The current User utility or delta, the increase or decrease in the allocated bandwidth.

Step 1: Initializations: User utility (U_u) $\leftarrow 0$

Step 2: While (TRUE) do

Receive message from Bandwidth Manager;

If (Message type is BandwidthAllocated) then

$b_i =$ bandwidth;

Else

If (connection is active) then

Rate (R) \leftarrow Token Bucket Parameters;

$U_u = R / b_i$;

Send User utility (U_u) to BM;

Else

Send ConnectionCloseRequest;

Endif;

Endif;

```

If ( $U_u > \text{MAX\_UU}$ ) then
     $\eta = \text{Incr}()$ ; // Algorithm 6
     $\delta = \eta * R$ ;
    Send BandwidthIncreaseRequest of  $\delta$  to BM;
Else
    If ( $U_u < \text{MIN\_UU}$ ) then
         $\delta = 0.8 * b_i$ ;
        Send BandwidthDecreaseRequest of  $\delta$  to BM;
    Endif;
Endif;
Endwhile;

```

Algorithm 6 Computing η using averaging approach.

Input: User utilizations at different time intervals (U_u)

Output: Return η

Step 1: Let $[(t_1, t_2), (t_2, t_3), \dots]$, be the size of the observing window and $[(U_u^1, U_u^2), (U_u^2, U_u^3), \dots]$ be the user utilizations at those times.

Step 2: Let η_p and η_c be the slope of incoming traffic in the previous and current intervals respectively.

Step 3: Compute $\eta_p = (U_u^{n-1} - U_u^{n-2}) / (t_{n-1} - t_{n-2})$

Step 4: Compute $\eta_c = (U_u^n - U_u^{n-1}) / (t_n - t_{n-1})$

Step 5: Compute $\eta = \rho * \eta_c + (1 - \rho) * \eta_p$, Where $(0 < \rho < 1)$

Step 6: Return η

Algorithm 7 Consensus Protocol

Step 1: Initializations: Leader \leftarrow Source BM, Follower Set (FS) \leftarrow BMs in the tunnel path, Majority $\leftarrow 0$, Vote [tunnel_i] \leftarrow False, SC_{tunnel_i} \leftarrow Spare Capacity of tunnel_i.

Step 2: Leader broadcasts VoteRequest to all the elements of FS.

Step 3: For every element of FS
 If $((\text{Vote}[\text{tunnel}_i] == \text{False}) \ \&\& \ (\text{SC}_{\text{tunnel}_i} > \delta))$
 Send Consensus to the Leader;
 Vote [tunnel_i] = True;

Step 4: For every Consensus message from a Follower, Leader computes
 majority = majority + 1.

Step 5: If $(\text{majority} == \text{Length}(\text{FS}))$ then Leader broadcasts ConsensusOK message to all the BMs in FS.

Selecting an agent process and reducing the capacity of this user to satisfy the requirement of another user will be done only for certain number of times. This is done for avoiding starvation of selecting VPN users that underutilize their capacity over a long period of time. Once a victim agent is selected, spare capacity plus the remaining requirement is taken from the victim agent. On the other hand, if all the users in a single tunnel are efficiently utilizing their allocated quota, then tunnel capacities are resized by changing the partitioning parameter (α) and allocating the required amount of bandwidth to meet the increasing demand of the current user. If neither of the above said conditions are true then entire tunnel path is recomputed and again the same set of steps are repeated expecting that we will get a path with higher bandwidth availability. Before a BM decides to satisfy the increased demand of a dynamic request, it first ensures with other BMs in the path regarding whether they have sufficient capacity available to satisfy the same. This is done by running a consensus protocol (Algorithm 7) where the source BM becomes the initiator of this protocol and will be called as leader. It polls votes from other BMs in the path one by one. If it gets OK reply from all the followers then declares that consensus is reached after which it goes for dynamic bandwidth management. If the request is for termination, then it terminates the agent and increases the spare by the current allocation of the terminating user.

Algorithm 5 models the behavior of an agent process that monitors the user activity. It computes the user utilization. If utilization is greater than the SLA defined upper bound, it computes η an incremental step using exponential averaging defined in algorithm 6. If the user utilization is below the lower bound, it requests for a decrease in bandwidth. This is based on a deterministic value, as we do not need to predict the future samples in this case. Algorithm 6 predicts η required due to dynamic traffic behavior at different time intervals.

5.6.3 Implementation

We implemented our algorithms in a distributed platform using Unix threads and communication APIs. Each BM forks one thread for a single user. These threads

communicate with the BM using message send and message receive primitives. The communication between peer bandwidth managers in different domains is also done using sends and receives. The synchronization between threads and the BM as well as between BMs is done by Unix conditional variables and mutexes. The network topology is input to the algorithm using a two dimensional matrix, where the row, column represent a link and the cross-section represents the cost over the link. The BMs are tied to the concerned node in the topology by a TCP/IP socket. Each user process is created as a foreground process, but BM, and agent processes are created as background processes. User processes communicate with Agent and BMs using socket API. Here, user processes act as packet generators.

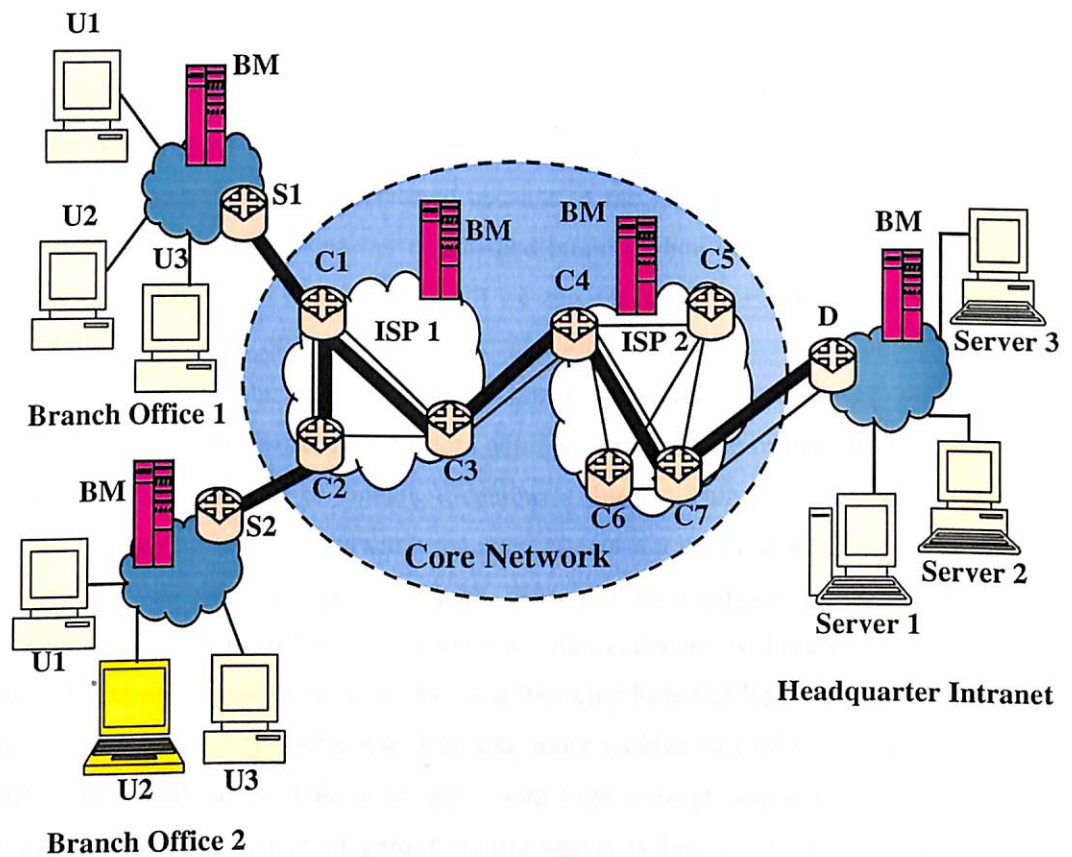


Figure 5.9 Intranet VPN Scenario with Bandwidth Manager

5.7 Simulation Results and Analysis of Dynamic Bandwidth Management Algorithms

There are two general methods for the performance evaluation of a network [Pitts00]. These are measurement techniques and predictive techniques. In this work we have used predictive techniques that use a simulation model to analyze the performance of the proposed algorithms. The results presented as part of the simulation results show the performance and efficiency by analyzing the impact of different parameters like bandwidth utilization, number of tunnels, average traffic rate, and request blocking rate etc. The results of our dynamic bandwidth allocation algorithms proposed in section 5.6.2 are as shown in the simulation results shown in figures 5.10 to 5.12. Figure 5.10 shows that with only one tunnel we do not get more benefit, as the curve is linear up to 50 users. But, we do get utilization better for this range (0 to 50) if numbers of tunnels are more. For large number of users, numbers of tunnels do not play any role in reducing the bandwidth utilization significantly. The behavior is nearly the same. This is shown in the graph for number of users equal to 50 and above. This behavior is resulted because of the fact that we have to reroute many user packets from the earlier established tunnels when the number of users is large. But, for links with more tunnels still we can get a better performance than the links with less number of tunnels. Figure 5.11 shows that irrespective of number of tunnels, up to some value of traffic rate (30 kbps), utilization decreases very slowly. But as we increase the traffic rate, for less number of tunnels the utilization increases where as for more number of tunnels, it decreases sharply. This is because when the average traffic rate is low, bandwidth manager allocates a major share of bandwidth to users, and when the average rate is high, the reallocation happens frequently. The more the number of tunnels on a link more are the reallocations. Figure 5.12 shows that with less number of tunnels, the Request Blocking Rate (RBR) increases linearly with increase in average traffic rate. But with more number of tunnels in a link, the RBR drops significantly. This is because, with high average traffic rate and more number of tunnels, chances of getting victim agents is larger. These victim agent processes are those that underutilize their capacity.

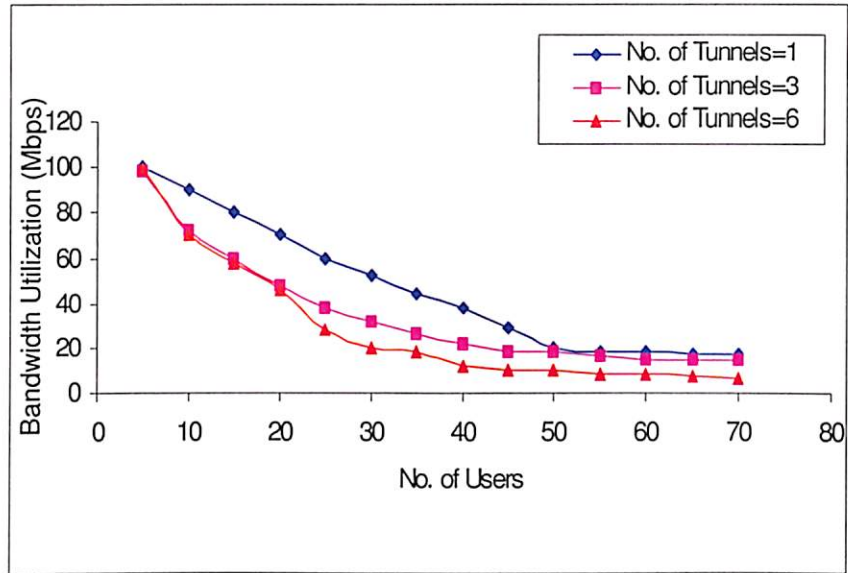


Figure 5.10 Utilization of a link with number of users

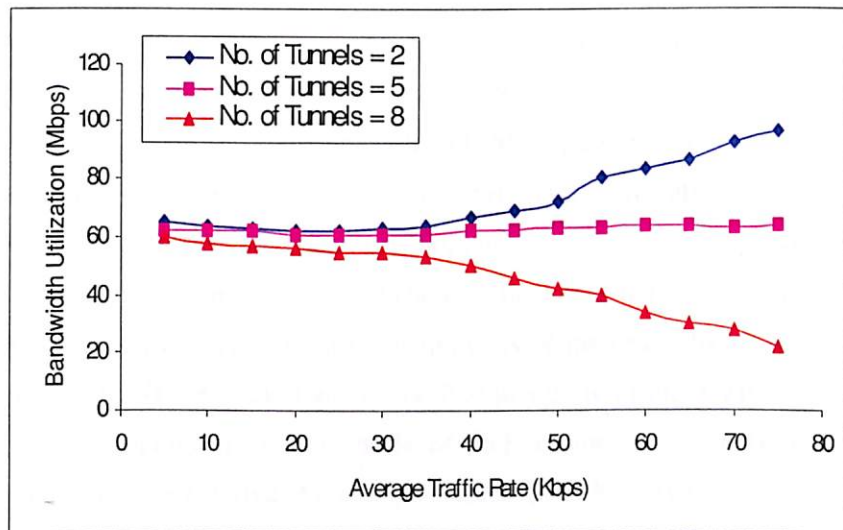


Figure 5.11 Utilization of a link with Average Traffic Rate

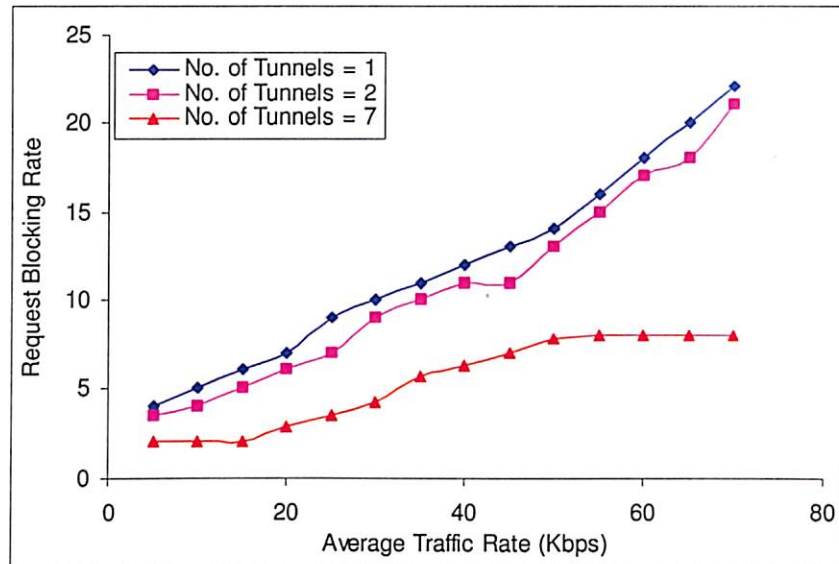


Figure 5.12 Request Blocking Rate with Average Traffic Rate

5.8 Summary

Managing bandwidth that is a critical resource in IPVPNs is a complex task. First we argued (section 5.1) that although IPVPN is a logical network, resource management is an integrated task that should be supported by other network management tasks. Then we defined several bandwidth allocation mechanisms including complete sharing, complete partitioning, virtual partitioning and trunk reservation etc. that are common in multiclass IP networks that also apply to our IPVPN model. Next, we stressed on the requirements of the utility-based bandwidth management in IPVPNs that will satisfy users. Bandwidth to different VPN customer will be managed dynamically on the basis of their online utility. This will also increase the chances of additional revenue generation for VPN service providers as it can service more number of customers with user satisfaction. We then described different approaches for managing bandwidth dynamically in an IPVPN including bandwidth reallocation, and path re-routing. Then we presented the literature survey on bandwidth management approaches in IPVPNs including hose model, distributed open signaling architecture, stochastic fair sharing approach and others.

Section 5.6 presented set of novel solutions or algorithms to the dynamic bandwidth allocation problem in IPVPNs. We used an agent based approach to this problem with bandwidth managers and agent processes carrying out this allocation task keeping in view the user satisfaction and the efficient utilization of network resources hence, enabling the service provider to generate additional revenue. For this work we have also defined a distributed paradigm of message communication between several entities involved. The bandwidth allocation to various users is managed as per their need i.e. if anyone underutilizes his allocated quota, some amount of this bandwidth can be given to another needy one. These algorithms will facilitate multimedia applications running over IPVPNs requiring more bandwidth more frequently.

Section 5.7 presented the analysis of our bandwidth management algorithms. Different behaviors plotted show the performance of our algorithms in terms of average traffic rate, bandwidth utilization, request blocking rate, etc.

Chapter 6

Restoration in IP Virtual Private Networks

6.1 Protection at Different Layers of OSI Stack

Guaranteeing QoS in IPVPNs also involves restoration of VPNs to ensure network survivability. Provision of survivable VPNs makes the VPN solution a robust proposal. The VPN must be capable of maintaining service continuity in the presence of faults. The types of failures in an IPVPN are link and node failures. Link failures are the outcome of disruptions like fiber cuts, power outage, etc. that causes the link to work incorrectly or stops communication flow in the link altogether. The routers/switches in the core network are considered as nodes. Node failure occurs if a node ceases to operate during some disruptions like the router shutdown or system reboots, etc. A study made by [Markopoulou04] reveals that around 15% of the network failures are because of planned maintenance. The majority of the unplanned failures are individual link failures (69.2%) with about 16.5% node failures. Single link failures and single node failures are the most common types of failure and thus most literature [Grover87][Herzberg95][Iraschko96][Kodialam03][Li03] have based the restoration model on these failures only. Restoration strategies should provide fast recovery with bandwidth guarantees from the common failures and use overprovisioning techniques for increasing the probability of recovery in multiple failures, which are less common. Over-provisioning refers to the extra bandwidth reserved for use when unexpected events occur.

The goal of restoration can be achieved by implementing restoration strategy at different layers in the OSI network protocol stack. Physical layer based strategies are much more static as it is built into the hardware devices. By developing more reliable network elements at device level this aim can be fulfilled. However, this solution does not override the need for failure detection and restoration capabilities to be implemented at different network layers. Physical layer strategies deliver the fastest restoration time [Stern99][Wu93] around 50ms. For physical layer strategies, a network is divided into interconnection of smaller network spans, and each of the network span is independent of each other. The failure in a network span is localized and restoration is based on using the resources in the same network span only. Examples of such strategies are Synchronous Optical Network's (SONET) Dual-Fed Unidirectional Path Switched Ring (UPSR) [Bellcore95a] and Bidirectional Line-Switched Ring (BLSR) [Bellcore95b]. Both strategies assume a ring topology structure for the network span and use pre-planned backup bandwidth allocation.

Link layer restoration strategies are mainly used for connection-oriented networks such as Asynchronous Transmission Mode (ATM) VPNs [Hassan00] and Multi-Protocol Label Switching (MPLS) VPNs [Awduche99][Boyle02]. These restoration strategies try to restore the traffic for the LSPs that are affected by the failure. Link layer strategies are more flexible because they do not assume topology structures or sub-division of the network into network spans, as is the case with physical layer restoration strategy. Any links and nodes in the network are allowed to be used for restoring traffic thus there is no notion of localizing the failure to a particular subset of the network (e.g., network span). Pre-planning algorithms are used to determine the best way to route traffic during a failure. Aggressive link layer strategies [Iraschko96][Kodialam03][Xiong99] are shown to be significantly more bandwidth efficient than physical layer strategies. But restoring traffic takes a longer time as the failure needs to be notified to the appropriate network components that are not localized as in the case of physical layer. Clearly, in the link layer restoration strategy, there is a trade-off being made between bandwidth efficiency and restoration times.

The third type i.e. the network layer restoration strategies rely on the routing protocol to re-route the traffic around the failed components. On detecting a failure, the routing protocol will trigger routing updates and then it will take time for the routes to converge to a stable state. There is also a hold-off timer [Bhattacharya03] for suppressing the failure event before the routing updates are triggered. Hold-off is normally required to avoid oscillations in the routing protocol (100ms-5s). This results in substantially longer restoration times that range from seconds to minutes. To reduce restoration times, routing protocol timers can be tuned down. But this will cause higher routing message overhead leading to overloading the CPU for processing the messages and for routing computation. Network layer strategies are based on an unplanned approach. This solution reacts to the failure when it happens. The route to use during the failure depends on the order of events and the routing state of the network. Thus it is an optimistic approach that does not provide any form of guarantee for restoring full traffic load during a failure. The common approach is to over-provision the links such that utilization during normal network operation is around 50% for each link [Bhattacharya03]. However, the advantage is that network layer strategies are more robust to cover for multiple simultaneous failures and are much simpler to operate than the physical layer or link layer strategies.

Bandwidth path rerouting is mostly used at the IP layer to restore service after link and node outages. The routing convergence, in many cases, could be too long. Faster restoration mechanisms can be used in the logical network layer in ATM / MPLS / GMPLS networks. Fast restoration is achieved by using pre-planned mechanisms, i.e. using backup tunnel paths [Yahara97] [Kawamura99] [Sharma02]. An active VPN can thus be protected using a backup tunnel path as shown in figure 6.1. As soon as a fault affecting an active path is detected, the connections going through it are rapidly switched to the backup path with a minimum packet loss. The backup path is already pre-established and does not need online calculation or convergence time. For example, in figure 6.1, a fault in the physical links 4-5, and 6-3 affect the active VPN paths 1-5, and 6-2 respectively. The active VPN₁₋₅ is protected by the backup VPN₁₋₅ and the connections going through the active VPN are diverted to the backup path. In the similar manner, connections going through the active

VPN_{6,2} are diverted to the backup VPN_{6,2}. There are several backup techniques in the literature and each one has its advantages and drawbacks [Calle01] [Marzo03].

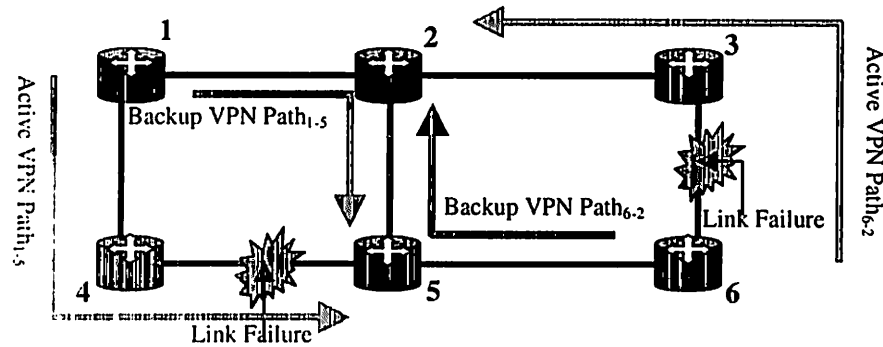


Figure 6.1 Active and Backup VPN Paths

Providing a restoration strategy for all the active VPN tunnels could be very costly in terms of bandwidth consumption. Hence, it would be appropriate to divide the logical network into several sub networks and provide survivability selectively. The VPN tunnels carrying high priority traffic or carrying special services are to be provided with backup VPN paths for the purpose of restorability. However, we could have many levels with different degrees of protection. The computation of backup paths is usually done using similar procedures to that of Active VPN path computation or Active VPN design as described in chapter 4.

6.2 Link Layer Restoration Strategies

Failure of any edge in a VPN tree would disrupt the service unless a backup path was established to reconnect the tree. A restoration algorithm selects a set of backup paths and allocates necessary bandwidth on them in advance, so that the traffic disrupted by failure of a primary edge can be re-routed via backup paths. Link layer restoration strategies are based on connection-oriented network architectures like MPLS [Awduche99][Rosen99], ATM [Hassan00], and GMPLS [Berger03],

which offer traffic-engineering mechanisms [Nadeau02][Swallo99] for better control of traffic delivery within the network.

The restoration techniques can be categorized into *link-based* [Herzberg95][Xiong99], and *path-based* [Sakauchi90][Li03]. Link restoration finds alternate paths between the nodes of the failed link. The optimal link restoration approach may involve circuitous restoration routes. Path restoration overcomes this problem by rerouting flow between the source and destination pairs affected by a link failure. This is illustrated in figure 6.2. Figure 6.2(a) shows a network consisting of 6 nodes and 8 links without any link failure. It also shows one of the possible ways for reaching at node 4 starting from node 1. Figure 6.2(b) shows one way of recovery when link 2-3 fails. This is called as link restoration where we find another path for the failed link. Another way of restoration is shown in figure 6.2(c). In this case we do not find alternate path for the failed link. We do try to find an alternate path for the original source destination pair. We have used path based restoration approach in our thesis.

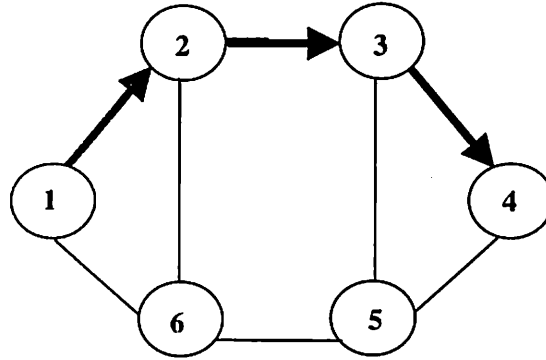
The incorporation of restoration leads to a new QoS enabled traffic-engineering problem [Kodialam03]. There are two approaches to handle this: First one is simultaneous formation of both active and backup paths, which aids in fast restoration by eliminating path computation and path setup signaling delays, and the other one is using a signaling protocol to re-establish the backup path only after active path fails.

Backup VPN paths can be computed in two ways, online and off-line planning [Lau04b]. We can expect future networks to be much more dynamic where unicast connections are setup on-demand and have relatively short-lifetime. The online backup path computation is a better approach in this type of network environment. The same strategy is also required for multicast environments. Multicast connections are expected to be established on-demand thus having prior knowledge of all the requests is not feasible for the restoration strategy. Members can join or leave the multicast group at any time, hence the restoration strategy for each multicast group needs to be recalculated after every modification. Calculating the minimum cost tree

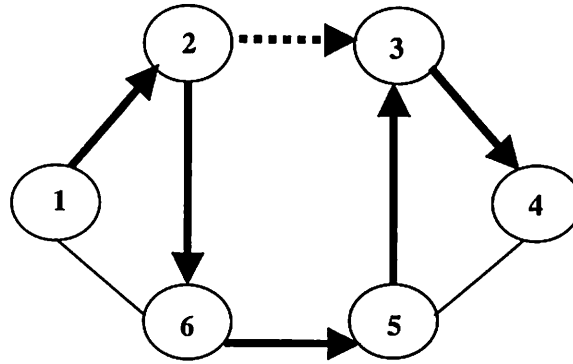
for a multicast group of nodes is a well-known NP-Complete problem [Singhal03c] thus trying to optimize for all multicast tree in a network at the same time will be too expensive. This is especially true for efficient multicast backup where multicast tree-structures are used for backup. This thesis proposes restoration algorithms for multicast VPNs as well where computation is done online. In respect to multicast restoration, we would require a complete backup tree that is disjoint from the active tree.

Centralized schemes of network planning for routing restorable tunnels typically assume that all the tunnel requests are available at the time of route computation. The problem is then solved using combinatorial optimization techniques. Though these schemes result in optimizing the amount of bandwidth consumed, the assumption that all tunnel requests are available at the time of route computation is too restrictive. Since we are interested in dynamic routing of tunnels, we cannot use offline algorithms that assume that all the restorable VPN demands that are to be routed are known *a priori*. Instead, the tunnel requests that arrive one-by-one have to be routed by an on-line algorithm that routes both the active path and the backup path. The routing should be such that it meets the service provider's traffic engineering requirement of optimizing network resource utilization so as to increase the number of potential future demands that can be routed. We assume that requests for tunnel setup arrive one at a time. Each request has a source and destination, and an associated bandwidth. The objective of the restoration algorithm is to compute an active path and a backup path for every request. This scheme of computing the backup path at the time of tunnel setup is referred to as the precomputed restoration path approach. This is different from the restoration approach where the backup path is determined at the time of failure. Such a restoration scheme suffers from two major problems. The first is that the restoration time increases significantly since the restoration path has to be computed at the point of failure. Further, since the backup bandwidth is not reserved ahead of time, there may not be sufficient bandwidth to route the backup path at the time of failure. Therefore, in our work, we assume that the active and the backup paths are computed at the time of VPN tunnel initiation and if sufficient bandwidth is not available to set up either the active path or the backup path then the tunnel setup request is rejected. For restoration to be feasible, a link or

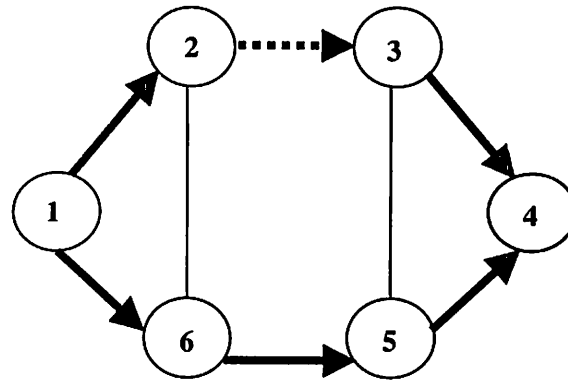
node failure should not cause both the active path and the backup path to fail and so they cannot share the same network resources.



(a) Active Path 1-2-3-4



(b) Backup Path 1-2-6-5-3-4



(c) Backup Path 1-6-5-4

Figure 6.2 Link/Path Restoration Strategies

Our work considers the problem of restoration upon link failures. Links are the network resources, which cannot be shared between the active and backup paths. We consider only link failures because the backup path is likely to be used only for a short time until a new active path is set up. Secondly, protection against multiple failures requires multiple backups and this is too expensive in backup resource requirements.

The schemes that we consider in this work are all path-based routing schemes. For path failure protection analogous to 1:1 link protection, when any single link fails in the network, the active VPN path that traverse that link should be switched to their pre-established backup paths. With 1:1 link protection, traffic is directed to the backup link by signaling after link failure. Since the backup path is used only after a failure, it can be used to carry lower priority traffic during normal operation. It is also possible to judiciously share backup paths. Sometimes, however, the delay involved in signaling the sender to start using the backup path upon failure may be unacceptable to applications. A faster restoration alternative is path protection analogous to link protection where data is simultaneously sent on both primary and backup paths. The receiver upon detection of primary path failure immediately starts using data from the backup path. Since the backup path is always used for sending data, it cannot be shared. Our primary focus in this work is path protection permitting sharing of backup paths. We use the link nonsharing case only for bandwidth-efficiency comparison purposes. Because we are protecting against single-link failure, note that two VPNs whose active paths are completely link disjoint can share backup links. If they are partially link disjoint, then they can share some backup links. For example, consider the network shown in figure 6.1. Let we need a protection from single link failure for both the active VPN paths. The backup VPN path₁₋₅ and the backup VPN path₆₋₂ both go through the physical link 2-5. We do not need to reserve additional bandwidth for both the backup paths in that link, only the one that requires more. This is because at a time only one backup path can use that link. If we reserve larger bandwidth, it can accommodate the smaller one when it fails. The objective of our work is to exploit this sharing in order to reduce the total amount of bandwidth consumed by the VPNs. The amount of sharing that can be achieved in the backup

paths is a function of the information available to the routing algorithm. This concept is called as backup path sharing. This can also be called as spare capacity management. Using this concept of sharing, we can guarantee the required bandwidth for a backup path from a reserved pool whenever it is required. However, none of the backup paths have their required bandwidth already assigned.

Mostly the Link restoration schemes are classified as centralized or distributed based on where the routing computations are done. Centralized schemes use a centralized server that has a complete view of the network. This is not always a realistic assumption in most networks. Online strategies open the way for decentralized computation model [Kodialam03] that distributes the computational load to multiple servers resident at different geographical locations. Decentralized strategies are much more complex due to many new issues such as information inconsistency (each server uses a snapshot of the network resource state) which can result in inferior decisions.

6.3 Desirable Characteristics of Joint Dynamic Bandwidth Management and Restoration of IPVPNs

An IPVPN architecture that integrates dynamic bandwidth management and path restoration must have the following characteristics:

Integration: To avoid interference, different mechanisms like dynamic bandwidth management, restoration, and spare capacity management must integrate and coordinate with each other. For example, when a reconfiguration occurs because of bandwidth management, the corresponding backup paths must be recomputed and spare capacities must be updated to reflect the new change.

Modularity: A VPN path management architecture providing bandwidth management, and survivability can be called as modular provided it can act as an independent management module and can be deployed (activated or deactivated) at the behest of network service provider.

Robustness: The whole system should not be affected if part of the VPN path management architecture goes down. Even if the entire architecture goes down, the underlying network should continue operating without the availability of these specific services.

Scalability: The growth of the network including the increase in VPN users must not degrade the performance of the VPN by overwhelming the network with additional messages generated as part of the functioning of our VPN management algorithms. However, scalability in network management systems are defined with the help of special characteristics like the system should operate continuously, with high reliability and over several time scales.

Independence: Bandwidth management and restoration algorithm actions must be independent in the sense that no other network management modules like say routing should notice the results or actions of these modules.

Simplicity: Bandwidth management and Path restoration modules should not put too much load on the network elements.

6.4 Literature Survey on Restoration Strategies in IPVPNs

6.4.1 Kar's Minimum Interference Routing

[Kar03] et al. proposed a routing algorithm that uses a critical index as the weight for using each link in the network. The critical index is related to the interference between the paths using this link. The objective is to find routes that minimize the interference with established paths in the expectation that it will lower the number of blocked requests. They have extended this work for 2-route flows where the objective is to find two disjoint paths between source and destination, with minimum interference. Their algorithm is based on fixed-route approach where all possible paths that can be used are pre-determined. Their algorithm is designed for

1+1 path protection strategy only where there is no bandwidth sharing between different backup paths. To allow backup sharing, it requires that at least the aggregate backup bandwidth used on each link is distributed to nodes performing route computations. If this information is not available, backup sharing is not possible. Their algorithm routes as many connections as possible for one at a time arrivals and no knowledge of future arrivals. They have used the concept of improved path selection in place of backup bandwidth sharing to achieve efficiency. They have demonstrated that the minimum-interference routing approach is better over other routing approaches like min-hop etc. for restorable routing.

6.4.2 Kodialam's Minimal Information & Complete Information Scenario

[Kodialam03] et al. proposed heuristics for computing bandwidth guaranteed active and backup paths allowing backup sharing. The backup link cost is modeled separately. They have employed 1:1 path protection strategy. They claimed that a straightforward solution to restoration problem like finding two disjoint paths might result in excessive resource usage. Hence, by judiciously sharing the backup bandwidth, bandwidth usage can be reduced. They proposed several variations of the heuristic algorithm that assumes different level of network information: complete, partial, and minimal. The objective was to devise an algorithm that uses only aggregate information rather than per-path information, to reduce the amount of information that must be exchanged in a distributed computation system. Complete information assumes that all link dependency information is available to the algorithm. Partial assumes only the aggregate information is available, which includes the backup bandwidth, the service bandwidth, and the link capacity. The partial information approach allows the information to be distributed easier thus more suitable for decentralized computations. Minimal only assumes the available bandwidth on each link. Results show that partial information requires significantly less total bandwidth than the minimal information based heuristic algorithm, but requires significantly more total bandwidth than the complete information based

algorithm. Also, they proposed a new integer programming solution for the joint-path computation.

6.4.3 Li's Full Information Restoration

[Li03] et al. proposed a polynomial time algorithm for 1:1 path protection strategy with backup bandwidth sharing based on separate path computation called: *Full Information Restoration (FIR)*. They also proposed a distributed approach where the network state information is kept locally in each switch/router. The computation server then retrieves the updated information from the required set of routers on-demand. This allows computation to be decentralized to multiple servers.

6.4.4 Liu's Successive Survivable Routing

[Liu01] et al. proposed a progressive online heuristic algorithm that exploit backup bandwidth sharing. They call this algorithm as *Successive Survivable Routing (SSR)*. The objective is to compute the backup path(s) that protects the active path using standard shortest hop path algorithm and then incrementally recompute all the backup paths at periodic time intervals for better solutions. Although each iteration of the recomputation process has polynomial-time complexity, the algorithm as a whole is non-polynomial and may take an exponential amount of time to converge to the local optima. The algorithm is designed to run in a distributed manner and its iterative nature pushes the network close to the local optimal state over time. However, the distributed algorithm may not converge to a stable state because of possible decisions that cause oscillations in path updates.

6.4.5 Iraschko's Algorithms

[Iraschko96] et al. proposed an Integer Linear Programming (ILP) formulation for offline restoration strategies. They compared the formulations that used path restoration and span restoration (where the network is divided into independent spans or subnetworks) and their results showed that path restoration requires significantly

lower backup bandwidth. They also showed that the use of joint path computation could reduce the backup bandwidth requirement significantly. However, they did not propose any heuristics based approximation algorithms. The idea of *stub release* was proposed where the bandwidth from the failed service path is released for use by any backup paths. Although performance is improved slightly, the practicability of stub release is questionable. Stub release requires all the affected routers to know how much bandwidth to shift from service bandwidth to backup bandwidth in any given failure scenario. It also requires the network equipment to allow shifting of bandwidth on the fly. Another drawback is that it can complicate the recovery and reversion process. During recovery, it takes time for the active paths' traffic to be rerouted to the corresponding backup paths. During the rerouting time, the bandwidth stub released may still be used by the active path and can potentially delay the time for the traffic to flow back to normal. The reversion process is the rerouting of traffic back to the service path after the failure is fixed. If the service path's bandwidth is released, there is no guarantee that traffic will revert to normal behavior until all the traffic has been flushed from the backup paths that are used during the failure.

A new concept called *path intermix* is proposed in this thesis, which is similar to stub release except that the bandwidth is not released to the general backup pool, rather, only the backup paths protecting the service path can exploit that service path's allocated bandwidth. [Herberg95] et al. proposed a restoration strategy based on online restoration where the objective is to restore connection around the failure using path segment. Path segment refers to a partial path that starts at the service path on the node preceding the failure and joins back the service path at the node succeeding the failure. However, [Iraschko96] et al. and [Xiong99] et al. observed that line restoration is inferior to path restoration because it does not distribute backup bandwidth as much as path restoration over the network.

6.4.6 Fei's Algorithm

[Fei01] et al. proposed a dual-tree structure approach for multicast resilience. Assuming a bi-connected network graph, they found a sub-graph that is also bi-connected and includes the source node and the destination nodes. This sub-graph is

used as the multicast structure that offers fault tolerance. A graph is bi-connected if there exists two disjoint paths connected between any two nodes in the graph. This is a stronger requirement than what is necessary by a multicast tree since multicast only requires disjoint paths between a source and the destination nodes only. Dual-Tree guarantees route connectivity but does not provide any bandwidth recovery guarantees as it does not do any bandwidth reservations. For the experiments, they assume the routes to restore traffic to the affected destinations being computed on the fly and the bandwidth being freely available (no capacity constraint). Another disadvantage of this scheme is that the links used in the multicast structure is not minimized as the whole structure is bi-connected.

6.4.7 Singhal's Algorithms

[Singhal03a][Singhal03b][Singhal03c] et al. proposed several new multicast restoration strategies and algorithms that are for online restoration. One of their proposed algorithms is called *Optimal Path-Pair Shared Disjoint Paths*. Optimal Path-Pair uses a strategy that establishes two disjoint paths between the source and each destination. Bandwidth sharing is allowed between the service paths, and also between the backup paths. Their simulation results show that Optimal Path-Pair requires substantially lower bandwidth, and causes significantly less number of request blocks.

6.4.8 Italiano's Approximation Algorithm

[Italiano02] et al. proposed restoration algorithms for coping with single link failure in the hose model VPN. Their approach is based on identifying a set of backup edges and reserving bandwidth on them such that service quality is ensured in case of any primary edge failure in the VPN tree. They have introduced several cost functions in the selection of backup edges and choose the one that minimizes the total bandwidth reserved in the backup edges. This restoration problem is a variant of optimal graph augmentation problem that is NP-Complete. Hence, their approach towards solving this problem is an approximation algorithm for minimum cost

restoration. The performance guarantee that their algorithm achieved is 16 times of the optimum. Their algorithm is based on designing two reductions to convert the original problem to one of adding minimum cost edges to the VPN tree so that the resulting graph is 2 connected, and hence can be easily solved in polynomial time. The two reductions introduce approximation factors of 8 and 2, respectively, hence resulting in a 16-approximation algorithm with polynomial time complexity.

6.5 Algorithms for Survivable IP Virtual Private Networks

In this section we have presented two heuristic algorithms for the online unicast restoration problem for IPVPNs discussed in section 6.2. The heuristics that is used for computing the unicast path between headquarter and branch office network is a simpler one than the one used in section 4.5. In section 4.5 BPCH is used to compute a minimal VPN tunnel path, but here in this section the two heuristics APH and BPH are used to solve the unicast restoration problem. In the next part, an example is described to understand the concept of active and backup paths. In the following section, algorithms are proposed.

Figure 6.3 shows an Intranet/Extranet VPN scenario with four branch offices connected to the corporate headquarter through VPN. This situation models a single source and multiple destinations. Every link has a cost associated with it. The cost we assume here is a function of QoS parameters like bandwidth, and loss over a link. Every node in this network is associated with a weight function that measures the capability of the node in the form of hardware and software resources available at that node. We also assume another metric called *funds* that is defined as a limit on the number of core routers that can be activated. Our objective in this work is to find efficient tunnel path spanning from source to multiple destinations. Here, by activating few core routers, the overall routing cost can be reduced. For example, in figure 6.3 the total cost incurred when no core router is activated is 23 units; $S-C1-C3-D1$ (5) plus $S-C1-C3-C4-D2$ (8) plus $S-C1-C3-C5-D3$ (5) plus $S-C1-C3-C5-D4$

(5). This is of course by traversing minimum distance. But, if we activate the core router C3, we get a cost benefit of 6 units for the same task. Activating C3 means it can act as a tunnel endpoint. So the tunnels that will be formed are S to C3, C3 to D1, C3 to D2, C3 to D3, and C3 to D4. The total cost in this arrangement is 17; $S-C1-C3$ (2) plus $C3-D1$ (3) plus $C3-C4-D2$ (6) plus $C3-C5-D3$ (3) plus $C3-C5-D4$ (3). This is because of the fact that in the later approach, we are able to save 2 units of cost each for D2, D3 and D4 as these three have a common path from source S up to C3. Activating two core routers (C3 and C5) gives a total cost of 15 units, saving again 2 units from the earlier. This is because of the provision of a tunnel between D3 and D4 as these have a common link C3 to C5 in the optimal path.

High availability of network connections is a key challenge to service providers to increase their revenue. When a path fails, the service provider must quickly re-establish another path so that the user can continue its VPN connectivity without interruption [Italiano02]. Restoration mechanisms help in this regard.

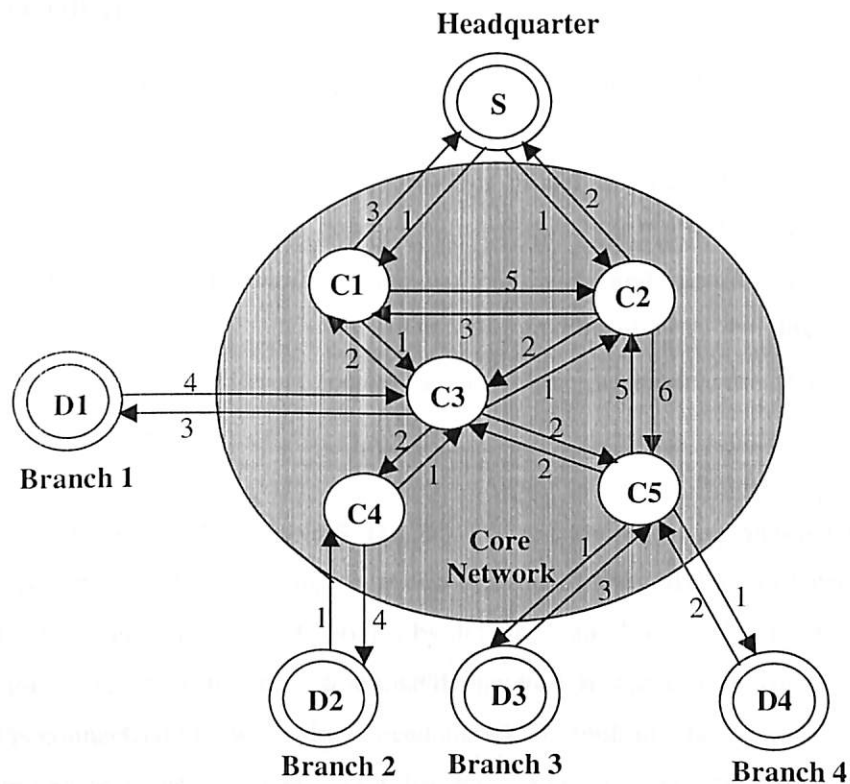


Figure 6.3 Intranet/Extranet VPN scenario

Some work has already been done on optimal tunnel path finding [Cohen00][Gabow86][Hac99]. However, the problem of computing optimal tunnel paths with restoration guarantees has not been studied extensively. Our work here is motivated by emerging trends in core networks or backbone networks towards fast provisioning of optimal restorable VPN tunnels in the case of link failures.

In a VPN tree, if any edge fails, this would definitely disrupt the service unless a backup path is established to reconnect the tree [Italiano02]. A restoration algorithm selects a backup path so that the traffic disrupted by failure of a tunnel can be rerouted via backup paths [Lau04a]. Our work in this part presents heuristic algorithms for restoring VPN path in case of link failures.

6.5.1 Minimal Cost (MC) Unicast Restorable IPVPN Problem

We have assumed a capacitated network, in which every link has a cost associated with it. The links are directed. We assume the cost over each link as function of link bandwidth and loss. Our model allows different costs on every direction in any link. Hence, the core network shown in figure 6.3 is a directed graph. More formally, the link with less residual bandwidth and more loss is expected to be more costly than the link with more residual bandwidth and less loss. As the network links have different cost on each direction, our model resembles the behavior of a real network.

In figure 6.3, we have S, D1, D2, D3, and D4 as border routers and C1 to C5 as core routers. If C3 is acting as an end of a tunnel, we call it as activated. To form a VPN between S and all destinations, by default, S and D1 to D4 must be active (VPN enabled). The objective is to minimize the network resources used for establishing the VPN connectivity between single headquarter and multiple destinations. The problem here can be viewed as obtaining a directed Steiner tree starting with the source and spanning all the branch office gateways. This is because VPN tunnels form a tree, as

we can remove a tunnel to reduce the layout cost without the loss of connectivity. As our corporate headquarter node acts as source in Intranet/Extranet VPN, we seek a minimal cost spanning tree from S to all destinations. Considering the resource scarcity, we should only activate those many core routers that do not exceed the *funds* available. This will give us the set of *Active paths*.

For the purpose of restoration, when either a single link or multiple links of a tunnel fails, we construct a backup path for the tunnel, not for the failed link. This gives us an optimal backup tree starting with S and covering all the branch offices. Next, we formulate this problem mathematically and propose new algorithms.

The problem can be mathematically formulated as: Given a directed graph G (V, E) representing a mesh network where V is the set of vertices and E is the set of edges with an edge weight function $c: E \rightarrow R^+$, a vertex weight function $w: V \rightarrow R^+$, a bound on available funds F for active core nodes, a group $M \subseteq V$ of border nodes, and a root (headquarter node) $s \in M$.

Our problem here is to find a set of directed paths both *Active* and *Backup*, and a set of active nodes A such that:

$$\begin{aligned}
 \text{Minimize:} \quad & \sum_{AP \in T} \sum_{e \in AP} c(e) + \sum_{BP \in BT} \sum_{e \in BP} c(e) \\
 \text{Such that:} \quad & \sum_{v \in A \setminus M} w(v) \leq F
 \end{aligned} \tag{1}$$

Where, for all $AP \in T$, $BP \in BT$, the endpoints in AP and BP are vertices in A, and for all $v \in M \setminus s$, there exists a sequence of one or more directed paths in T and BT that leads from s to v. Here, T represents the optimal VPN Active Tree, and BT represents the optimal VPN Backup Tree.

6.5.2 Proposed APH and BPH Algorithms for Unicast Restoration

Algorithm 8: Heuristics for Active Path (APH)

Input: A directed network graph $G(V, E)$ with edge costs and vertex weight functions, a source headquarter node s , a set of branch office nodes M , and available funds F

Output: A Steiner tree rooted at s and spanning all the nodes in M

Step 1: Initializations: $X \leftarrow M, T \leftarrow \phi$

Step 2: CPE based Solution to MC restorable VPN

While ($X \neq \phi$) do

a) Find the shortest path p from s to a node $t \in X$ using algorithm DIJKSTRA

b) $T \leftarrow T \cup \{p\}$

c) Update the cost of all the edges along p to 0

d) $X \leftarrow X - \{t\}$

Endwhile;

Step 3: $A \leftarrow M$

Step 4: Call Optimize (T, A).

Step 5: Report (T, A) as the desired solution.

We present two heuristic algorithms and a procedure for selecting the core routers. In [Cohen00], the Active Shortest Path and Active Double Tree heuristic algorithms were proposed for STP on directed graph. Our algorithm 8 is variant of ASPH. Algorithm 8 computes shortest path from S to all the destinations one by one. The core routers encountered during the tree creation (iteratively) are the intermediate nodes with shortest distance from S . So we update the cost of the links along that path as 0 for the next iteration. Hence, any new destination node that uses earlier explored nodes need not incur extra cost. This is repeated until the complete Active Tree is built. A similar approach is used in [Cohen00] except that in our algorithm there is no multiple usage of edges. Also in our case we do not recreate the optimal VPN tree for any new border router getting dynamically added to the set M . So, our approach has an edge in the sense that it takes less time to compute the paths and also ensures survivability.

Procedure 1 Optimize (T, A): N/W Based Solution*Step 1:* Compute the benefit for each core router using predecessor chains, and then the cost.

a) Depth_First (T)

If (T ≠ 0) {

- i. S ← Root of T
- ii. Depth_First (left subtree)
- iii. Pred [LST] ← S
- iv. Depth_First (right subtree)
- v. Pred [RST] ← S

b) For (every core router K ∈ A) do {

- i. Compute No_Succ using Pred [] chains
- ii. Profit = (No_Succ - 1) * $\sum_{i=S}^{j=K} Cost_{i,j}$
- iii. Insert (K, Profit) into ActiveList
- iv. Increase Size_of_ActiveList

Endfor;

Step 2: Select the core routers that are to be activated.

While (F ≠ φ) do {

If (Size_of_ActiveList == φ) then Exit;

n ← Element from ActiveList with highest profit

If (Profit (n) == 0) {

Delete 'n' from ActiveList;
Decrease Size_of_ActiveList;

Else {

If (w (n) > F) {

Delete n from ActiveList;
Decrease Size_of_ActiveList;

Else {

A ← A ∪ {n};
F ← F - w (n);
Delete n from ActiveList;
Decrease Size_of_ActiveList;

Endif;}

Endif;}

Endwhile:

Step 3: Return (T, A).

An example of steps 1 and 2 of Algorithm 8 is as described in figure 6.4. We consider the VPN in figure 6.3 as the input to this algorithm. Here, $M = \{D1, D2, D3, D4\}$. Dijkstra's shortest path algorithm is used to compute the shortest path between S and every member of M. Let the order in which the shortest paths computed be S-C1-C3-D1, S-C1-C3-C4-D2, S-C1-C3-C5-D3, and S-C1-C3-C5-D4. The VPN tree that is

created is shown in figure 6.4. Here, every edge is used once that differs from the heuristics used in [Cohen00] where edges are used for multiple times.

Algorithm 9: Heuristics for Backup Path with Link Failures (BPH)

Input: A directed network graph $G(V, E)$ with edge costs and vertex weight functions, a source headquarter node s , a set of branch office nodes M , the minimal Active Path Tree (T), and active set A from Algorithm 8, the failure scenario set FS , and Funds (F)

Output: A minimal cost directed backup VPN tree rooted at s and spanning all the nodes in M

Step 1: Initializations: $BT \leftarrow T$

Step 2: Compute backup tree iteratively for all the failure scenarios

For (each $f \in FS$) do

 While ($f \neq \emptyset$) do

- i. Select an element (t) from f
- ii. Remove $l \in t$ from graph G and from BT
- iii. Find the shortest path p from s to destination node given in t using DIJKSTRA in graph G
- iv. $BT \leftarrow BT \cup \{p\}$
- v. Update cost of all edges along p in BT to 0
- vi. $f \leftarrow f - \{t\}$

 Endwhile;

Endfor;

Step 3: Update BT such that it will be a source routed tree covering leaf nodes as destinations

Step 4: Call Optimize (BT, A).

Step 5: Report (BT, A) as the desired solution.

The procedure 1 is used to find a network-based solution to the example VPN. This selects the core routers that are to be activated keeping in mind the limit on available funds. As stated earlier the weight function defines the resource capabilities of a core router. A depth first search is carried out on the tree as shown with a dotted line in figure 6.4 to know the predecessor chains in the tree. Cost benefit for each core router is computed next. ActiveList is a list structure where each node represents a core and it's associated cost benefit. For example, C3 in figure 6.4 has three successors. The profit for C3 is twice the cost of links from S to C3. This is because of the fact that out of three paths; we pay for only one path, the cost of common links. So, $P[C3] = 2 * (1+1) = 4$. This way the profit for rest all will be $P[C4]=0$, $P[C5]=4$, $P[C1]=0$. Hence, either C3 or C5 can be selected first. The selection of these core

routers is carried out until the funds are exhausted. A core router whose profit is highest is selected iff the weight function of that node does not exceed funds. At the end of this we get an optimal VPN tree. This is shown in figure 6.5, and figure 6.6 with funds = 1, and funds = 2 respectively.

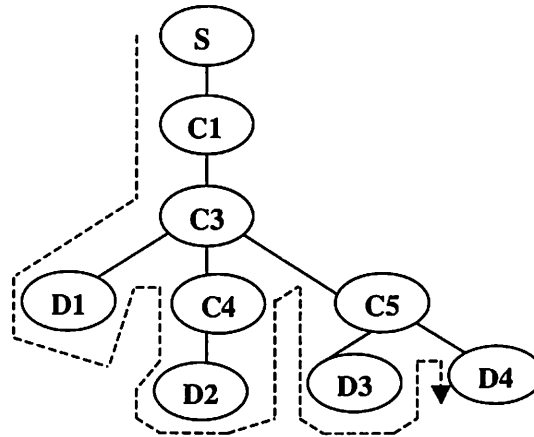


Figure 6.4 VPN Tree with source and destinations

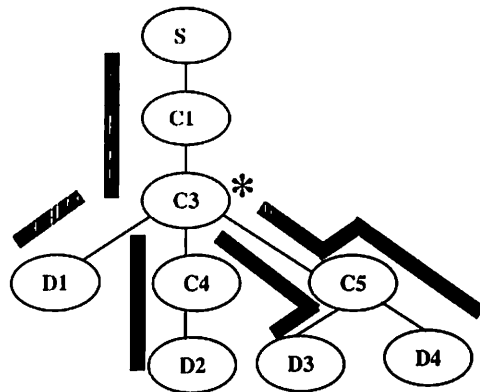


Figure 6.5 Optimal VPN Tree with Funds = 1

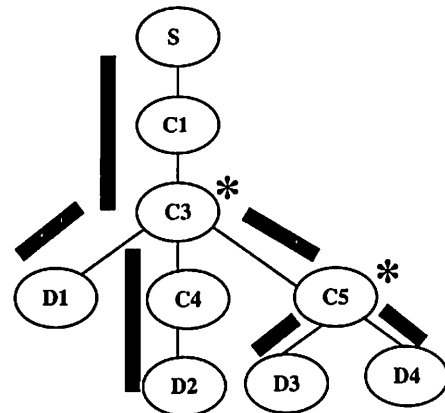


Figure 6.6 Optimal VPN Tree with Funds = 2

The set of active nodes (A) becomes [D1 to D4, C3] for funds = 1, and [D1 to D4, C3, and C5] for funds = 2. Nodes with asterisks (in figures 6.5 and 6.6) represent active core routers. These two figures show the segments or tunnels that are required to be established to carry out a communication.

Algorithm 9 is used for computing backup tree in case of link failures. Let us consider an example of few failure scenarios for the optimal active tree shown in figure 6.6. The complete failure scenario set will be $FS = f(l_{S-C1}, C3; l_{C3-C5}, C5), (l_{S-C1}, C3; l_{C1-C3}, C3), (l_{C1-C3}, C3; l_{C3-C5}, C5), (l_{S-C1}, C3), (l_{C1-C3}, C3), (l_{C3-C5}, C5), (l_{S-C1}, l_{C1-C3}, l_{C3-C5}, C3, C5)$. Here, our main objective is to compute a backup path for common segments. We also assume that the optimal backup tree should connect the single source with all the destinations. Let us again consider the backup path computation for the first failure scenario i.e. $f = (l_{S-C1}, C3; l_{C3-C5}, C5)$. Here, every element of f signifies the failed links and the tunnel endpoint in which it lies. For example first element says S-C1 is failed and it is in the tunnel path whose endpoint is C3. This is the destination node in the element. Step 1 to Step 3 with this as the input gives a backup tree as shown in figure 6.7. The updating step removes any leaf node in the tree that is not a border router. Step 4 of algorithm 2 calls the optimization procedure (Procedure 1) with active set $A = \{C3, C5\}$, and the optimal VPN tree T produced from algorithm 1. Let additional *funds* be 1 for backup tree. The set A already contains C3, and C5 inherited from active path. Here we have two core routers whose profit is computed as $P[C4]=0$, and $P[C2]=1$. So, C2 is selected as the activating core. The result is the optimal backup VPN tree for the failure scenario f . This is shown in figure 6.8 which has 7 tunnels shown as dark segments.

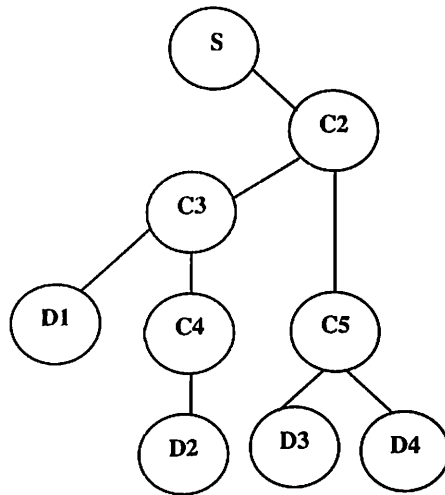


Figure 6.7 Backup Tree for Failure Scenario $f(S-C1, C3-C5)$

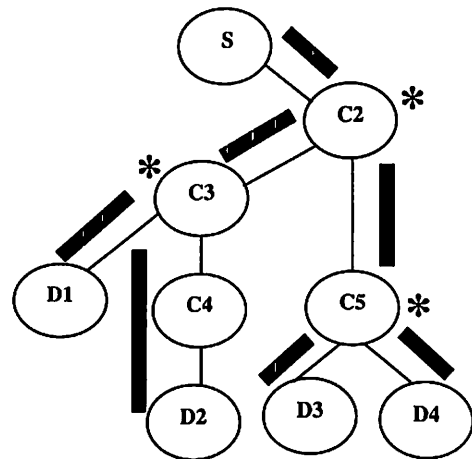


Figure 6.8 Optimal Backup Tree for Failure Scenario f with Funds = 1

The run time of APH and BPH given in algorithm 8 and algorithm 9 is analyzed below. In APH, the while loop runs for every border router, and hence a total of $|M|$ times. The Dijkstra's shortest path algorithm takes $O(E+V\log|V|)$ using an efficient heap implementation. Every other step in the while loop takes $O(E)$ times. So, our APH runs in $O(E|M| + V|M|\log|V|)$. The BPH (Algorithm 9) can be run in $O(|FS|f|(E+V\log|V|))$, where FS is the failure scenario set and f is independent failure scenario.

6.5.3 Multicast IPVPN Restoration

This part of the work defines multicast resilience problem in IPVPNs and proposes Integer Linear Programming based optimal solution and heuristics based approximate solution to this restoration problem. Multicast VPN delivers contents to multiple recipients efficiently by eliminating redundant traffic that traverses the same links in the network. This efficiency creates the basis for providing a scalable multicast service. The work in this part assumes that the multicast connections are long lived and do not have rapid changing group membership. The type of applications under multicast category running over an IPVPN could be video conferencing, multimedia broadcast etc. A multicast session requires a "point-to-multipoint" connection from a source node to multiple destination nodes, and it forms the active VPN tree, with the source node as root and destination nodes as leaves. For example, a multicast session from source node to destination nodes in figure 6.3 forms an active VPN multicast tree, with node S as root and nodes $D1$, $D2$, $D3$ and $D4$ as leaves. The cost of carrying traffic between adjacent nodes, which may be the number of hops, equipment operating cost, tunnel maintenance cost, etc., is mapped as weights on network links. The cost of a session is the sum of the weights along the links occupied by it.

The type of IPVPN we target here in this part of work is hose VPNs. Our objective is to compute restorable bandwidth guaranteed IPVPNs under hose model [Duffield99]. We expect that users will be unwilling to, or simply unable to, predict loads between pairs of endpoints. Similarly, it will become increasingly difficult to

specify QoS requirements on a point-to-point basis or pipe model VPNs, the conventional approach. The solution, called the hose model serves as both a VPN service interface i.e. the way a customer thinks of a VPN as well as a performance abstraction i.e. the way a provider thinks of a VPN. A hose offers performance guarantees from a given endpoint to the set of all other endpoints in the VPN, and for the traffic to the given endpoint from the set of all other endpoints in the VPN. The hose is the customer's interface into the network, and is the equivalent of the customer having a "link" into the network. The hose service interface allows the customer to send traffic into the network without the need to predict point-to-point loads.

A simple service model for an IPVPN is to emulate the private line or frame relay service. For all our previous algorithms described in this chapter, we used this model of IPVPN. This model would require a customer to buy a set of customer-pipes i.e. allocations of specific bandwidth on paths between source-destination pairs of endpoints of the VPN (much like virtual circuits). Figure 6.9 illustrates an example of the use of this kind of interface. The network provider would need to provision adequate bandwidth along the path of each pipe to ensure that the service level agreement is satisfied. The primary disadvantage of this approach is that it requires the customer to have precise knowledge of the traffic matrix between all the VPN sites. Resources made available to a customer-pipe cannot be allocated to other traffic. It is important to note that the network provider may not be able to take advantage of statistical multiplexing gains across the customer-pipes.

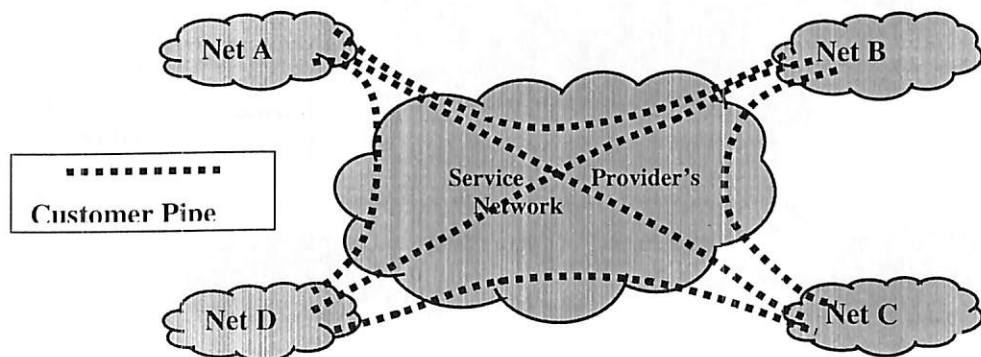


Figure 6.9 An IPVPN based on the customer-pipe model

A richer and more flexible VPN model proposed is the hose. In the hose model, a VPN customer specifies a set of endpoints to be connected with common endpoint-to-endpoint performance guarantees. The connectivity of each endpoint to the network is specified by a hose, comprising –

- The capacity required for aggregate outgoing traffic from the endpoint, into the network (to the other endpoints of the VPN).
- The capacity required for aggregate incoming traffic out of the network to the endpoint (from the other endpoints of the VPN).
- The performance guarantee for the hose, conditioned only on the aggregate traffic seen at the hose interface.

Figure 6.10 illustrates an example of the use of hoses. Say there are 4 VPN sites: A, B, C and D. The customer buys 4 hoses at each of these sites and specifies the aggregate outgoing and incoming traffic for each of these hoses. The hose specification may arrive in a variety of ways. For example, if it is known that each of the sites B, C and D sends and receives at no more than 3 Mb/s to site A, and that each of these sites sends and receives no more than 2 Mb/s in aggregate to each other, then the hose capacity would be chosen as: $A_{in} = A_{out} = 9$ Mb/s and $B_{in} = B_{out} = C_{in} = C_{out} = D_{in} = D_{out} = 5$ Mb/s. Figure 6.10 depicts one possible realization of the connectivity from hose A by means of a set of provider pipes. Different implementation options are considered in the following sections.

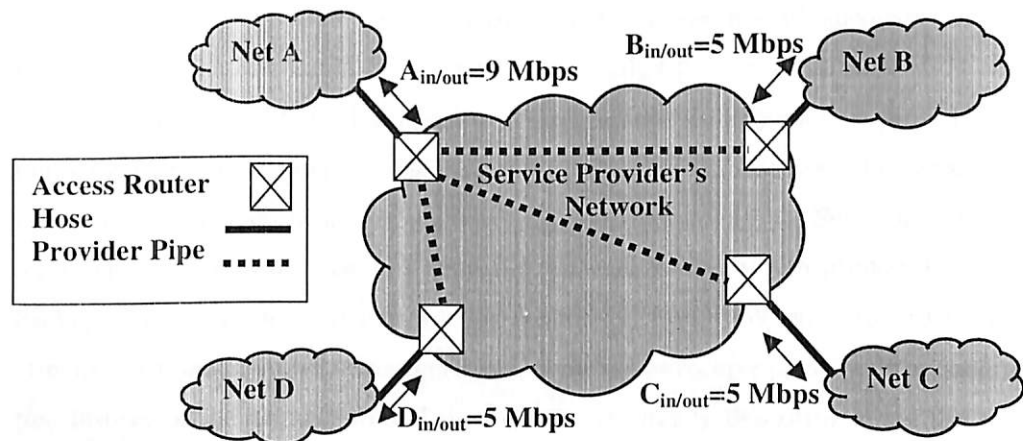


Figure 6.10 A VPN based on the hose model.

In this hose VPN model, each VPN has a specified ingress point and a set of egresses. There is no detail knowledge of the VPN's ingress-egress pair-wise bandwidth requirements. Instead, only the total bandwidth offered by the VPN to the ingress is known. This traffic may split in an unknown manner to the different destinations. However, the service guarantee to the VPN in the hose model assures that all traffic will be carried by the network. To provision this service both with the active and backup path (for the sake of survivability), a bandwidth guaranteed multicast VPN tree from the ingress to the set of egresses can be set-up with the bandwidth guarantee being equal to the total offered traffic at the ingress for both active and backups. In the worst case, this could give rise to a scenario where all the traffic from the ingress may go to one of the egresses (the actual egress being not known). However, in this application, although a multicast VPN tree with active and backup paths is computed for resource allocation and restoration, the actual data transfer takes place as unicasts. Off-line multicast restoration algorithms cannot be used since they require a priori knowledge of all multicast requests that are to be routed. Hence, as we want to allow dynamic set up of IPVPN services in the hose model without rerouting previously setup multicasts, we need here an on-line multicast restoration algorithm. The objective is to compute multicast active and backup routes so as to permit as many future demands to be routed as possible.

Link failures in an IPVPN occur often enough to cause service disruption, and they may lead to significant information loss in the absence of adequate backup mechanisms. The loss could be more when the failed link in a multicast VPN tree carries traffic for multiple destinations. A straightforward approach to protecting a multicast tree is to compute a *link-disjoint* [Singhal03b] backup tree. Two trees are said to be *link disjoint* if their edges don't include a common link. Such link-disjoint trees can be used to provide 1+1 dedicated protection where both primary tree and backup tree carry identical data to the destination nodes. When a link fails, the affected destination nodes reconfigure their switches to receive data from the backup tree instead of the primary tree. This approach is already described in section 6.2. Demerits of this approach include excessive use of resources and at times insufficient availability of network resources to establish link-disjoint trees in a mesh network,

leading to the blocking of a large number of multicast sessions [Singhal03a]. Hence, this link disjoint restoration approach is not the most efficient solution for protecting a multicast IPVPN session, and a connection may be blocked due to resource unavailability. In this thesis we proposed a segment based protection algorithm that can provide an efficient active and backup path multicast restorable IPVPN. We also compared our results with other available approaches in the literature [Singhal03a] in the next chapter. We next describe the segment protection approach with the help of an example.

In a segment-protection scheme, each segment of the active multicast VPN tree is protected by computing a backup segment disjoint to its corresponding primary segment. We define a *segment* as sequence of arcs from the source or from any splitting point (on the multicast tree) to a leaf node or to a downstream splitting point. A destination node is always considered as a *segment end-node* because it is either a leaf node in the tree or is a splitting point where a portion of a signal is dropped locally and the remainder continues downstream. Let us consider figure 6.3 as a hose VPN that has ingress-point as S and egress-points as D1, D2, D3 and D4. Hence, the objective here in this part of the work is to provide a survivable VPN that can tolerate the link failures if any and hence will not disturb any of the ongoing multicast communications in which that link might have been used. For example in figure 6.11, the active multicast VPN tree from S to D1, D2, D3 and D4 occupies arcs S→C1, C1→C3, C3→D1, C3→C4, C4→D2, C3→C5, C5→D3 and C5→D4. Here, node C3 and C5 are two splitting points as they have multiple outgoing multicast links. Hence, once the active multicast VPN tree is identified, the splitting points in the tree determine segments on the multicast tree. Hence we have six segments here. These are S→C3, C3→D1, C3→D2, C3→C5, C5→D3 and C5→D4. Out of all these primary or active segments, one is critical segment. Critical segment is that which is used as segment to carry data to all the egress points in the hose model. The notion of criticality is used because if we do not get a backup segment for this critical primary segment, then none of the egress points can be connected with the ingress point in the hose VPN. However, for any non-critical primary segment, if we do not get any backup segment, the corresponding egress points will be affected whereas the rest

part of the multicast tree will work correctly. So, in this example we have critical primary segment (CPS) as $S \rightarrow C1 \rightarrow C3$ and rest all are non-critical primary segments. In this example, we have two segments $S \rightarrow C3$ and $C3 \rightarrow C5$ for which the segment disjoint sub paths $\langle S \rightarrow C2 \rightarrow C3 \rangle$ and $\langle C3 \rightarrow C2 \rightarrow C5 \rangle$ could be computed as backup paths. These backup segments are also minimal paths. It is interesting to observe that backup segments might share arcs with already found primary or backup paths. For example, the link $C3-C2$ is being shared by two backup segments ($S \rightarrow C2 \rightarrow C3$, and $C3 \rightarrow C2 \rightarrow C5$), although the sharing is in different directions (arc disjoint), but they are sharing a single link. Consider the same example with a minor modification for better explanation to this sharing concept. Assume that we have an arc $C4 \rightarrow C5$ in figure 6.3. Let us again assume that the backup segment for primary segment $C3 \rightarrow C5$ comes to be $C3 \rightarrow C4 \rightarrow C5$. Now we can realize the fact that the same arc $C3 \rightarrow C4$ is being used to carry both active traffic and backup traffic for the segments $C3 \rightarrow D2$ and $C3 \rightarrow C5$ respectively. So, here backup and primary segments both share a link in the same direction. Another important concept in sharing is to find same link in the same direction being used as part of two or more backup segments. This could be a case where same link being used as part of two or more backup segments and primary segments. In both the scenarios, we can reserve only the maximum bandwidth required over that link to facilitate the optimum resource usage.

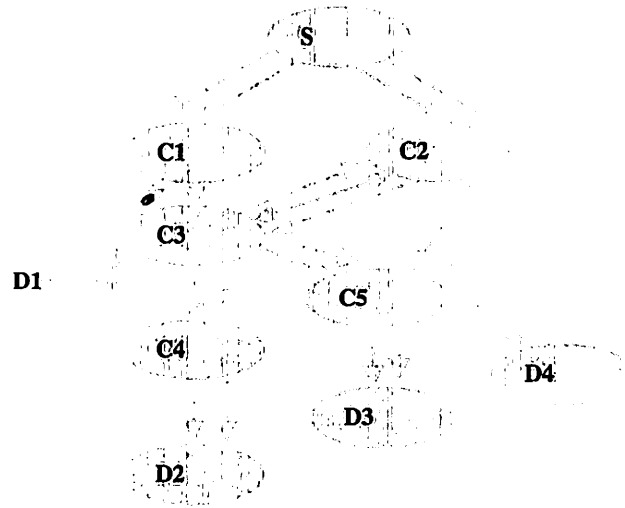


Figure 6.11 Active and Backup Multicast VPN tree (Dotted Lines show edges of Active and dashed lines show edges of backup VPN)

The problem of setting up a multicast VPN session in hose model with protection against any link failure at optimal cost is NP-complete because the subproblem of setting up a minimum-cost tree from ingress node to egress nodes (without protection), known as *minimum-cost Steiner tree*, has been shown to be an NP-complete problem [Hakimi71]. Hence, we develop efficient heuristics to protect multicast VPN connections against link failures.

Here, our objective is to find a minimal cost set of tunnels for both active and backup paths forming a logical tree routed at source node and spanning other destination or border routers keeping the number of active nodes within a bound. Also we propose to protect each multicast connection from any single segment failure. Connections are from ingress to egress only. In hose model VPN, every ingress to egress path has a different bandwidth requirement. We can assume here that every egress or destination is participating in a multicast communication with the single source or ingress like a videoconference.

We formulate the above problem mathematically as an *Integer Linear Program (ILP)* as below:

Given a directed graph $G(V, E)$ representing a mesh network where V is the set of vertices and E is the set of edges with an edge weight function $c: E \rightarrow \mathbb{R}^+$, a vertex weight function $w: V \rightarrow \mathbb{R}^+$, available bandwidth over each edge $b: E \rightarrow \mathbb{R}^+$, a bound on available funds F for active core nodes, a group $M \subseteq V$ of egress or border nodes, and a root or an ingress (headquarter node) $s \in M$.

With these notations our task is to find an active path and a backup path for every $(s, v \in M)$ such that the bandwidth required is guaranteed and with possible backup path sharing.

Let $E_1, E_2,$ and E_b be the subsets of E . So, we have to find two paths AP and BP for every source destination pair such that $AP \subseteq E_1,$ and $BP \subseteq E_2,$ and $AP \cap BP \cap E_b = \emptyset$ i.e. the edges that are used in both the paths must not be in E_b . When

$E_b = E$. the model will find disjoint paths. We have to find a set of directed paths T and a set of active nodes A for both active and backup paths.

The ILP formulation is as given below:

Let vector X represent flow on active path and Y represent flow on backup path. C_{ij} is the cost of the link (i,j) . The variable X_{ij} is set to 1 if there is a flow in the link (i, j) for the active path, and the variable Y_{ij} is also set to 1 if there is a flow in the link (i, j) used in the backup path otherwise set to 0. The variable R , and L denote nodes in the active and backup paths respectively. The variables P and Q are for deciding if the link between i , and j are used in the session for both active and backup paths. The objective function is given in (1).

Minimize

$$\sum_{(i,j) \in E1} b_{ij} * c_{ij} * P_{ij} + \sum_{(i,j) \in E2} b_{ij} * c_{ij} * Q_{ij} \quad (1)$$

Subjected to:

$$X_{ij} \geq 0 \quad \forall ij \in E1 \quad (2) \quad Y_{ij} \geq 0 \quad \forall ij \in E2 \quad (3)$$

$$\sum_j X_{js} = M \quad (4) \quad \sum_j Y_{js} = 0 \quad (5)$$

$$\sum_j X_{ij} - \sum_j X_{ji} = 0 \quad \forall i \neq s,d \quad (6) \quad X_{ij} \leq M \quad (7)$$

$$\forall j = d: \sum_i X_{ij} = \sum_i X_{ij} + 1 \quad (8) \quad \forall j \neq s: \sum_i P_{ij} = R_j \quad (9)$$

$$\forall i \neq d: \sum_j P_{ij} \geq R_i \quad (10) \quad P_{ij} \leq 1 \quad (11)$$

$$\forall i: \sum_j P_{ij} \leq D_p(i) * R_i \quad (12) \quad \sum_j Y_{sj} = M \quad (13)$$

$$\sum_j Y_{js} = 0 \quad (14) \quad \sum_j Y_{ij} - \sum_j Y_{ji} = 0 \quad \forall i \neq s,d \quad (15)$$

$$Y_{ij} \leq M \quad (16) \quad \forall j = d: \sum_i Y_{ij} = \sum_i Y_{ij} + 1 \quad (17)$$

$$\forall j \neq s: \sum_i Q_{ij} = L_j \quad (18) \quad \forall i \neq d: \sum_j Q_{ij} \geq L_i \quad (19)$$

$$Q_{ij} \leq 1 \quad (20) \quad \forall i: \sum_j Q_{ij} \leq D_p(i) * L_i \quad (21)$$

$$R_s \cap R_d = 1 \quad \forall d \in M \quad (22) \quad L_s \cap L_d = 1 \quad \forall d \in M \quad (23)$$

$$P_{ij} + Q_{ij} \leq 1 \quad \forall (i,j) \in E_b \quad (24) \quad \sum_{v \in AM} P_{ij} * w(v) + \sum_{v \in AM} Q_{ij} * w(v) \leq F \quad (25)$$

The above equations can be interpreted as below. Equation 1 is the objective function to be solved for finding out restorable bandwidth guaranteed tunnel paths between ingress and multiple egress points of multicast VPN. The backup paths that

are created are all segment disjoint. Equations 2 and 3 define the variables for existence or non-existence of links in both the paths i.e. either 0 or 1. Equation 4 defines that the source node has an outgoing flow of M units (number of destinations) in the primary path. Equation 5 defines that the source node has no incoming flow. Equation 6 defines flow balance for an intermediate node, where the flow into it and the flow out of it are equal. Equation 7 defines flow on any link for a multicast session is limited by the number of destinations in the session. Equation 8 defines that the total outgoing flow of the session is one less than the incoming flow for the destination nodes. Equation 9 says that every node that belongs to active path in the multicast session has at least an incoming edge. Equation 10 says every node except the destination nodes has atleast one outgoing edge. Equation 11 says that in an active path, any link can hold only one tunnel. Equation 12 defines that every node with at least one outgoing edge belongs to the tree, where $D_p(i)$ is the physical degree of node i . Equations 13 to 21 are for backup path with the same interpretation as that of active path. Equations 22 and 23 define the mandatory existence of source and all destination nodes in primary as well as backup paths. Equation 24 guarantees that the primary and backup paths are disjoint. Equation 25 guarantees that the total number of active nodes in both active and backup path is within the given bound.

The backup of an active path can be shared by other backups. We particularly restrict the sharing of backup paths by only assuming that no two active paths will fail simultaneously which have same backup segment. To allow sharing of backup bandwidth, we find out the maximum bandwidth usage and minimum bandwidth usage among all the active paths that have same backup segment. So, the backup link has an upper bound, say ' μ ', and a lower bound say ' δ ' representing these values. If the backup segment supports the maximum bound i.e. the available bandwidth over the backup link is more than μ , then it can support any active path which maximizes the sharing. If the available bandwidth is below δ , then it cannot support any active path and hence cost is set to ∞ . If it is between the lower and upper bound we can use that much available i.e. only that much can be shared among all. So, now any additional protocol can be used to signal the extra amount of bandwidth required. This is modeled by equation 26.

$$C_{ij} = \left\{ \begin{array}{l} \infty \cdot b_{ij} < \delta \\ \mu \times c_{ij} \cdot b_{ij} > \mu \\ b_{ij} \times c_{ij} \cdot \delta < b_{ij} < \mu \end{array} \right\} \quad (26)$$

Minimal Cost multicast VPN is NP-hard as it is similar to directed STP when we make the funds available for active nodes equivalent to infinite, and hence it is hard to approximate. So we also propose here our heuristic algorithm to solve the problem.

Algorithm 10: Bandwidth Guaranteed Multicast primary or active VPN tree creation.

INPUT: Graph $G(V, E)$, ingress (s), a set of egresses (M), bandwidth required for every pair, and Funds (F).

OUTPUT: A minimal primary VPN tree, and active core nodes (A), and all the Active Paths.

Step 1: Initialization: $X \leftarrow \{s\}$, $T \leftarrow \phi$, $M \leftarrow$ set of egress nodes.

Step 2: Using SPH build up a VPN tree to connect ingress with egresses **minimally**.

While ($M \neq \phi$) **do** {

- a) Find the shortest path 'p' connecting a vertex $x \in X$ to a vertex $v \in M \setminus X$ guaranteeing b_k amount of bandwidth.
- b) $T \leftarrow T \cup \{p\}$
- c) $X \leftarrow X \cup \{v\}$
- d) $M \leftarrow M - \{v\}$

Step 3: Identify the active core routers from the tree computed in the previous step.

- a) Find the **out-degree** of every core router
- b) Drop the nodes with out-degree equal to 0 or 1, as they are not the contenders for selection
- c) Sort the remaining nodes in descending order and store them in a list L .
- d) $A \leftarrow \phi$
- e) **While** ($F \neq \phi$) or ($L \neq \phi$) **do** {
 - i) Select the left most element of L , say 't'.
 - ii) $L \leftarrow L - \{t\}$
 - iii) $A \leftarrow A \cup \{t\}$
 - iv) $F \leftarrow F - 1$

Step 4: Identify the Primary Segments (PS) using the given network graph (G) and the active core nodes (A) identified in the previous step.

a) $M \leftarrow$ set of border nodes, $X \leftarrow \{s\}$, $PS \leftarrow \phi$

b) **While** ($A \neq \phi$) **do** {

- i) Compute the shortest path 'p' connecting a vertex $x \in X$ to a vertex $v \in A \setminus X$
- ii) Remove v from A
- iii) Insert v into X
- iv) $PS \leftarrow PS \cup \{p\}$

c) **For every egress node do** {

- i) Compute the shortest possible path 'p' from the nearest element of X
- ii) $PS \leftarrow PS \cup \{p\}$

Step 5: Build the minimal active VPN multicast tree using these Primary Segments (PS).

Step 6: For every destination node in the multicast session compute **Active Paths** from the ingress by traversing the tree produced in step 5.

We have considered the example network given in figure 6.3 earlier. Also, let the funds available be equal to 2. An application of Algorithm 10 on figure 6.3 is given below. Figure 6.12 shows the VPN tree after step 2 of our algorithm. First, we traverse to the shortest possible neighbor and then we aim towards any of the destinations that can be reached from that point onward by again traversing shortest possible path. This process we repeat till all the destination nodes are reached. Step 3 results in C3 and C5 as the core routers with highest out degree as 3 and 2. The intuitive explanation for this step is we are choosing the node that has high degree of connectivity to maximum number of border nodes possible. As we have assumed initially that we have funds available is 2, so we can activate maximum 2 core routers. In other words we can use two multicast enabled core routers for the purpose of gaining optimal path between every pair of single source and destinations. Hence, these two are chosen as the activating core routers. In step 4 we identify the segments. Step 5 is depicted by figure 6.13 that shows the MC VPN tree with the primary segments identified as $(S \rightarrow C1 \rightarrow C3, C3 \rightarrow D1, C3 \rightarrow C4 \rightarrow D2, C3 \rightarrow C5, C5 \rightarrow D3, \text{ and } C5 \rightarrow D4)$. These segments are all available in the set PS, primary segments set. These tunnels or virtual links are going to give us optimal cost path from source to all destinations in the multicast VPN environment.

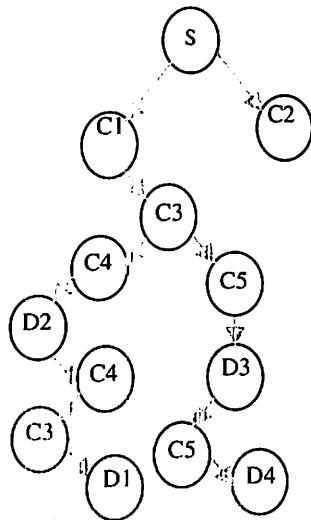


Figure 6.12 VPN Multicast tree after step 2

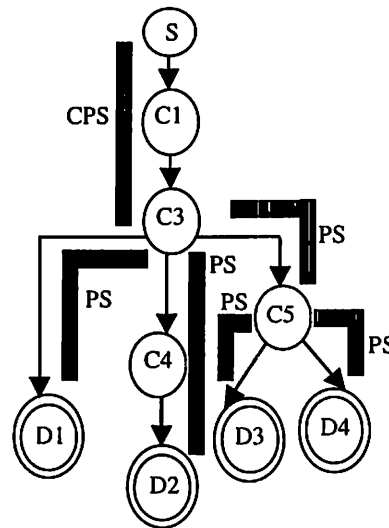


Figure 6.13 Multicast active tree with CPS and PS

We now describe our restoration strategy i.e. backup path computation for all the primary segments identified by algorithm 10. As we are dealing with tunnels for multicast sessions, we believe in segment protection rather than arc, link or path protection mechanisms as these could be inefficient as described in literature [Singhal03b][Kodialam03][Lau04a].

Step 2 of the algorithm 11 gives S-C1-C3 as the critical primary segment as it is part of every active path from S to all destinations. The $SAP_{S-C1-C3}$ will contain all active paths. SAP_{C3-D1} contains AP_{S-D1} . SAP_{C3-D2} will contain AP_{S-D2} . SAP_{C3-C5} will contain two active paths as AP_{S-D3} , and AP_{S-D4} . SAP_{C5-D3} contains AP_{S-D3} , and SAP_{C5-D4} contains AP_{S-D4} . The critical and non-critical primary segments are all shown in figure 6.13. Here, there is only one critical segment and rests are non-critical segments.

For critical primary segment, there are three failure scenarios. These are S-C1 fails only, C1-C3 fails only, or both links fail simultaneously. When S-C1 fails, we remove the link S-C1 and recompute the network link cost using equation 26. This is for allowing sharing of backup paths. The backup paths are as shown in figures 6.14-6.16.

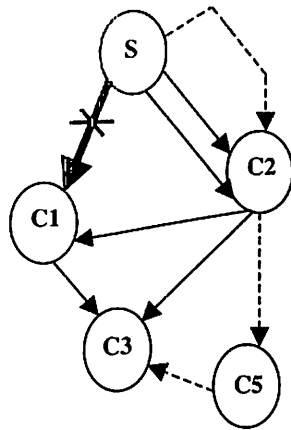


Figure 6.14 Backups for FS1 of CPS

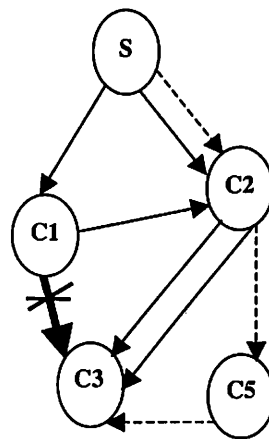


Figure 6.15 Backups for FS2 of CPS

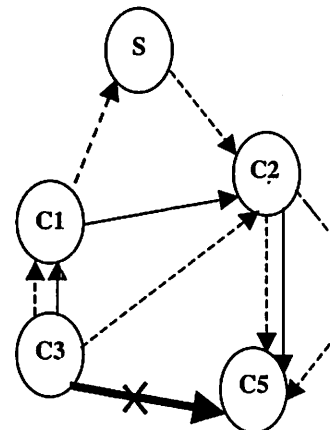


Figure 6.16 Backups for FS of PS

Algorithm 11: Backup segments for Critical and Non Critical Primary Segments.

Input: Network Graph: $G (V, E)$, Primary Segment Set (PS), and the Active path set (AP) for all the destinations identified by algorithm 10.

Output: Back up segments for Every Primary segment

Step 1: Initializations: $SAP_s \leftarrow \phi$, $BP \leftarrow \phi$

Step 2: Compute the set of APs that have a common Primary segment s if any, where $s \in PS$

For every $s \in PS$ do {

Until all the active paths (AP) are examined do {

 1. Select a path 'p' from the set AP; 2. If $s \in p$ then $SAP_s \leftarrow SAP_s \cup p$ }

Endfor;

Step 3: Compute the backup path allowing sharing of bandwidth among the backups for both critical and non-critical Primary Segment (PS)

For every $s \in PS$ do

 If SAP_s contains all active paths then go to step 4

 Else

 If SAP_s contains only one element then there is no backup segment

 Else go to step 5;

End for;

Step 4: Compute all the backup segments and choose the best one amongst all.

a. Set backup segment $BS = \phi$

b. Compute the set of failures FS_p affecting the Critical segment

c. For each $s \in FS_p$ do {

 i) Find the set of failed links E_s in the current failure scenario

 ii) Remove all the failed links $l \in E_s$ from G

 iii) Update the network link costs of every link using equation 26

 iv) Compute all possible backup segments between upstream and downstream nodes

 a) $Min[s] \leftarrow \infty$, $C[s] \leftarrow 0$, $BS_ALL \leftarrow \phi$

 b) For every vertex $v \in V$ do

 Flag [v]=false; Endfor;

 c) Push 's' onto the stack;

 d) While stack $\neq \phi$ do {

 i. $u \leftarrow \text{pop stack}$;

 ii. If (Flag [u]==false)

 1. Flag [u]=true;

 2. If (u==d) then

$BS_ALL \leftarrow BS_ALL \cup \text{path}$;

 Else

 If u has no more adjacent nodes then skip;

 Else

 For each 't' \in Adj [u] do

 If (Flag [t] ==false)

 { Push t; $C[t] \leftarrow Cost_{t,u} + C[u]$;

 If ($b_{t,u} < Min[u]$) then

$Min [t] \leftarrow b_{t,u}$;

 Else

$Min [t] \leftarrow Min [u]$;

 End if; P [t] \leftarrow u; } Endif;

 End for; } Endif; Endif;

End while;

v) Identify the element from BS_ALL that has highest available bandwidth. If two or more paths have same bandwidth then choose the one with less cost as the backup.

a) Max=0, Min=0;

b) While BS_ALL $\neq \phi$ do {

i. Select one element 'p' from BS_ALL

ii. If (Bandwidth (p) > Max) then

{ 1.BS \leftarrow p; 2.Max \leftarrow Bandwidth (p); 3.Min \leftarrow Cost (p); }

Else

If (Bandwidth (p) ==Max) then

If (Cost (p) < min) then

{ BS \leftarrow p; Min \leftarrow Cost (p); }

End if;

End if;

End if; }

End while;

vi) BP \leftarrow BP \cup BS;

vii) Update cost=0 for already found backup segment

viii) Insert failed links back into G }

End for;

Step 5: Compute the shortest backup segment for all the non-critical Primary Segments

a) Set BS = 0;

b) Compute δ from all the paths in SAP,

c) Compute the set of failures FS_p affecting the Critical segment

d) For each s \in FS_p do {

a) Find the set of failed links E_s in the current failure scenario

b) Remove all the failed links l \in E_s from G

c) Update the network link costs of every link using equation 26

d) Compute the shortest possible backup segment between upstream and downstream nodes with a bandwidth guarantee of minimum δ units

1) For each node v \in V do {

a) d [v] $\leftarrow \infty$, b) w [v] \leftarrow nil } Endfor;

2) d [upstream] \leftarrow 0

3) Q \leftarrow V

4) While Q $\neq \phi$ do {

a) u \leftarrow Extract-Min (Q)

b) If (u ==downstream) then

{ BS \leftarrow BS \cup u; Return Path i.e. BS; }

c) For each v adjacent to u do {

If (b_{v,u} < δ) then skip

Else

If (d [v] > d [u] +cost (v, u))

{ d [v] = d [u] +cost (v, u); w [v]=u; }

End if;

End if; }

End for; }

End while;

e) BP \leftarrow BP \cup BS

f) Update cost = 0 for already found backup segment

g) Insert failed links back into G }

End for;

Figures 6.14 and 6.15 show the backup segments for the failure scenarios of critical primary segment. In figure 6.14, we have three backup segments shown with different arrows. For every segment, our algorithm computes the maximum bandwidth available and the cost of the segment. Then it chooses the segment as backup segment that has least cost and maximum available bandwidth. We claim here the sharing of backup segment bandwidth in a way that the bandwidth reserved over the backup is maximum and it can support any of the primary segments. Similar explanations apply to figure 6.15 where the link C1-C3 is failing. Figure 6.16 is for non-critical primary segment, we compute here shortest backup segment along all available that can guarantee at least the minimum bandwidth requirement amongst all the active paths for which this is the backup segment. In our example, we have only one failure scenario i.e. link C3-C5 fails. We have used modified DIJKSTRA for this part, where if any link is not supporting minimum bandwidth, we are pruning that link from the path.

6.5.4 Simulation Results of Unicast Restoration

The simulation run of our algorithms presented in section 6.5.2 was made on a metropolitan network topology that represents a big network as shown in figure 6.17. Antisymmetric edges were assigned equal costs. We ran our algorithms with varied border group sizes. These groups were reselected at random. The available funds were also varied for different cases. The results are plotted in figures 6.18-6.21.

We compared our results with the well-known SPH, which is an approximation algorithm for STP. SPH does not find a solution to minimal cost unicast restorable IPVPN problem. It constructs a spanning tree, and not an optimal active and backup path.

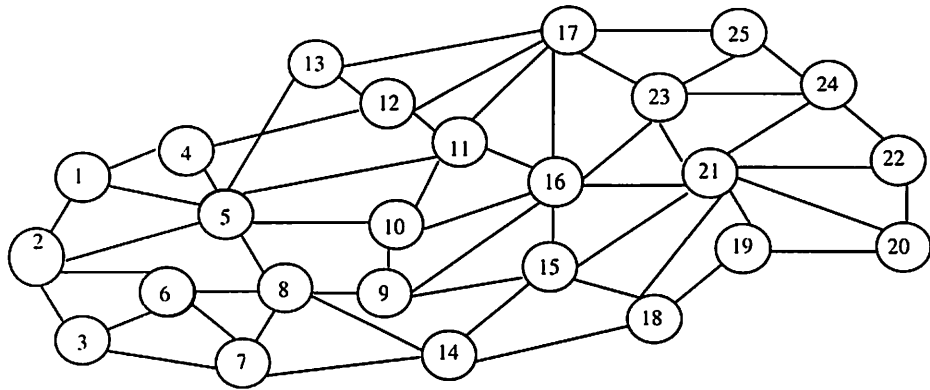


Figure 6.17 A Metropolitan Network

Figure 6.18 shows the plot between varying number of border routers and VPN cost for both Active and Backup paths when funds are equal to 2. Both the costs increase linearly with the increase in number of border nodes. BPH with core routers i.e. the optimal backup path is better over the rest three. But, it is not similar to SPH as we have fewer funds. The worst-case performance is given by APH without optimization. As BPH uses the active set (A) produced by APH, it further optimizes the tree; hence it gives a better-reduced VPN cost than APH. Figure 6.19 shows the same plot with funds equal to 13, a relatively higher value. In this scenario, both the APH with core and BPH with core are closer to SPH. This is possible because more destinations have common segments in this case. Figure 6.20 shows VPN cost for varying funds. We keep here the number of destination border nodes equal to 15. For low funds the cost of active path with core is higher than the backup. Increasing funds brings the backup path cost nearer to SPH, but the decrease in Active path cost is marginal. This is because of more core routers getting activated in case of BPH. In figure 6.21, for small group sizes up to 9, both APH and BPH use same number of core routers. For group size 11 to 26, the average number of core routers increased linearly. For above 26, it remains flat. This behavior results from the fact that when less number of destinations are to be reached from source, few common segments will be traversed in the path, and when the group size is large, the common segments will remain fixed. Also BPH activates more core routers than APH. This is because BPH creates few new tunnels than that of APH.

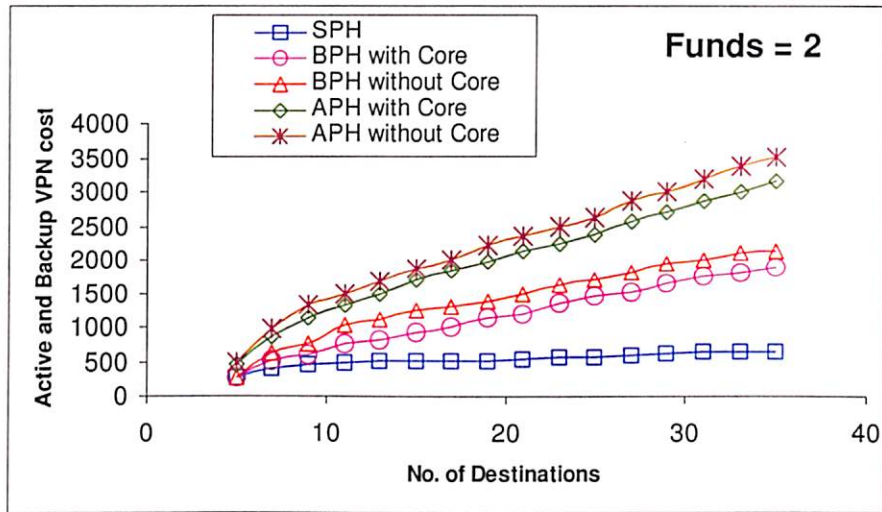


Figure 6.18 Active and Backup Path Cost with Funds = 2

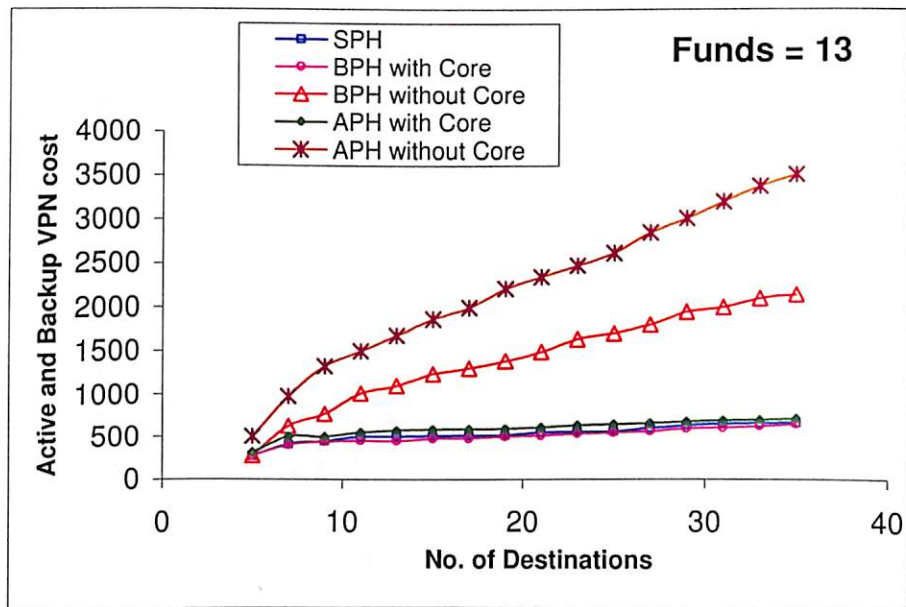


Figure 6.19 Active and Backup Path cost with Funds =13

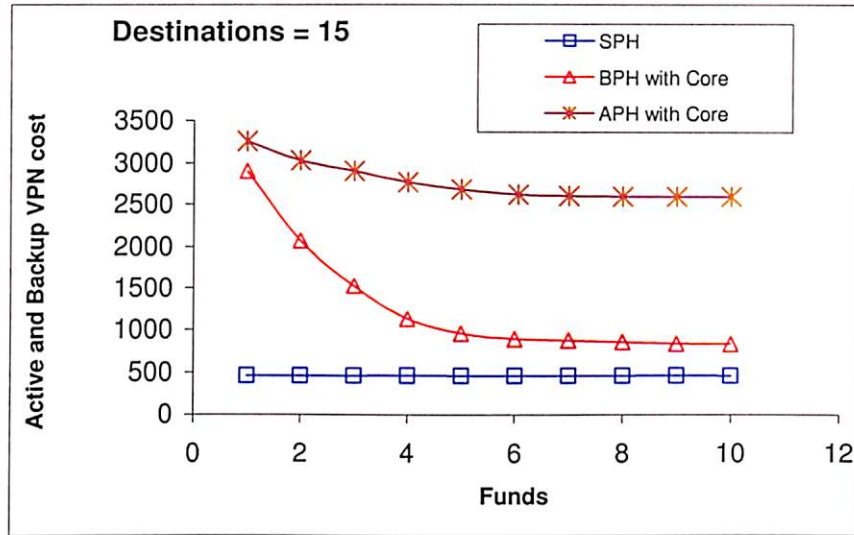


Figure 6.20 Active and Backup Path Cost with varied funds for group size = 15

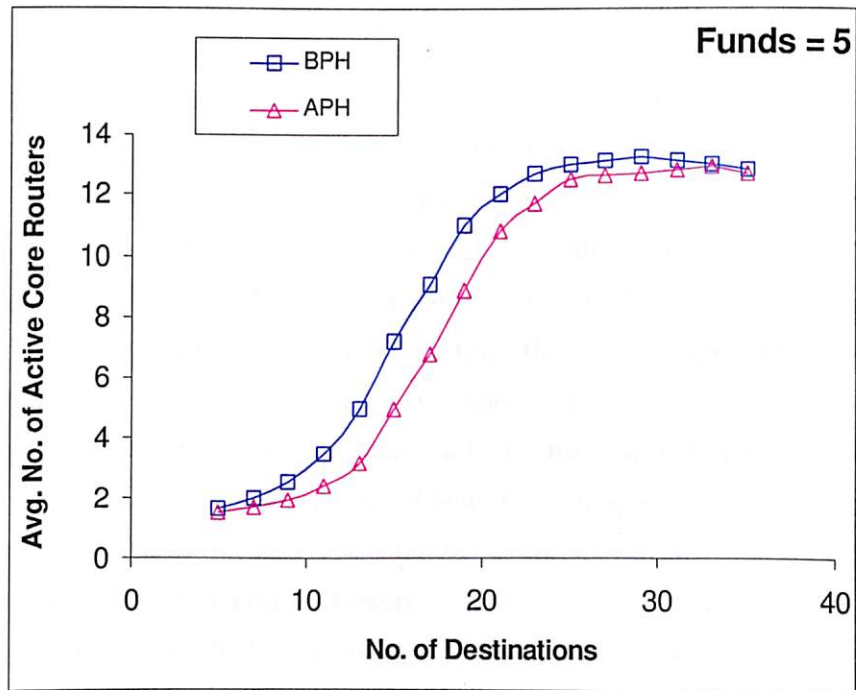


Figure 6.21 Average number of Participating Core Routers with Group Size

6.5.5 Simulation Results of Multicast Restoration

The simulation run of our algorithms described in section 6.5.3 was made on two different network topologies, a metropolitan network (figure 6.17) and a random parse network (figure 6.22). Each link was assigned some random bandwidth. Antisymmetric edges were assigned equal weights. On every network, we ran our algorithms with varied group sizes and single source.

These groups were reselected at random for every network. Funds available were varied for different cases. The results were averaged over both the networks and plotted as in figures 6.23-6.25.

Algorithms given in section 6.5.3 are compared with well-known SPH [Takahashi80], which is a well-known approximation algorithm for STP. It is used as a benchmark for calculating an optimal solution for multicast VPNs, as it is an NP-hard problem. We also used CPLEX optimization tool to solve the multicast VPN problem as given in section 6.5.3. Increasing number of nodes in the multicast session, with no funds, SPH is better than our algorithms in computing the VPN cost. But as we increase funds, we found our approach more nearer to SPH in multicast VPN scenario. Figure 6.23 shows varying multicast group size with average VPN cost. As we go on increasing this size we find increase in cost as well. But the increase is rapid for low funds. When funds are equal to null, we get worst cost. As we increase it, it approaches the best solution (SPH). Figure 6.24 shows the same for multiple link failures (a different failure scenario). Here the average VPN cost increases rapidly and we do not even get any response nearer to the optimal one (SPH). This is because when multiple links fail, we have very less number of alternate paths that are available as backups. Figure 6.25 compares both active and backup path costs for varied funds. For low funds, cost of active paths is lower than backups, but for higher funds, cost of backups is lower than active. This is possible because of backup bandwidth sharing among backup paths. As we increase funds sharing will be more i.e. more active paths will share a primary segment.

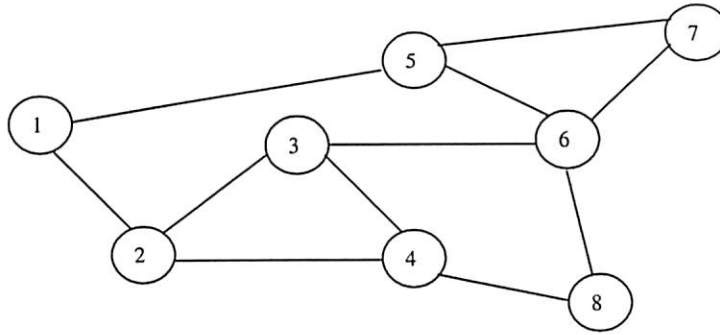


Figure 6.22 A Random Sparse Network

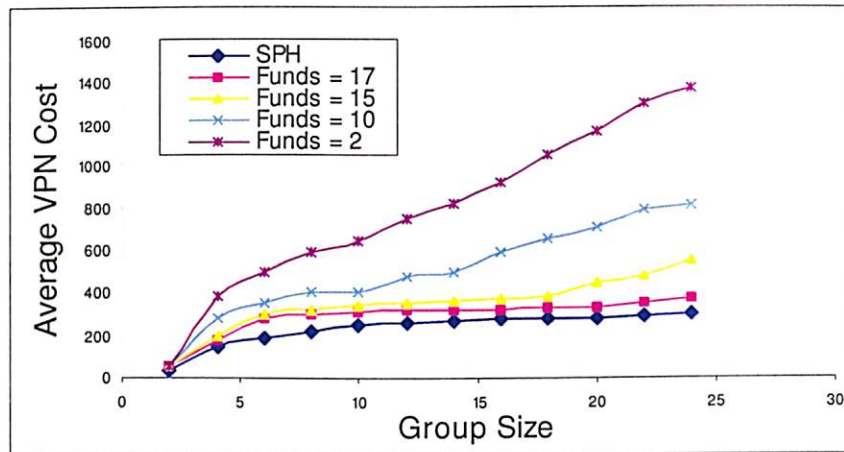


Figure 6.23 Average Cost with Single Link Failure

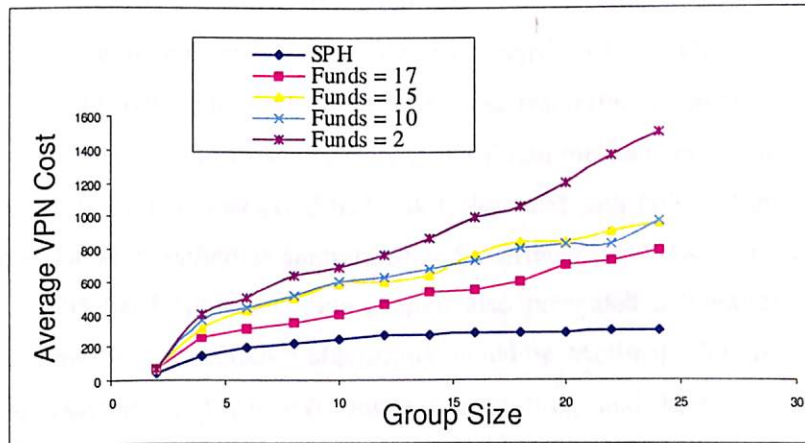


Figure 6.24 Average Cost with Multiple Link Failures

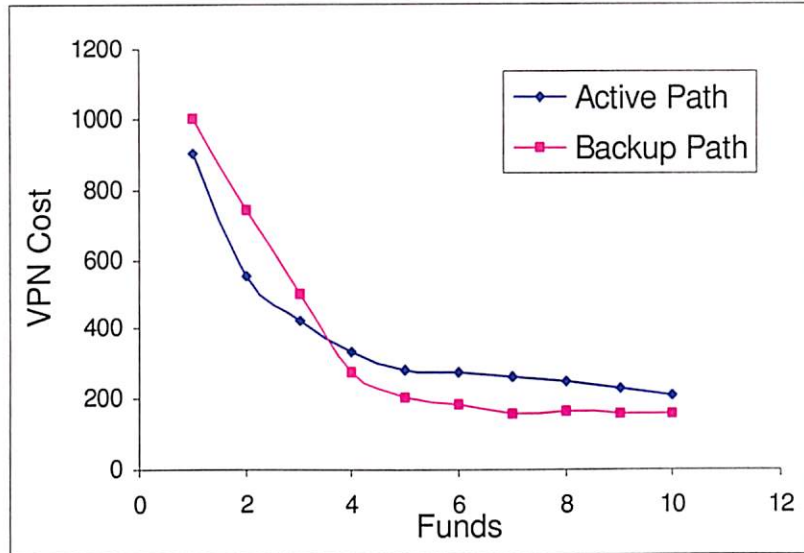


Figure 6.25 Funds Vs. Active Path and Backup Path Cost

6.6 Summary

This chapter described the strategies and algorithms for restoration of IPVPNs. Restoration can be implemented at different layers of OSI stack, physical, link, and network layer out of which physical layer restoration guarantee is the fastest one. We compared each of these approaches. Different mechanisms for implementing link layer restoration are discussed like link failure and path failure. Link restoration algorithms can be classified as centralized and distributed. Our work in this thesis is based on centralized approach. This chapter also presented a literature survey on unicast and multicast restoration algorithms including Minimal, Complete, and Full information scenarios; Successive survivable routing; and Minimal interference algorithm etc.

Section 6.5 described the problem of unicast restoration and multicast restoration in IPVPNs. We also presented unique solutions to these two problems. For

handling unicast restoration, our algorithms first compute active VPN tree. This heuristic algorithm first traverses each node in the graph using Dijkstra's SPH producing a VPN tree, and then it is optimized using Funds to decide upon the number of tunnels that are to be formed in the active VPN tree. Next, we have proposed a backup path computation heuristic that computes backup VPN tree, which is then optimized using the same optimization procedure to produce an optimal backup VPN tree. The next part of this section (section 6.5.3) proposes two heuristics for solving minimal cost multicast VPN problem which has application in the hose model. In this part we don't again recreate the multicast VPN tree whenever there is a single link failure. These algorithms are segment based which have definitely an edge over link disjoint or arc disjoint approaches of restoration. We compute critical primary segment and non-critical primary segments first. Then backup segments are computed for guaranteeing survivability. The backup bandwidth is shared with other active and or backup path bandwidth.

The sections 6.5.4 and 6.5.5 present the analysis of our unicast and multicast restoration algorithms. Different behaviors plotted show the performance of our algorithms in terms of active and backup VPN cost, average cost with multiple link failures, average number of active core routers etc.

Chapter 7

Layered Framework for IPVPN Deployment

7.1 VPN Deployment Challenges

Today's enterprises have many needs that can vary due to their size and physical infrastructure comprising of single locations, multiple locations, mobile workers, telecommuters etc. These enterprises all have one thing in common though i.e. they need to securely connect all of their workforce locations internally with each other and with the outside world. Normally, leased lines establish a private network connecting campuses or branch offices of an enterprise over a traditional WAN. As the lines are dedicated, security and bandwidth guarantees are ensured. But, as the corporate business environment has changed over past years, the above traditional hub and spoke WAN design is facing challenges like adding a new site can be slow and difficult, meshing is costly, off-net connectivity with remote users and extranet partners is expensive and difficult to manage, adding IP services is costly, and international connections can be very expensive. Hence, traditional networks will not meet these demands. The Internet revolution and rise of public networking has dramatically altered the networking requirements and opportunities of the enterprise. The deployment of new IP applications and the availability of ubiquitous Internet connectivity promise to facilitate the exchange of critical information both within the enterprise and throughout its sphere of influence. As a result, in recent years there has been a considerable amount of interest in offering Virtual Private Network services over the Internet that has become a low cost backbone. It is "virtual" in the sense that

it logically shares the physical links. It is “private” in the sense that it isolates traffic using tunneling and encryption. The VPN hype will continue in the years to come, due to the rising desire for economical, reliable, and secure communications [Zeng03]. Cahners In-Stat Group estimated that VPN services would hold a \$23.7 billion share of the total \$104.4 billion worldwide IP service revenues in 2005. In the year 2000 only about 20% of corporate traffic was carried by IPVPNs, but as expected by Yankee group, it would increase to 46% by 2006 as shown in figure 7.1.

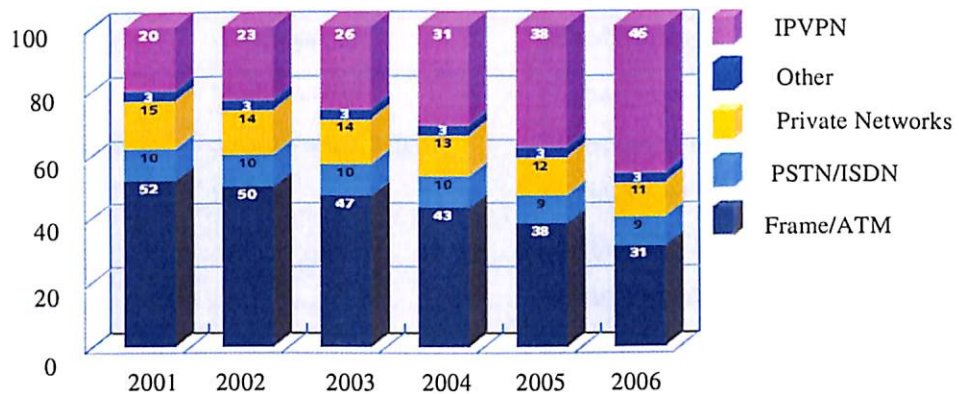


Figure 7.1 Data Traffic Mix for North American Multinational Corporations [Source: the Yankee Group, 2002]

According to IDC US WAN manager survey, corporations primarily go for implementing an IPVPN because of security concerns. This is shown in figure 7.2.

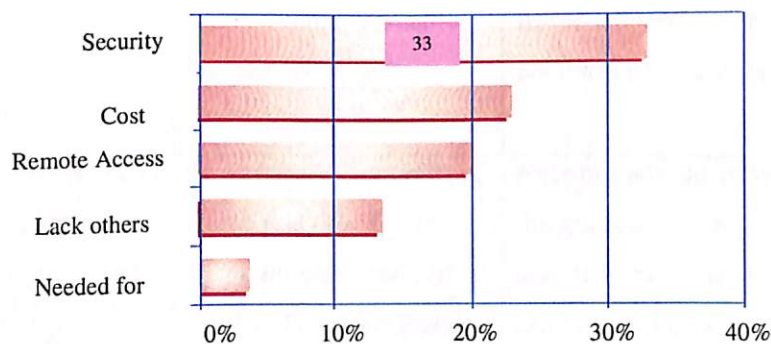


Figure 7.2 IDC WAN Manager Survey

7.2 Challenges with Current Legacy Technologies

Traditionally, the corporate has addressed the problem of connecting multiple sites, business partners, and suppliers using legacy technologies private lines, frame relay, and ATMs. However, with the shifting enterprise requirement landscape these legacy technologies fall short. The following table (Table 7.1) outlines the key advantages and disadvantages of these technologies.

Table 7.1 Advantages and disadvantages of current legacy solutions

Technology	Advantages	Disadvantages
Private Leased Line	High level of performance, Highly Secure.	Highest cost solution, Less flexible in designing network redundancy, and meshing is not economically practicable, Integrating remote users requires additional infrastructure, No support for burstable speed.
Frame Relay	Software defined VCs, Redundancy is built within the SP's network, Lower cost, Good alternative for sites with low bandwidth requirements, Virtually any service speed.	Expensive for large mesh networks as each site to site requires a PVC, Network administration is difficult, Integrating remote users requires additional infrastructure, Lack of interconnectivity between different frame relay providers hence not conducive for B2B solutions.
Asynchronous Transfer Mode (ATM)	Leverages VC concept of frame relay for lower costs, Inherent class of service, Good for higher bandwidth connections.	Network administration and integration challenges are same as that of frame relay, Generally more costly than frame relay, High complexity.

Overall, legacy solutions guarantee basic connectivity in a reliable fashion, but at the same time these are costly and difficult to manage. This is because of the new requirement of the enterprise to fit these technologies into the IP-centric world. Enterprises can find out a solution for this problem in the form of IPVPN deployment. The enterprises wishing to implement a VPN should strongly consider the issues discussed in the following section related to the planning phase.

7.3 VPN Planning Issues

In general, the following issues should be considered by an enterprise for implementing a VPN in its planning phase.

Scope: When deciding to use a VPN technology, it is important to consider the potential future uses rather than just the immediate requirement. If it is not done, then the enterprise may have to be locked into using the current technology for a long period. Under this planning task, the enterprise has to decide where to position the endpoints of the VPN i.e. the VPN gateways that are mainly responsible for encrypting/decrypting packets and forming tunnels. There are mainly four different types of VPN that drive this requirement. These are Office-to-Office, Intra-Office, Dial-in user to office, and Office-Partner / Supplier. The issues to consider for Dial-in and Partner/Supplier are:

- How many individual VPNs are required?
- What kind of traffic will be permitted through the VPN? - open or controlled
- Whether internal network hiding or address translation is required?
- How deep the VPN is permitted to reach inside the enterprise network? – stops at perimeter or at an internal subnet
- How user authentication is managed?
- The consequences of the VPN endpoints not being available i.e. failover or redundancy
- How much data is expected to flow through the VPN?

Overall, legacy solutions guarantee basic connectivity in a reliable fashion, but at the same time these are costly and difficult to manage. This is because of the new requirement of the enterprise to fit these technologies into the IP-centric world. Enterprises can find out a solution for this problem in the form of IPVPN deployment. The enterprises wishing to implement a VPN should strongly consider the issues discussed in the following section related to the planning phase.

7.3 VPN Planning Issues

In general, the following issues should be considered by an enterprise for implementing a VPN in its planning phase.

Scope: When deciding to use a VPN technology, it is important to consider the potential future uses rather than just the immediate requirement. If it is not done, then the enterprise may have to be locked into using the current technology for a long period. Under this planning task, the enterprise has to decide where to position the endpoints of the VPN i.e. the VPN gateways that are mainly responsible for encrypting/decrypting packets and forming tunnels. There are mainly four different types of VPN that drive this requirement. These are Office-to-Office, Intra-Office, Dial-in user to office, and Office-Partner / Supplier. The issues to consider for Dial-in and Partner/Supplier are:

- How many individual VPNs are required?
- What kind of traffic will be permitted through the VPN? - open or controlled
- Whether internal network hiding or address translation is required?
- How deep the VPN is permitted to reach inside the enterprise network? – stops at perimeter or at an internal subnet
- How user authentication is managed?
- The consequences of the VPN endpoints not being available i.e. failover or redundancy
- How much data is expected to flow through the VPN?

Encryption and Key Management: Encryption uses some form of key to protect the data. VPNs can use the same key for setting up the tunnel and then for encrypting the data for the entire duration of its' existence. Alternatively, a fixed shared key can be used for the VPN initialization and then a unique session key to encrypt the data for the entire lifetime of the tunnel. A more robust mechanism could be to negotiate and change the session key either on demand of either communicating entity composing the VPN or at a regular interval. This is the basis of many key exchange protocols.

Resilience/Failover: Security in VPNs is taken care by tunneling, cryptographic algorithms, and integration with firewalls. But what about the connectivity? How effective is the VPN solution, if it goes down? Resiliency of the VPN is defined as the ability of the VPN to continue despite part of the system being unavailable due to component fault or attack. If resilience is required, then it further impacts the performance of the gateway and potentially the configuration of the communications infrastructure. Single points of failure should be avoided and replication of key resources should be required. Graceful degradation paths should be planned and disaster recovery contingencies should be tested. Also, the restoration timing below network and service thresholds is critical.

User Authentication: In order to use the VPN, how the users should authenticate themselves. As the VPNs should be used to transmit sensitive information, anyone should not be allowed to send anything. Similarly, the receiving gateway should authenticate the sending gateway or the user to confirm that it is the right entity.

Application/Service Access Rights: Implementing a VPN without any content control be at a protocol level, and/or, intra-packet level provides open access using any protocol to the destination system. The destination system has its' own security configuration as the only protection mechanism. So, providing application level access rights adds a second level of protection. Hence, providing filtering in the VPN is highly desirable.

Encryption and Key Management: Encryption uses some form of key to protect the data. VPNs can use the same key for setting up the tunnel and then for encrypting the data for the entire duration of its' existence. Alternatively, a fixed shared key can be used for the VPN initialization and then a unique session key to encrypt the data for the entire lifetime of the tunnel. A more robust mechanism could be to negotiate and change the session key either on demand of either communicating entity composing the VPN or at a regular interval. This is the basis of many key exchange protocols.

Resilience/Failover: Security in VPNs is taken care by tunneling, cryptographic algorithms, and integration with firewalls. But what about the connectivity? How effective is the VPN solution, if it goes down? Resiliency of the VPN is defined as the ability of the VPN to continue despite part of the system being unavailable due to component fault or attack. If resilience is required, then it further impacts the performance of the gateway and potentially the configuration of the communications infrastructure. Single points of failure should be avoided and replication of key resources should be required. Graceful degradation paths should be planned and disaster recovery contingencies should be tested. Also, the restoration timing below network and service thresholds is critical.

User Authentication: In order to use the VPN, how the users should authenticate themselves. As the VPNs should be used to transmit sensitive information, anyone should not be allowed to send anything. Similarly, the receiving gateway should authenticate the sending gateway or the user to confirm that it is the right entity.

Application/Service Access Rights: Implementing a VPN without any content control be at a protocol level, and/or, intra-packet level provides open access using any protocol to the destination system. The destination system has its' own security configuration as the only protection mechanism. So, providing application level access rights adds a second level of protection. Hence, providing filtering in the VPN is highly desirable.

Network Hiding: This issue relates to hiding an enterprise's internal network from visibility on the other side of the VPN gateway. Network Address Translation (NAT) functions being widely used for this task. Though it provides an additional level of security by eliminating a view of the network, it can in some cases lead to complex routing problems to solve.

Key Management and Interoperability: In CPE based VPNs, management of keys associated with VPN implementations is not a major problem. However, if different enterprises are trying to connect using a Network-Based VPN, this problem may be severe. If two parties agree to have some common approach, and exchange common methods for managing VPN related keys, then this problem is simplified. Due to different legislation in different countries, a key length available in one country may not be used in a partner's country. This undoubtedly leads to a reduction in encryption strength if two parties want to communicate across a VPN.

7.4 Comparison of VPN Solutions with Traditional Solutions

The legacy solutions described earlier were effective because they worked perfectly well for both VPN users and service providers at the time of their introduction to the market. How suitable are they in today's world? Table 7.2 compares their strength and weaknesses with well known CPE and Network-based IPVPNs.

Table 7.2 Traditional Solutions versus IPVPN solutions

Parameter	Legacy Services		IPVPN Services	
	<i>Leased Line</i>	<i>Frame Relay</i>	<i>CPE-Based</i>	<i>Network-Based</i>
Perceived Cost	Highest Cost	Cost effective for hub and spoke networks only.	Cost effective	Lowered capital and operational expenditure, Most cost effective.

Scalability	Least Scalable	Scalable for hub and spoke.	IP is scalable, but configuring individual CPE is an administrative challenge.	Highest Scalability for larger networks, Fully meshed and preconfigured.
Converged Video, Voice and Data Support	Well suited for individual applications.	Strong support for data applications, Voice and video endpoints have to be on the same frame relay network.	With recent innovations in the QoS and better service delivery, IPVPNs are the most suited for converged multimedia networks.	
Perceived Security	Secure due to dedicated circuits, but no encryption and authentication.	Perceived to be secure but lack encryption and authentication.	IP is non secure, but CPE based IPVPNs are secure.	Basic configuration perceived as secure from POP to POP and on par with frame relay, less secure than CPE based as the last mile is "in the clear", Optional encryption

				over the last mile makes this on par with CPE based.
Any-to-Any Connectivity	Static connection oriented technologies.		Inherent in IP, Cover most types of customer sites.	Inherent in IP, Cover most types of customer sites, Act as intermediaries for multivendor equipment connectivity.
Network availability	Redundant links require dedicated circuits.	More resilient than leased line, but carrier outages impact the entire network.	Multiple paths through multiple carrier networks are possible, Dynamically routed around network problems.	
Key Distinctions	More Controllable, Highest Level of Security at highest cost.	Cost effective to Leased Line, Good for multiprotocol traffic like SNA.	Higher perceived levels of security and control, harder to maintain and more expensive than network based.	Cost effective, secure and simplified native IP connectivity.

7.5 IPVPNs Supporting Mobile Users

Flexible, secure and cost effective in comparison to traditional solutions, IP-VPNs exhibit many of the traits enterprises are seeking today. Mobile VPN users have special requirements relating to resource adoption and customization. In this thesis we have discussed how mobile agent technology can be applied to IPVPN in finding a route for a tunnel with QoS guarantees and also how can it provide service capabilities to mobile users. In our approach, we have considered the requirement from a group of VPN users. We have developed a framework to use active packets that can help us in discovering an optimal QoS route based on the available bandwidth and link cost. Also we have described an alternative approach for resource management using a distributed resource management component. Taking into account, the structuring mechanisms enabled by standard mobile agent platforms, like regions, agencies grouped within regions, and places belonging to agencies, we have applied these structural principles to our target mobile communications environment. We have assumed that service providers have access to node's control environment, algorithms and states. A possible framework for supporting mobile user groups after the deployment of the VPN is suggested in this thesis. We have considered the concept of Place Oriented VPNs that are based on agent technology and can be built on top of existing VPN infrastructure. Under this section, the preliminaries related to mobile user support in a VPN is discussed and in section 7.9 we have presented the architecture and algorithms related to this objective.

7.6 Place Oriented VPNs (PO-VPNs)

A PO-VPN is an alternative virtual private network [Karnouskos00b]. It differs from a legacy VPN where the legacy VPN has its own resources which are managed by its own users in a policy related way. But current VPNs are not sufficiently flexible, have long set up and deployment times, are not dynamic enough to accommodate rapid changes, and can't be controlled by the users. It is again more difficult to manage if the VPN spawns different networks with different policies. PO-VPNs are a right solution for this problem. These are built on top of existing physical

networks and are controlled by agents. An example of a PO-VPN is shown in figure 7.3. Each node in figure 7.3 hosts at least one agency. An agency is the actual runtime environment where agents execute. Each agency runs at a host and has one or more places. A place provides a logical grouping of functionality within an agency. Agents that operate within a place have something in common. An agency contains a number of places and each place can have several subplaces. Places are created dynamically or statically. Each place has its own set of resources assigned by the administrator. Agents operate at these places and their subplaces.

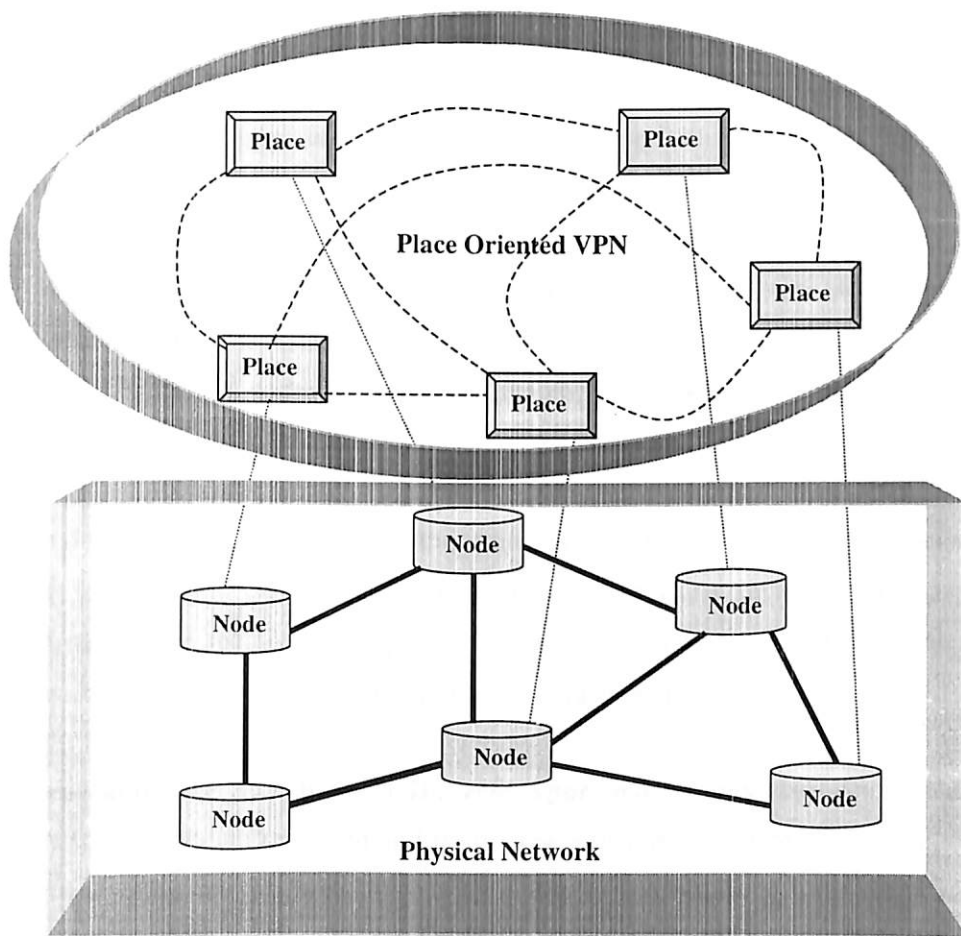


Figure 7.3 Place Oriented Virtual Private Network

These agents pass through the authentication and authorization procedures to access dependent resources from other places. Based on these agencies we could set up and administer a number of places in strategically located nodes, creating an art of virtual private network of places called as PO-VPNs.

7.7 Active Networks and Software Agents

Active Networks are a novel approach to network architecture in which the switches of the network perform customized computations on the message flowing through them. The first piece of this architecture is a protocol that would replace original “passive” IP packet with the “active” object code. The motivation of active networks is to permit applications to inject programs into the nodes of networks and then to support faster service innovation by making it easier to deploy new network services. Active networks are highly programmable networks that perform computations on the user data that is passing through them. In contrast, nodes of the traditional networks, such as ATM switches or IP routers, are closed, integrated systems, whose functions and interfaces are determined through standardized processes. These nodes transport packets (or cells) between end-systems and the processing of these packets is limited to operation on the header, primarily for the routing purposes. Specially, the network nodes neither interpret nor modify the packet payloads. Active networks, on the other hand, break this tradition by letting the network perform customized computation on the packets, including their payloads. There can be mainly two approaches to active networks: discrete and integrated, depending on whether programs and data are carried discretely.

Programmable Switches – a Discrete Approach: In this approach, the existing packet formats are remained and it provides a discrete mechanism that supports the downloading of programs, which separates the injection of programs from the processing of messages.

Capsules – An Integrated Approach: In this approach, the existing “passive” packet will be replaced with the “active” miniature programs, called “capsules”, which are

encapsulated in the transmission frames and executed at each node along their path. User data, the payload, will be computed and embedded in the capsules.

Mobile agents [Murch98] are software components that act alone or in communities on behalf of an entity and are delegated to perform tasks under some constraints or action plans. The essential characteristics of these agents are mobility that allows them to move from one node to another and continue their execution. An agent is a process that enables users to control their system, and at the same time, to allow sharing of resources, that is it controls the access to resources of a remote computer, and also provides protection. An agent system consists mainly of places. A place is a context in which an agent is executed [Karnouskos00a]. This context can provide services like access to local resources. A place consists of place name and address of agent system within which the place resides. A place can contain other places. All places follow parent child relationship. The assignment of places to agent system is done by ways like dynamically as they enter into different nodes based on some criteria like originating from same user or statically assigned per entity i.e. per user or per enterprise. The active node consists of a programmable router at the lowest layer, above which a node OS lies, which again supports Execution Environments. At the top, we have the application programs running. We have developed a novel architecture and algorithms for deploying VPNs for mobile VPN users in a group with quality of service guarantees using all these concepts described in this section.

7.8 Design Challenges: A Case Study

As an enterprise, if you plan to deploy an IPVPN as a replacement to your private networking, detailed planning and well thought out migration strategy is essential. We have considered our university (BITS) as an enterprise and proposed the design and deployment of IPVPN for its' users in this part of the thesis.

BITS network known as NEURON has several different users situated at different locations all over the world including remote access, extranet and intranet users. At Pilani, BITS has a campus wide intranet. The LAN is based on CISCO catalyst switches, multimode and single mode fiber segments, and UTP segments. The default gateway is a CISCO 3600 router that connects BITS LAN to the Internet with a 2 Mbps-leased line. A firewall protects the intranet from unauthorized outside access. The security mechanism currently enforced by few of the applications (like Virtual University, Student Welfare Division operations etc.) is based on authentication and authorization mechanisms implemented at application layer only. But as the hackers now days find new methods to break the security cover, a VPN solution could be the best that does not compromise the security at any level. The different types of applications that can be run with this new solution are:

- Off-Campus Programs like Virtual University, Collaborative Programs, etc.
- Admission Tests conducted at predefined locations all over India
- Placement Tests conducted for Practice School students
- Student Welfare Division Operations
- E-Commerce applications with Extranet Partners

But the problems in providing an IPVPN solution could be seen at different levels like Planning, Implementation, and Management as discussed in chapter 4. In this section we have tried to analyze a viable solution addressing the above issues and challenges meeting the objectives. The VPN solution that we are looking for could be accomplished in two different ways: one is by the Corporate itself (Enterprise-Managed Solution) and the other way is by outsourcing to a third party (Service-Provider Managed solution). In the former one, enterprises manage their own VPN-enabled customer premises equipment. In the later one, the VPN management complexity is shifted to provider edge devices. In this part of work, we have considered the former one. To understand the intricacies of IPVPN implementation both in Tunnel and Transport modes, a simulation was carried out as described in Appendix C.

7.8.1 VPN Gateways

VPN gateways act as endpoints for tunnels. An outbound IP packet from the private network arriving on the private interface is passed through to the public interface, where it is examined as per the outbound policy rules to determine whether it should be tunneled, sent without modification or dropped. An inbound packet arriving on a public interface will be decapsulated and examined against the inbound policy rules. Figure 7.4 shows inbound and outbound processing in a VPN gateway.

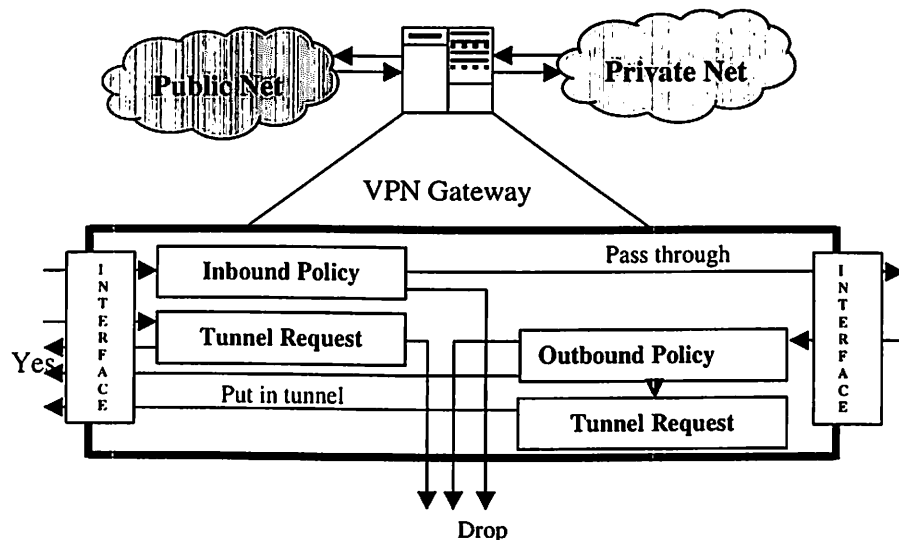


Figure 7.4 Inbound and Outbound Processing in a VPN Gateway

7.8.1.1 VPN Gateway Functions

The gateway should implement four basic functions - Tunneling, Authentication, Access Control, Data Integrity and Confidentiality. These functionalities of VPN gateways vary with respect to the type of user it is supporting like Remote, Intranet, and Extranet. In addition to the above categories of functionality, these gateways might have basic packet routing and forwarding capabilities, and also some basic firewall capabilities. As the routing protocols are not necessarily secure, gateways do not rely on them but rather rely on statically configured routes. Even if the VPN gateway is not processing the routing information, it should be able to relay routing exchanges to its next hop routers. VPN gateway might also implement various firewall functions, NAT being perhaps the

most common. NAT allows two privately addressed sites to share the same private address space. These gateways might also implement packet filtering to further restrict the traffic going into and coming out of the tunnel. VPN gateways also can perform another set of advanced functions like Quality of Service support for Network Applications, Redundant Failover, Hardware Acceleration, and Load Balancing capabilities for mission critical applications. Certain applications like voice over IP might require strict latency and jitter control so that the quality of the voice signal is acceptable. But QoS is a network wide consideration, i.e. for a VPN tunnel to have QoS, every hop and link along the tunnel path must have QoS support. But for a VPN gateway to have QoS support, we can assume that it has the capability of classifying and marking the offered traffic based on the type of the service quality the traffic should receive, and the capability of performing QoS negotiation with the network routers in the Internet infrastructure. For supporting mission critical applications, Gateway should have redundant failover capabilities or even better two or more VPN gateways should share the load of the VPN traffic simultaneously. Redundant failover can be implemented by two gateways communicating with a Hello protocol as in OSPF, and if one is failed, other should take over the connection. The encryption operation is computationally intensive and repetitive that takes lot of processor cycles. So, this work could be off loaded to hardware. This is essential for network speeds of T3 (45 Mbps).

7.8.2 Interaction with Firewalls

Firewalls prohibit entry and exit of unauthorized traffic into and from the intranet. The various mechanisms that can be employed for this are packet filtering, through stateful inspection, and using an application level proxy. Firewalls do not consider the security of the traffic after it leaves the corporate private network, but VPN gateways allow legitimate traffic into the private network and also ensure that traffic is still secure after it leaves the private network. We can have the following arrangements where firewalls and gateways can complement with each other as their functions overlap in many ways.

7.8.2.1 Gateway and Firewall in Parallel

Here, traffic going through the VPN gateway and firewall are separated; VPN traffic goes through the gateway and all other goes through the firewall. These two do not interfere with each other. The WAN side router needs to have a routing table entry for the gateway only, but the LAN side router must separate the traffic destined to a VPN tunnel from all other based on the destination address of the IP packets. This scenario is as shown in figure 7.5.

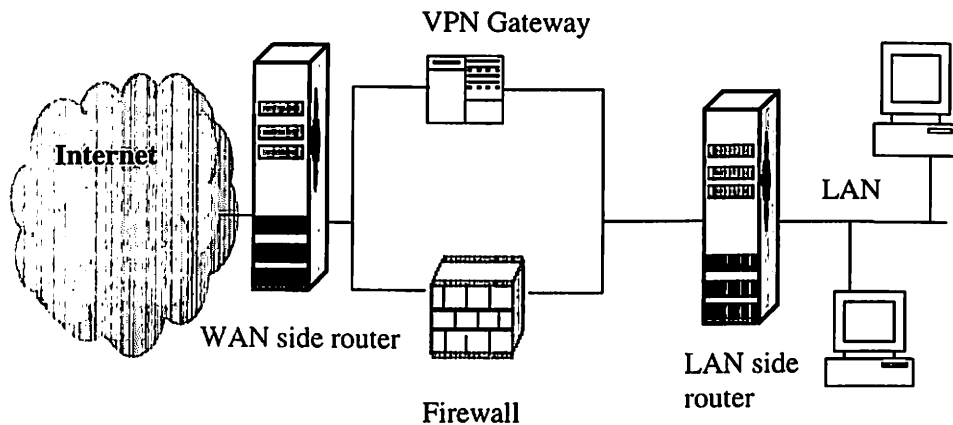


Figure 7.5 VPN Gateway and Firewall in Parallel

7.8.2.2 Gateway and Firewall in Series

There could be two ways of arranging these. VPN gateway is on LAN side and the firewall is on the WAN side, or the other way around. In the first case, specific rules must be installed in the firewall to allow VPN traffic to pass through. It is as good as opening a hole through the firewall; any tunneled traffic or tunnel creation message should be permitted access. As the tunneled packets are opaque, allowing them to pass through the firewall weakens the protection. Although the firewall cannot screen the tunneled traffic, it can establish rules for which type of tunnels are legitimate. All the traffic that crosses the firewall is also processed by

gateway. VPN packets get right treatment where non-VPN ones are simply passed through. Here access control policies are enforced by the firewall. In the later case, VPN gateway can also perform some access control on non-VPN traffic. As no tunnels penetrate past the gateway, all the traffic is in the clear and the firewall can apply policy rules. Figures 7.6 and 7.7 depict different scenarios.

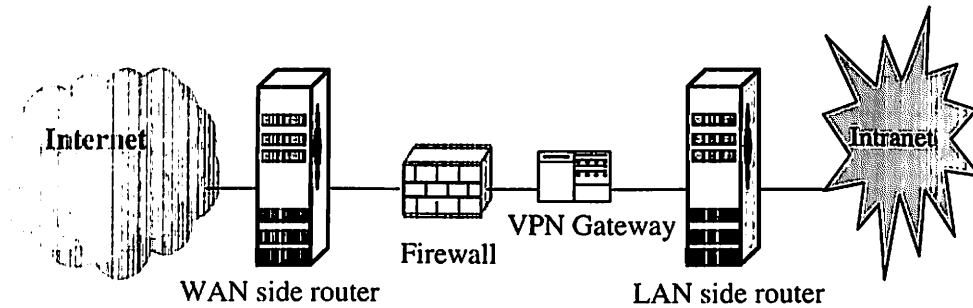


Figure 7.6 Gateway first, Firewall next

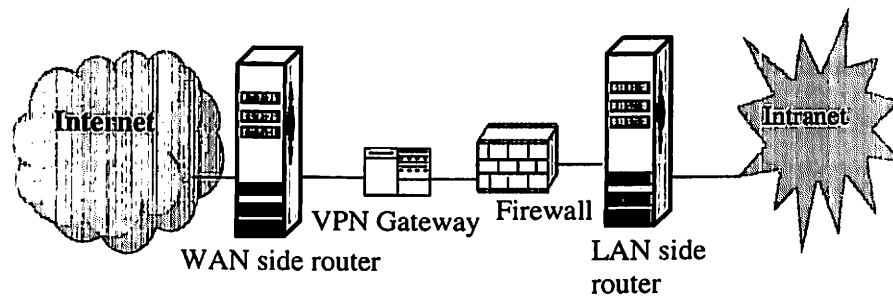


Figure 7.7 Firewall first, Gateway next

7.8.3 VPN Topology

Surely firewalls are not the only devices that interact with the gateway at the network perimeter. Other devices like routers, DNS servers, Authentication servers etc. do interact as well. We shall now consider another important design issue i.e. topology. Following types could be thought of for VPN deployment. A full mesh allows any site to directly talk to any other. But the designing is complex and it does not scale well. In hub and spoke one could be a central hub and spoke sites do interact indirectly through the hub. In a partial mesh, only selected sites are directly connected

and others indirectly. As the hub and spoke had a single routing bottleneck, partial mesh one is preferred. Different topologies are as shown in figure 7.8.

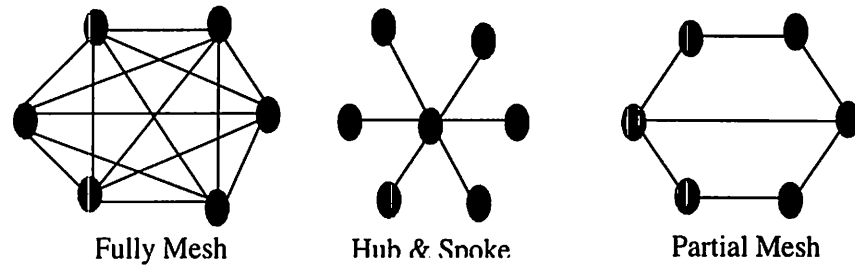


Figure 7.8 VPN Topologies

7.8.4 Addressing & Routing

When packets traverse through the Internet, private IP addresses must be hidden which can be achieved through the use of NAT. A registered IP address is required at the public interface of the VPN gateway to route encapsulated packets. Static routes are built up into the VPN gateway and neighboring routers to forward encrypted and clear text packets.

Address management in remote access VPN is complex than that of Intranet and Extranet VPNs since in LAN-to-LAN application the tunnel endpoints will most likely need to be fixed in advance. For remote access VPNs, it is often desirable that the user appears to the network in the same manner as if they had dialed into a traditional remote access server with a modem.

The user needs to have an address assigned from the address space of the intranet into which they are dialing in order to function as part of that network. Address management usually happens by one of the following four methods:

- The User's IP address is hard coded in their PC.
- The remote access server dynamically assigns the user an IP address from an internal address pool setup within the server.

- The RADIUS authentication server assigns the user an IP address as part of the authentication process. The address can be affixed one associated with the individual user or drawn from a RADIUS managed address pool.
- The user can have an address dynamically assigned from an address pool on an external server, managed by the Dynamic Host Configuration Protocol (DHCP).

In IPVPN applications, mechanisms assigning the address from an address pool are the most often used. IP addresses may be a precious commodity and preallocating IP addresses to individual users, whether or not they are currently connected, can use up a lot of addresses. The key challenge in assigning address from a pool lies in putting them back when the user logs off. In a scenario where the user's PC crashes or experiences network problems, the user's address may not be returned. This is known as pool "leakage". Over time this leakage can build up, emptying the pool that would cause subsequent users to be denied service. The DHCP protocol protects against this eventuality by using a system of "leases" that must be renewed periodically. If a user does not renew his or her lease, the address is returned to the pool rather than lost.

Once the user is assigned an IP address, it must be advertised to other devices within the enterprise intranet. Servers and other resources that the user may wish to communicate with have to know that the path back to the user is via the VPN server. This is the job of the routing protocol. VPN servers can advertise the subnets reachable through them to other routers within the network. They can even use protocols like Open Shortest Path First (OSPF), and Routing Information Protocol (RIP). Also, these routes could be manually entered into a neighbor router for propagation throughout the intranet.

7.8.5 Quality of Service

Despite most people's intuitive understanding of quality of service issues based on everyday experiences with airlines or on the roads, QoS remains a subject steeped

in controversy and emotion. Many of the protocol mechanisms needed to implement various flavors of QoS are not fully standardized. Neither a standard industry practice nor even a clear definition of terminology has been developed. QoS is very much a work in progress till recently. QoS mechanisms can be mainly classified into two categories:

- Fixed: These are the QoS mechanisms requiring outside intervention to preprovision network resources.
- Triggered: These are the QoS mechanisms done only on an as-needed basis, typically in congested periods.

Specific applications and IP addresses can be designated as a particular class or level of service. Applications are given a particular level of service based on their business importance and/or their technical nature (i.e. ability to handle latency). This assignment of QoS allows the IPVPN to support higher-level applications such as voice and video. End-to-end QoS is achieved through techniques such as Diffserv on the customer router. When QoS is required, packets must be classified, marked, and processed according to the QoS policy. VPN gateway can classify and provide QoS guarantees to the IP packet by examining the Differentiated Service Code Point (DSCP) field of IP header. In chapter 3, we have discussed different methods of QoS guarantees in IPVPNs using the standard techniques like Diffserv, MPLS and QoS based routing etc. that are available from the Internet community. Chapters (4, 5 and 6) of this thesis discuss the research objectives related to guaranteeing QoS in IPVPNs. Our main work in this thesis is based on this issue of quality of service guarantee in IPVPNs.

7.8.6 Scalability

An IPVPN solution that does not grow with the needs of your business is no solution at all. The successful IPVPN must be scalable in terms of both size and feature functionality. Scalability must be seamless, rapid, and easy to implement for

the IT administrator. Solutions must not scale at the cost of performance. Solutions that are poorly designed may experience performance degradation as features are turned on (e.g. firewall or IP services) or as more subscribers or sites are added. This makes for a poor solution in the long run as the shortcomings become clear.

Topology plays a role in making a VPN scalable one. Other factors that can influence this could be load balancing, the number of concurrent tunnels, and rapid bandwidth provisioning. In Intranet VPN, amount of bandwidth is the major concern, where as in Remote access VPN, both the number of simultaneous tunnels and the amount of bandwidth required are of major concern.

CPE-based VPN devices are limited by the number of tunnels they can support. Equipment that can support several hundred simultaneous tunnels gets very expensive and time consuming to manage. Network-based VPNs offer fully meshed, pre-configured tunnels thereby increasing scalability. It is much easier to add virtual routers, security policies and links in the network than configuring devices on premise. This reduces the time to bring a site online and allows it to start contributing to enterprise's productivity. Flexible licensing schemes and remotely enabled software and feature upgrades should be part of the solution. Finally, a range of access and trunking speeds and protocols should be available to support both growth and the mix of protocols that enterprise network administrators must deal with.

7.8.7 Resiliency

Security in VPNs is taken care by tunneling, cryptographic algorithms, and integration with firewalls, but what about the connectivity? How effective is the VPN solution, if it goes down? Resiliency of the VPN is defined as the ability of the VPN to continue despite part of the system being unavailable due to component fault or attack. VPN vendors can meet this challenge by providing any of the following solutions:

- Compute Redundant Physical Paths to reduce dependency on a single Service Provider.

- Support component or device redundancy to eliminate a single point of failure.
- Share state and VPN state to maintain both the session and VPN connection in the case of a failure.
- Automatically establish a new route without manual intervention. This can be achieved by employing any dynamic routing strategy.
- Provide multiple overlay paths or redundant VPN tunnels from one point to another.
- Achieve VPN connection failover quickly.

By combining all these above resiliency components, we can achieve a truly fault tolerant solution or optimal resiliency. Few examples are given in figure 7.9.

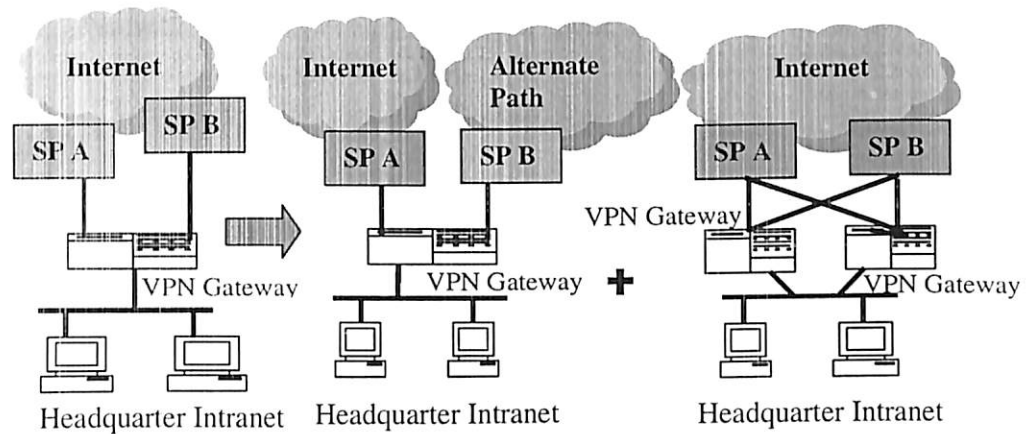


Figure 7.9 IPVPN Resiliency

7.8.8 VPN Solution Scenario

Let us assume that the following are the details of the internal networks and servers:

BITS, Pilani Headquarters Internal Networks	IP Address Pools
	10.0.1.0 / 24
	10.0.2.0 / 24
	10.0.3.0 / 24

Entrance Exam Network	4.0.4.10 / 24
Placement Test Server	4.0.3.10
Business Partner Server	4.0.5.10
E-Mail Server	4.0.6.10
SAP Server (SWD & PS)	4.0.7.10
VoD (Virtual University)	4.0.8.10
BITS, Dubai Branch Office	
Internal Network	10.0.4.0 / 24
BITS, Goa Branch Office	
Internal Network	10.0.5.0 / 24
Business Partner Network	
Partner 1, Bangalore	170.1.0.0 / 24
Partner 2, Hyderabad	172.1.0.0 / 24

Here, we want to develop a VPN solution for the above scenario. Let, authentication between all gateways is to be performed via digital certificates and authentication between remote users and VPN gateway is to be done via RADIUS authentication. We are also assuming that the whole solution is an Enterprise-Managed Solution.

Following are the steps to achieve this.

Step 1: Getting Internet Connectivity from ISP.

Let us assume that the above networks have Internet connectivity as given below:

Pilani, Dubai and Goa	:	T1 (1.5Mbps)
Partner 1 and Partner 2	:	T3 (45 Mbps)

Step 2: Select a specific VPN gateway for each location: Based on the performance level required at each location like bandwidth, throughput, and the number of remote users supported, decide upon the VPN gateway (7100, 3600, etc.) i.e. high end or low-end gateways are required. Let we have all 7100 series routers.

Step 3: Decide upon the topology.

Choose a full-mesh topology as we have only three corporate offices (BITS). Managing this will not cost much. In this we can form tunnels between any pair of locations directly.

Step 4: Decide upon the Intranet, Remote User, and Extranet VPN functions and also whether to combine these in the same VPN gateway for each location.

a) Intranet VPN: Let the requirements be described as follows:

- All private traffic must be encrypted using 3DES
- All the subnets should communicate with each other
- Digital Certificates should be used for authentication between VPN gateways.

IPSec with IKE are the probable choices for the first requirement. We will use IPSec's ESP in tunnel mode to fully encapsulate the intrasite IP addresses (4 or 10) into publicly addressed packets, so that the original net 10 or 4 addresses are preserved. Partners IP addresses do not need full encapsulation, as they are routable through the Internet. For the third requirement we will use PKI support, and also the digital certificate functions in IKE.

b) Remote Access VPN: As the remote access client does not have a permanent IP (every time a new IP will be assigned to him, by his ISP), the VPN gateway must provide the remote user an IP address that can be easily routed within the corporate network. So, the address assignment must be under the control of VPN gateway. Also, VPN gateway should pass information regarding other corporate servers like DNS, WINS etc. to the remote users without which he cannot communicate with the corporate network.

Let the requirements be described as:

- All private traffic must be encrypted using 3DES
- Users 1, 2, and 3 can access all the subnets at the corporate office, where as users 4, and 5 can access only the Entrance exam server (4.0.4.10)
- All users must access corporate intranet and the public Internet simultaneously

- Remote access users can use digital certificates or RADIUS to authenticate themselves with the VPN gateway.

All the above requirements define which functions are to be implemented in the VPN gateway, and which in the VPN client side. For the first requirement, we can use ESP and IKE. For the second requirement, group users so that access control rules can be applied to each group. The third requirement states that access be permitted to both public Internet and private intranet. For the fourth requirement, VPN gateway should support multiple authentication mechanisms.

c) Extranet VPN: These requirements will almost be similar to that of Intranet VPN scenario. The difference is that two companies and not just one are involved in the VPN configuration and policies. The gateway should provide fine-grain access control on the VPN traffic that can be achieved say by establishing restrictions at the level of applications on a specific host.

Step 5: Fix up the location of Gateway and Firewall. Let both be in parallel at all the locations.

Step 6: Decide upon the redundant failover gateways and load balancers. Let we do not have these features at present.

Step 7: Configure the CA (certification authority) server and an X.500 directory server at the corporate headquarter (BITS, Pilani) that should be accessible by all VPN gateways using digital certificates for authentication. Configure also a RADIUS server that should be accessible to the Gateway serving the Remote access users.

Step 8: Configure all the four VPN gateways as described below:

BITS, Pilani Gateway:

Four site-to-site VPN tunnels

- (10.0.1.0 / 24, 10.0.2.0 / 24, 10.0.3.0 / 24) < == > (10.0.4.0/24) serving all IP traffic

- (10.0.1.0 / 24, 10.0.2.0 / 24, 10.0.3.0 / 24) < == > (10.0.5.0/24) serving all IP traffic
- (4.0.5.10) < == > (170.1.0.0 / 24) serving Web traffic only
- (4.0.5.10) < == > (172.1.0.0 / 24) serving Web traffic only

Five User Groups

- Entrance exam user group: can access only the subnet 4.0.4.10 / 24
- Placement exam user groups: can access only the server 4.0.3.10
- PS students group: can access the SAP server 4.0.7.10 and 4.0.3.10
- Distance Learning Students group: can access VoD server 4.0.8.10
- Email user group: can access 4.0.6.10

All these users are assigned addresses from the address pool, and care should be taken to see that these addresses do not overlap with any address within the corporate network.

BITS, Dubai Gateway:

Two site-to-site VPN tunnels

- (10.0.4.0 / 24) < == > (10.0.1.0 / 24, 10.0.2.0 / 24, 10.0.3.0 / 24) serving all IP traffic
- (10.0.4.0 / 24) < == > (10.0.5.0 / 24) serving all IP traffic

BITS, Goa Gateway:

Two site-to-site VPN tunnels

- (10.0.5.0 / 24) < == > (10.0.1.0 / 24, 10.0.2.0 / 24, 10.0.3.0/24) serving all IP traffic
- (10.0.5.0 / 24) < == > (10.0.4.0 / 24) serving all IP traffic

Partner 1, Bangalore Gateway:

One site-to-site VPN tunnel

- (170.1.0.0 / 24) < == > (4.0.5.10) serving Web traffic only

Partner 2, Hyderabad Gateway:

One site-to-site VPN tunnel

- (172.1.0.0 / 24) <==> (4.0.5.10) serving Web traffic only

After the configurations are applied to the VPN gateways, it is time to implement the VPN and verify that all the components are working as designed. The proposed VPN design is given in figure 7.10.

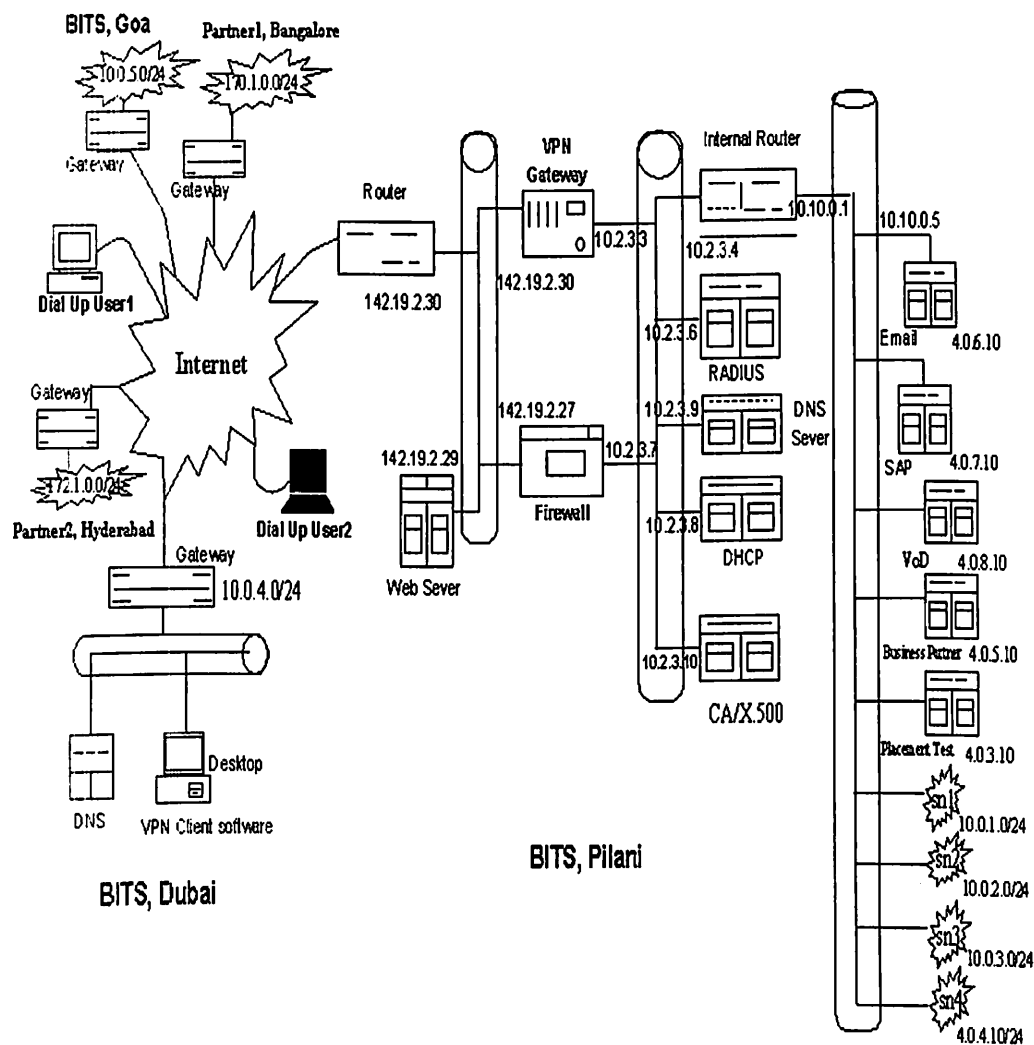


Figure 7.10 Proposed IPVPN Design

7.9 Mobile VPN User Support using Active Networks and Intelligent Agents

A majority of the theoretical work being done currently on the performance enhancement of the VPNs concerns security [Kent98]. Researchers have focused on devising different architectures for different deployment scenarios. Little has been done to provide QoS support in a VPN environment supporting roaming users (users who often move from one network to another). The main problem we face in supporting mobile users is the technology integration. These traveling VPN users must be supported in small groups or separately. We discuss here, how we can use agent technology and active networking concept to support our roaming users to get the VPN services from the corporate network. Also the tunnel that is to be formed between source and destination should guarantee maximum quality of service. VPNs are designed mostly with static users in mind and little has been done to integrate mobile users or to provide mobile user support after their deployment. The problem lies in the availability of information and communication services varies from place to place and from time to time. A major requirement here is the dynamic reconfiguration of the new environment and customization.

The advent of high-speed networking technology has enabled many multimedia applications like video conferencing, medical imaging and VOIP. These applications have different performance requirements of bandwidth, delay, jitter and loss rate, which leads to an important issue of how to support QoS on the modern high-speed virtual private networks.

It is known that the original Internet is a “best-effort” network, which has no QoS support for different classes of services. However, there may not exist one solution to all the problems of QoS support, because the application-specific state of QoS should be frequently changeable or easily deployable when needed, while the conventional network is simple and stateless. Now, recent research has shown that the programmable “*Active Networks*” may be a better solution to guarantee QoS in the VPNs.

Packet switched networks provide data exchange between end users. There is a sharp bound between what a user can do and what is offered by the network. These kind of networks can be looked as “passive networks”. They are affected by a number of problems: for example, it is difficult to integrate new technologies in a shared infrastructure and it is even harder to add new services to a pre-existing architectural model [Psounis99]. Traditional networks do not fit into the needs raised by new applications. This explains why it seemed suitable to figure out an innovative scenario where users can program the network themselves.

Active Networks represent a novel approach in the field of networking research [Calvert98]. In such networks, routers and switches have now an active role: they leave the simple “store and forward” packet processing task in favor of a new paradigm where intermediate network nodes may perform complex computations when forwarding packets. Furthermore, users can program the network to decide the specific treatment that their packets will experiment while in transit through the node. There are several proposals for active nodes developed at various universities. These may be categorized in two different families. In one, code injection is done out-of-band, bringing the concept of “configurable nodes” (*discrete approach*). On the other hand, in case of in-band code injection, the active nodes help realize distributed packet programming techniques (*integrated approach*). The architectural model of active nodes has been defined by DARPA (Defense Advanced Research Projects Agency). It is composed of a *Node Operating System* (NodeOS), one or more *Execution Environments* (EEs) and a number of *Active Applications* (AAs). The NodeOS is layered on top of a traditional operating system and has to manage requests for access coming from EEs. These, in turn, define a network API between users and active nodes. Finally, the AAs help customizing the services provided to the end users. They hold and execute the code directly injected by them.

VPNs are designed with static users in mind and almost all the recent VPN technologies do not support mobile users, i.e. once the VPN is deployed if the user is moving into another network (a third party) then again the same VPN has to be reestablished with the new network as the source. To do this, we must also have the required client software available at the new node and also it should support VPN

connections. If everything is right, we then have to reestablish the whole VPN connection starting from the scratch. This scenario of VPN deployment can also be called as flexible VPN. This requires resource adaptation and customization. To make this feasible, we have applied the concept of Agent technology & Active networks. The mobile agents used are active, mobile, and can travel. They may reside in a host or client computer, and roam other computers, networks, or the Internet to execute their tasks. They are frequently used to collect data, information, or changes. The use of these agents have several advantages like reducing the network load, overcoming network latency, encapsulating protocols, executing asynchronously and autonomously, dynamic and heterogeneous. These mobile agents can migrate along with mobile users, adopt local and remote resources dynamically and generally manage and mediate all requirements of mobile users.

7.9.1 Mobile VPN User Support

To support mobile users, we have allowed each user to act within his own place. This place hosts one or more cooperating agents that keep track of user's requirements and current status. Agents here take the role of customer, the service provider and the network provider. These agents reconfigure the services that user wants in the new environment. They also see that the adoption is optimal in the new environment. The reconfiguration is fully transparent to the users.

A VPN with mobile users comprises of a graph with changing nodes. As the user moves, we face the challenge of re-assigning the connections between the nodes in order to provide the same set of services. This whole process does take place without the notice of the user. To support mobility, we have to dynamically and flexibly provision VPNs. The deployment should take minimal time. There are multiple network and service providers in the underground infrastructure on top of which we want to build our VPN. Here, we consider three agents UA (user agent), VPN SPA (service provider agent), and NPA (network provider agent). The process for supporting mobile users is two fold: Initial VPN provisioning and Dynamic VPN

adoption. These are described in the following sections with the help of a set of steps that are to be performed in chronological order as described.

7.9.1.1 Initial VPN Provisioning

1. User Agents (UAs) negotiate and decide upon a common set of requirements and services desired by the users.
2. All UAs elect a leader using the Bully leader election algorithm (using the priority number, election and coordinator messages), and the leader is called as GA, the group agent.
3. Then GA tries to find out an optimal tunnel path for the VPN topology on the basis of bandwidth and link cost covering all the UAs. Here, GA issues self-routable packets, which travel in the network and get happiness levels from NPAs. The self-routable packet contains a time stamp according to which it has to follow back to GA. GA also negotiates with VPN NPAs to find out different resource supports.
4. Now GA interacts with the SPA in order to find out which topology supports the desired services. At the end of this phase, GA has a network topology that supports the services desired fulfilling the SLAs demanded by the user or customer or the UA.
5. GA then, integrates all the information received and uses a rule-based system to process the information. The happiness levels are averaged and the deviation is measured. The rule based system acts as a control element using this deviation as input.
6. It then issues active packets based on the rule-based system, which tries to restore the happiness levels. The happiness is here nothing but the bandwidth consumption and link cost. Thus routes are added and deleted from the routers dynamically to redirect the load so that the VPN is happy or well behaved. The GA does this thing. It can also consider other parameters like error statistics, reputation of nodes, cost security etc. in deciding an optimal path for the VPN.

7. Finally GA requests service and network providers to set up the services and the connections. As, a last step, GA informs all UA that the VPN is ready to use, after which it may die or remain alive as a central authority to monitor future requests and topologies.

7.9.1.2 Dynamic VPN Adoption

When a user moves from one network to another, his services must be maintained in the new environment. The steps below are needed to achieve this:

1. When a user moves from the home network, all the User Agents stop after getting an "operation stop" signal from the system. All the UAs then inform the GA, the central agent that the user is no more active in his home network.
2. When the user logs into another network after migrating, this new location address is sent back to the User Agents those were available in the previous domain but were stopped.
3. With this information, all the user agents migrate themselves to the new environment after authenticating themselves in the new domain.
4. The UA after establishing itself over the new domain examines the services needed by the user and whether these are supported & provided by the new domain. If yes, it configures those in the new environment. If not, UA may ask the old node to get the active code that implements the missing services in the new domain. This is where, active networks concept comes into play. The other possibility is having an agent based resource management system as shown in figure 7.11.

Local agent must perform the following actions:

- Locate a remote computer with the required resources
- Negotiate with that Computer to borrow the resources
- Invoke the service requested by the client
- Handle the Exception Conditions (resource revocation, and location of a new host and deportation of the guest process) in a way that is transparent to the client and guest

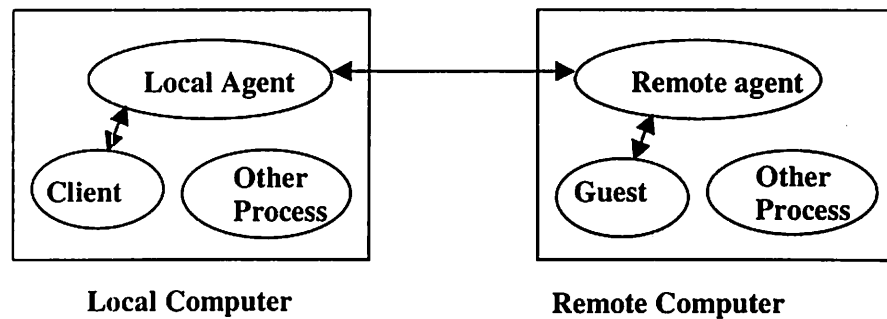


Figure 7.11 Relationship Between Different Processes

The remote agent process has to lend the resources and protect the interests of the computer owner. In particular it should perform the following actions:

- Receive the borrowing request from an agent
- Analyze the request to determine whether the resource can be granted
- Create an appropriate execution environment for the guest process (Creating a new process and supplying the guest with capabilities to access other components of the system, according to the request specification e.g. network connections to the client so that the guest and client can communicate directly).
- Exception handling, for example attempts to access resources, which the guest was not allowed to utilize, revocation of rights.

Providing remote resources or services in the above type of agent based resource management system can be categorized into three phases as Negotiation, Resource Utilization, and Resource Release with different messages being labeled as shown in figure 7.12.

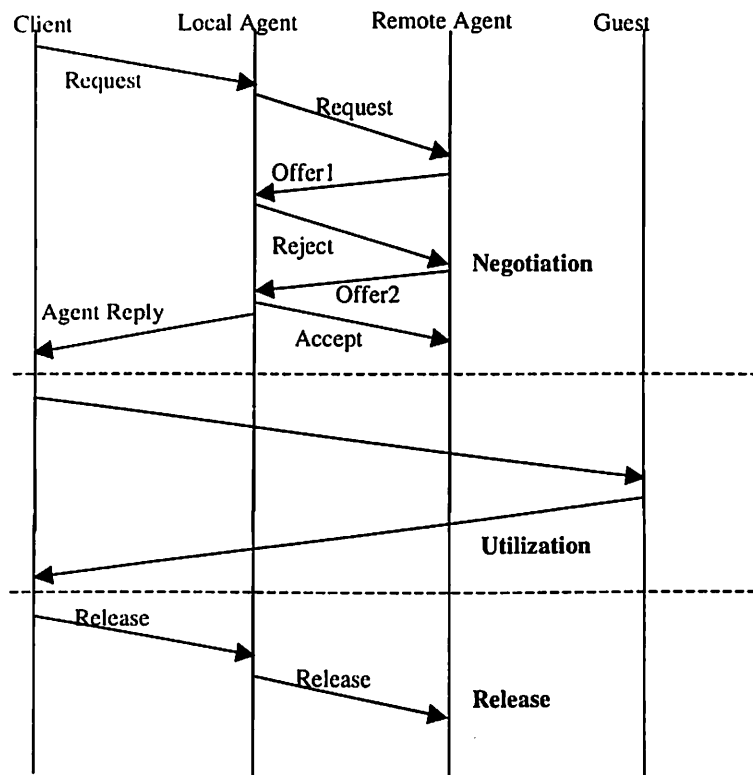


Figure 7.12 Negotiation and Utilization Messages

5. Finally, after establishing the new environment either with the help of active code getting transferred or with the help of a distributed message communication model as described in the above step, UA informs GA, which in turn multicasts this new VPN node to rest all affected UAs. This is done to enable all UAs to configure their local services to comply with the new topology of the VPN.

7.9.2 Active Quality of Service Routing

Steps 3 to 6 of section 7.9.1.1 (Initial VPN provisioning) has been considered as Active QoS routing, which is based on the concepts of active networks, where the behaviors of network elements, such as routers and switches, can be modified and manipulated dynamically by injecting customized codes into the network elements.

The basic functionality of active QoS routing is to find a network path to satisfy the given constraints that are required by the QoS requirements of a specific connection. In traditional routing, packets are delivered by using routing tables that are located on routers along the chosen route and the routing tables are formed only by the route information that is based on hops, source and destination address. But here selecting a path that provides QoS requires additional information, which are the traffic quality requirements such as bandwidth, delay, jitter and cost etc. In addition to the routing table, in AQR, the network elements also keep the state representation of the whole network or their sub-network and also with the routing algorithms the best feasible route from a given source to a destination that satisfies the QoS constraints is chosen. In addition, most active QoS routing algorithms consider the optimization of resource utilization. Here we would like to introduce two of the AQR technologies.

One of the technologies is using Active Packet [Karnouskos00a] that is a packet with a piece of executable code in its header. Basically, the idea of this technology is to find the best path according to QoS specifications by sending Active Packets from the source to destination, each time the first packet reaches the destination it sends a reply packet carrying the updated QoS and path information back to the source address. Thereafter, subsequent packets will follow this path. With a certain interval, the Active Packets are sent again to detect the network state, then, the source may change the path according to the reply packet information.

The other technology is using agent code at chosen routers through the network [Karnouskos00a]. Basically the agent is an independent executing code that can autonomously perform in response to events. With AQR, for the router to select the best path, the QoS parameters and traffic specifications are the most important factors, so these parameters are going to be kept in the agent routers' routing table. For example, the QoS parameters are residual bandwidth, loss probability, delay, Jitter, security status and cost etc. The traffic specifications can be categorized as video conferencing, FTP, email, audio stream and video stream or voice etc. As a result, the routing tables become multiplex and complex, which are called as search matrix as shown in the figures below.

Table 7.3 Search Matrix

	R0	R1	R2	R3
R0		R01	R02	R03
R1	R10		R12	R13
R2	R20	R21		R23
R3	R30	R31	R32	

In the above table, the entry R01 to R32 refer to the costs incurred by traffic delivering between two routers in the sub networks, for example R01 is the cost of traffic from R0 to R1. The following table illustrates every kind of the cost. As shown below, the cost depends on the QoS requirements and the types of traffic. With all these information the router will decide which path is the optimal and effective route, but probably not the shortest path, for the specific traffic. Off-course, every connection will not include every cost, it depends on the grade of QoS service that this connection requests. Like the routing table in OSPF (Open Shortest Path First) algorithm, the matrices are going to be updated with a certain interval. Otherwise the rapid changing link parameters will cause incorrect routing decisions.

Table 7.4 Different Kinds of Cost for Different Traffic

	Bandwidth	Cost	Delay	Hops
Video-Conf	M00	M01	M02	M03
FTP	M10	M11	M12	M13
Email	M20	M21	M22	M23

7.9.2.1 Active Packet Format

An active packet is the basic entity on which the whole Active Network Infrastructure is built. The structure of this packet is as shown in figure 7.13.

Seq No.	Type	Uniqueid
Groupid		pn
Data		

Figure 7.13 Active Packet Format

SEQ NO is an integer field that denotes the sequence number of the packet. The protocol used for their transmission is UDP hence proper assembling requires some ordering as datagrams arrive out of order. *TYPE* field can carry binary values of 0 and 1, where Zero denotes that payload is some code while one denotes that it is either normal user data or data used by some code previously sent. *UNIQUEID* field is used to authenticate the packet as having a valid ID. The field is also helpful in constructing the filenames for various protocols sent by multiple users. *GROUPID* field assists the above field of uniqueid in verification and differentiating among protocols sent by multiple users over the network. *PN* is the priority number of the packet. *DATA* is the normal user data. Only here it is wrapped around with another layer of headers for active identification of this packet.

These packets are wrapped in a UDP packet and sent for transmission. UDP is chosen to wrap it because the packet can itself control the reliability by the code in it. Also if the code is written in perspicacious manner the packets can also self-route itself based on the conditions imposed by the code & data placed in it. Here, the idea is to place intelligence in the packet, and not on the devices controlling the packet.

7.9.2.2 Server & Client Implementation

Server and client programs are used to send and receive active packets at various machines. Both the server and the client implementation are required at both the ends if they want to communicate. This is so because the nature of packet object is recursive, in the sense it can go to destination and try to contact back the master.

These two programs can be together called as the *Execution Environment*. The server process listens to connections from various hosts and receives packets from them. If they contain code then the packets are not only saved on the machine but also compiled and executed to change the behavior of the machine.

These daemons receive the active packets wrapped under the UDP cover. They open the UDP layer and read the active layer headers, validate the packet, and set an environment with access restrictions as laid by the system administrator for these packets at that end host.

As shown in figure 7.14, the execution environment consists of three processes namely daemon process, server process and client process, which are running together and using IPC (inter process communication) between them. First the daemon listens at UDP layer and picks up all active packets and delivers them to EE of the machine. The EE SERVER process of the machine understands the code in the packet. If valid, the code is sent for further processing else the active packet is dropped. The EE manipulator segregates the packet into code and data and forks a copy of it. The child process exec's to load on the executor and executes the code inside the packet. If this packet has code, which demands for new packet formation and delivery, then the EE CLIENT side is used to frame such packets and deliver them to UDP via DAEMON process for delivery. The acknowledgements of the active packets are in turn active packets again and use the same EE CLIENT side process.

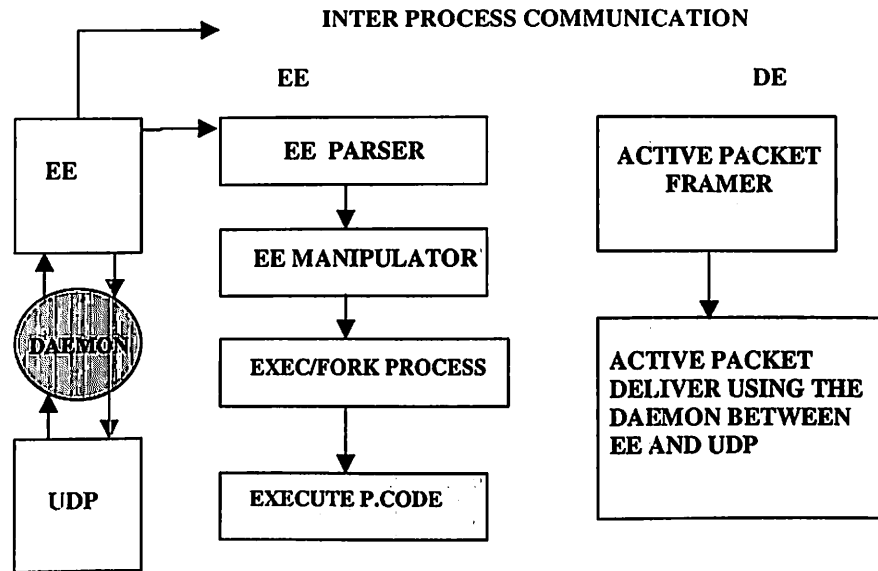


Figure 7.14 Structure of Execution Environment

7.9.2.3 Route Optimization & Routing Agent

Route optimization is implemented using two QoS parameters i.e. bandwidth & cost of the link. Their tables are maintained as explained in section 7.9.2. Routing agent is a simple C code that is transferred using the client to server where it is assembled and installed. The routing agent queries the route optimization algorithm, giving it the destination and asking it for next best possible router to move at. The route optimization code can be transferred prior to routing agent on the routers on the path and installed over there using the active network scheme. The routing agent just queries this to find the next possible position. The routing algorithm returns a structure called "*RouteLink*" which has the information regarding the next IP to go to and bandwidth available along with the cost of the link. The routing agent also connects back to the home address to tell the current router specification where it is. As this initial packet follows the whole route it keeps sending the "*communicator packets*" regarding the information of links it is traversing. The failure or loss of this

initial packet shall not lead to catastrophic failure of the whole system as the information about the whole route is being queued at the endpoint.

As the information is built up at the end node the rest of the data packets can follow the route that is built up. After some time another routing agent can be sent to get the route, as changes are possible over the path because of ascertain behavior of the parameters on which the system is working.

7.9.3 Analysis of Proposed Layered Framework

In the rapidly expanding world of business communications and services, IPVPNs are in the earlier stages of adoption among enterprises and service providers. However, by enabling large numbers of enterprises with high value services in a cost effective manner for both the enterprise and service provider their success will only increase over time. Here, we have presented a framework for deploying IPVPN as Customer Premises Equipment based solution in section 7.8. We have identified distinct layers and described major decision points within each layer. We have proposed a VPN solution and tested also our design by building a testbed with the available VPN hardware and software. We identified the design goals and proposed ways to achieve those. Each different VPN solution has its' own strengths, weaknesses, and vulnerabilities and it is anticipated that no single VPN solution will be sufficient enough but instead a diversity of choices will continue to emerge.

7.9.4 Analysis of Proposed Framework for Mobile VPN User Support

The Internet as of today does not support secure communication between groups of mobile users with minimal effort. We have considered the notion of PO-VPNs suggested in [Karnouskos00b], an alternative form of VPNs whose aim is to provide an infrastructure for groups with special characteristics like short time to live, low population etc. These PO-VPNs can be deployed on top of existing VPNs and offer customized services in a flexible, interoperable and cost effective way. We

have implemented the active QoS routing (section 7.9.2) using Unix socket API. As a result of this implementation, we found a path for a VPN tunnel (between a mobile user and corporate office) satisfying the QoS requirements as demanded by the user's service level agreement. To fulfill the requirement of mobile user group, a virtual environment with the help of active networks and intelligent agents can be a better option as described in our approach. Also in our approach, an alternative to active networks is suggested, where for resource management in the new environment the agents can communicate with each other as described in step 4 of section 7.9.1.2. But this option is a slower one because of message communication overhead. This can only be preferred when we have access restrictions or security problems in supporting active networks concept. It can be seen that active networks have many advantages over the traditional networks to support QoS. Firstly, it is an adaptive scheme that can support many classes of services. Secondly, it is more scalable than RSVP and possible to develop a simpler architecture to enable QoS than using IntServ and Diffserv together. Thirdly, active networks support user-driven computations and provide more powerful way for user to shape the traffic for particular QoS requirements.

7.10 Summary

This chapter described the challenges IPVPN customers are facing for their VPN deployment. It also provided clear-cut advantages of IPVPN over other legacy solutions like private leased lines, frame relay, and Asynchronous Transfer Mode (ATM). We compared all these on the basis of several factors like cost, scalability, security, connectivity and network availability etc. Another important part of this chapter is the use of agent technology to build an IPVPN to support nomadic or mobile VPN users in a group with QoS guarantees. We have described a viable solution and the technologies that are used to support mobile VPN users. These include active networking, agent technology, and place oriented VPNs.

We proposed a viable layered framework for designing IPVPN for our organization (BITS, Pilani) which is a deemed university. We outlined the

architecture with the help of several design challenges and policies. We also implemented the proposed case study with the help of a testbed with the available hardware and software. This chapter also describes the use of intelligent agents and active networking concepts to build up mobile VPN users in a group with quality of service guarantees. We have presented a framework for deploying intelligent agents to provision a VPN for mobile users in a group. We have also proposed a new architecture for active QoS routing to provision QoS enabled VPN paths for mobile VPN users. It also includes the way we can optimize the VPN route.

The sections 7.9.3 and 7.9.4 describe the merits and usefulness of our architectural solutions for deployment and supporting mobile VPN users in an IPVPN deployment scenario respectively.

Chapter 8

Conclusions

Customers ranging from large to small enterprises are depending more than ever on the networks to conduct their businesses. For that reason, either they are acquiring the appropriate management tools to administer and maintain their growing customer premises networks or outsourcing the management of their network resources to third parties. Moreover, customers are seeking to control and manage the VPN services they are subscribing to. There are several reasons for that. Above all is the possibility for customers to control and manage their VPNs according to their own policies reflecting their business goals. A VPN service provider cannot easily accommodate a large variety of service requirements of the various customers. Customers may have different traffic requirements like data, voice, and video with different priority schemes and performance characteristics. They often require different security levels. Another important reason for customers to control and manage their VPN is to perform the necessary partitioning of the VPN resources among the different end users and applications they support, and to implement their own policing mechanisms.

This thesis described algorithms and strategies for design and implementation of VPNs guaranteeing Quality of Service in IP Networks. The main constraint of the current IPVPN architecture is its inability to provide quality of service guarantees to sophisticated real time applications like integrated voice, data, and multimedia applications. In this thesis we have provided novel optimal tunnel path finding algorithms, bandwidth management algorithms, and restoration algorithms for the purpose of guaranteeing QoS. We have also presented an architecture and algorithm for supporting nomadic users in IPVPNs by using the active network concepts. This chapter contains discussion and conclusion of the work done and future work.

8.1 Optimal Tunnel Path Finding

As new applications like IP telephony, video on demand, video conferencing, etc. are being planned on the Internet; the demands for the network infrastructure will grow. To accommodate these applications, and added features like ubiquitous reach, adding new nodes in real time, proving QoS guarantees, and enhanced security, current day networks like Frame relay, ATMs will not be enough. IPVPNs can provide all these features and will be the preferred network of the future. Nowadays OSPF, and BGP are being considered for providing some sort of QoS guarantees like performing load balancing, and automatically recovering from failed access links etc. Even, multiconstrained routing algorithms or heuristics can be used to satisfy multiple independent QoS constraints. These routing algorithms have not been tried out with IPVPNs for reducing VPN tunnel maintenance cost. Also these are complex and time consuming routing algorithms.

In this part our approach with an intelligent search and optimization technique definitely gives a different dimensionality to the problem of minimal VPN path finding. Here, we have addressed the problem of determining a layout of VPN tunnels for an Intranet VPN taking into consideration the cost of links, which is expressed in the form of QoS parameters and the nodes have heuristic values depending on their suitability to provide more QoS. We have not activated any core router as the tunnel endpoint in the initial part to get a solution to MC-VPN problem in a CPE based approach. The heuristic function and search algorithm is used to find the optimal route. This algorithm (section 4.5) is admissible, monotonic and highly informed. It solves the problem of NP-hardness of MC-VPNs. This solution ignores the use of core routers as active nodes thinking that these are ISP routers. In the later part, as an extension, optimization was used to find out which of the core routers can be activated taking into consideration the maximum allowed number of activating core routers. The graphs plotted (section 4.6) show the usability of our approach in providing quality of service enabled tunnels between corporate gateway and its' branch offices. This approach gives us better services with the expense of computational cost involved in computing the heuristic estimation.

8.2 Dynamic Bandwidth Management

One objective of our thesis was to formulate new dynamic bandwidth allocation algorithms which successfully address the QoS problem. In a dynamic environment the VPN users must be provided with additional capacities if they are exceeding the allocated quota as per their service level agreement. If they are under-utilizing the allocated capacity, then some amount of bandwidth from this can be given to the users who are exceeding their quota so as to increase the revenue generation of service provider. This automatically meets the desired user satisfaction level.

The traffic model considers IPVPN traffic into three main classes brittle, stream, and elastic. Brittle applications require constant bandwidth, stream applications require only the minimal bandwidth, and elastic traffic does not require any bandwidth guarantee. We have used user utility as the metric to decide upon the increase or decrease of the allocated bandwidth.

In this part of thesis, we have proposed bandwidth-resizing algorithms for Intranet Virtual Private Network scenario using software agents. The dynamic bandwidth management algorithms are designed keeping in view of efficient network (ISP's) bandwidth utilization and the different degrees of urgency between the users. Our algorithms are user oriented. The way the link bandwidth or the bandwidth allocated to tunnels over the link is partitioned depends on the user's utility. Each sudden change in the users requirement is addressed by a linear change in the partitioning parameter (α). By using the approach suggested VPN service providers can satisfy more number of users with quality of service guarantees and also at the same time they can increase their revenue. The proposed algorithms are scalable. They work fine with a decent number of user connections per tunnel.

8.3 Unicast IPVPN Restoration

In this part of the work new algorithms for restoration strategies that provision bandwidth guaranteed recovery for unicast connections in an IPVPN environment are proposed. The primary focus is on online restoration strategies that are more suitable for on-demand and dynamic traffic engineering control. We evaluate the algorithms in four dimensions: the bandwidth requirement, number of connections, funds, and the computational time.

In this part we proposed heuristics for finding restorable paths for IPVPNs with QoS guarantees. The APH and BPH produced a better solution to MC restorable VPN problem. Both the algorithms first used CPE based approach to find out VPN path and then network-based optimization to strategically activate core routers to produce an optimal path. Our restoration algorithms are path based. We compared our results with well-known approximation algorithms for STP [Takahashi80], and found the results close to optimal performance.

8.4 Multicast IPVPN Restoration

This part of the thesis proposes new restoration strategies based on the state-dependent tree restoration where a different backup tree is calculated for each failure scenario that affects the service tree. The new strategies are converted into Integer Linear Programming formulations and are evaluated using simulations. Results show that the proposed state-dependent tree restoration strategies are significantly more bandwidth efficient than simple restoration (unicast backup paths). A mathematical model along with heuristics for building optimal restorable tunnel path in multicast IPVPNs is proposed in this part. We obtained an optimal solution to the problem through CPLEX optimization tool. In our approach, the backup path that we compute for each active segment shares its bandwidth with other backup paths hence we compute only one backup path for all those active segments that pass through a common set of links. We claim here that for restoring any active path that fails, we compute a backup path that gives us bandwidth guarantee at the cost of less overall

bandwidth usage. The optimum active path that we have computed also uses a new heuristics defined in this part of the thesis.

8.5 Layered Framework for IPVPN Deployment

In this part of the work we presented a framework for VPN deployment decision-making based on actual implementation experience. We identified different layers and described the decision points within each. IPVPNs are cost-effective, secure, and flexible solutions to a wide range of networking requirements. They readily meet the needs of the enterprise for secure, reliable, and cost-effective connectivity between sites, remote users, and business partners. However, choosing the right solution requires a look at the complete IPVPN solution from both a technology and a business case point of view.

8.6 Mobile User Support in IPVPNs

Mobile users are part of groups that have requirements that change unpredictably over time, because of the fact that they move constantly and spawn heterogeneous infrastructures. This kind of VPN user groups are interested in flexible VPNs that immediately adapt to environmental changes. Such groups can't use a basic version of a VPN to cover their requirements as they are not built with mobility in mind and they are difficult and uncomfortable in reconfiguration requests. Because of their behavior, these groups need to set-up and delete VPNs in minimal time. This flexibility to create and teardown such a virtual environment can be provided with the active network approach. Agents can be used to deploy new services and program the nodes according to application's needs.

In this part we have presented a novel approach to the implementation of Virtual Private Networks using the active networks concept. Our assumption here is that the current IP network models are not satisfactory with respect to the effective VPN support for mobile VPN users. We have shown how programmable network technologies help build and dynamically configure IPVPNs in a modular way and

with a high degree of flexibility. We presented a scenario where the VPN itself is seen as a powerful means to support heterogeneous information exchange between closed VPN user groups. We are convinced that active networks represent a big plus when dealing with such issues, thanks to the flexibility they provide when implementing non-trivial services for a large number of users. We have also described an architecture using which QoS path routing can be implemented using active networks concept. This model also includes the active packet format, and the search matrix definitions. We have also described an agent based resource management approach as an alternative to active networks.

8.7 Future Work

Nowdays many parties collaborate between themselves over the Internet using many available multicast applications like, video conferencing, whiteboard sessions, distance learning, multimedia video and audio broadcasts etc. Future VPN research directions would include MPLS VPNs towards this objective. MPLS together with its signaling protocols, either RSVP-TE (Resource Reservation Protocol-Traffic Engineering) or CR-LDP (Constraint Based Routing using the LDP) is the important mechanism for enabling QoS in IPVPNs. We have future plans to address these issues in IPVPNs.

The proposed bandwidth management algorithms in this thesis are scalable. They work fine with a decent number of user connections per tunnel. But, in a scenario where the numbers of users are very high, we may have to go for some sort of distributed scheduling. The workload of bandwidth managers can be distributed by either using broker architecture or by using distributed scheduling approaches. We plan to investigate these in our future work. Our bandwidth management algorithms can work as a good basis for an interesting research issue like utility-based admission control. In this approach, the incoming traffic flow should not be accepted in the IPVPN if its' acceptance degrades the performance of other already accepted traffic flows beyond the agreed performance levels. If we can uniformly assess and define the performance deterioration, and the ability to provide the requested service, this

approach then would be efficient. In connection utility, an incoming flow could present its' utility requirement to the IPVPN, and then the bandwidth manager would calculate whether it is able to provide that much required capacity or not. Using some of the admission control procedures we can do this. An alternate way is to simplify the admission control procedure by calculating a trunk reservation parameter, which would be equal to the amount of bandwidth that can be accepted in the IPVPN so that the overall utility of the active traffic on the VPN would increase. Then we can perform the usual admission control procedure based on the available bandwidth, with trunk reservation parameter as additional control of the utility level. We can also include other quality of service parameters in the utility function. We plan to investigate this further.

As the world places growing reliance on communication or computer networks, network resilience (survivability) will continue to be a mission critical topic for the research community. The new restoration strategies and algorithms proposed in this thesis contribute little towards this field of interest. We do not consider here node failures. Here, we do not consider dynamic traffic i.e. setting up path for multicast sessions as and when the requests arrive. We do not also consider multiple multicast VPN scenarios. We plan to extend our work along these lines. The restoration strategies proposed in this thesis guarantees bandwidth recovery in the event of single failures. One of the objectives of the algorithms is to minimize allocations of resource to support the bandwidth guarantees. In the event of multiple failures, it may so happen that more resources are required to restore full bandwidth to the connections. Instead of providing guaranteed bandwidth recovery for multiple failures, which is hard or may be impossible, a better approach is to provide guarantees to single failures only and use dynamic restoration strategies to deal with the unexpected failures. Resource overprovisioning can be used to increase the probability of traffic recovery during multiple failures. Research can be directed into finding dynamic restoration strategies (and algorithms) that provide the most efficient trade-off between the level of overprovisioning and the probability of traffic restoration.

References

- [Alchaal03] L. Alchaal, V.Roca, A. El-Sayed and M. Habert, "A VPRN Solution for Fully Secure and Efficient Group Communications," *Research report, INRIA*, April 2003.
- [Amit01] Amit Kumar, Rajeeb Rastogi, A. Silberschatz and B. Yener, "Algorithms for Provisioning Virtual Private Networks in the Hose Model," in *Proceedings of 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, p.135-146, August 2001, San Diego, California, USA.
- [Andersson00] L. Andersson, P. Doolan, N. Feldman, A. Fredette and Bob Thomas, "LDP Specification," *Internet Draft*, August 2000.
- [Anupam01] Anupam Gupta, J. Kleinberg, Amit Kumar, Rajeev Rastogi and B. Yener, "Provisioning a Virtual Private Network: A Network Design Problem for Multicommodity Flow," in *Proceedings of Thirty-third Annual ACM Symposium on Theory of Computing*, Hersonissos, Greece 2001.
- [Awduche99] D. Awduche, J. Malcom, J. Gobua, O'Dell and McManus, "Requirements for Traffic Engineering over MPLS," *RFC 2702*, September 1999.
- [Banchs00] A. Banchs and Denda, "A Scalable Share Differentiation Architecture for Elastic and Real-Time Traffic," *8th IEEE/IFIP International Workshop on Quality of Service (IWQoS 2000)*, Pittsburgh, PA, June 2000.
- [Barr01] A. B. Barr, Afek, H. Kaplan, E. Cohen and M. Merritt, "Restoration by Path Concatenation: Fast Recovery of MPLS Paths," in *Proceedings of ACM SIGMETRICS*, 2001.
- [Barto83] A. G. Barto, et al. "Neuron like adaptive elements that can solve difficult learning control problems," *IEEE Transactions on Systems, Man, and Cybernetics*, 1983.
- [Bellcore95a] Bellcore, "SONET Dual-Fed Unidirectional Path Switched Ring (UPSR) Equipment Generic Criteria," *GR-1400-Core*, 1995.
- [Bellcore95b] Bellcore, "SONET Bidirectional Line-Switched Ring Equipment (BLSR) Generic Criteria," *GR-1230-Core*, 1995.

- [Berger03] L. Berger, "Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource Reservation Protocol-Traffic Engineering (RSVP-TE) Extensions," *RFC 3473*, 2003.
- [Bhattacharya03] S. Bhattacharya and Iannaccone, "Availability and Survivability in IP Networks," in *Tutorial for IEEE ICNP*, Atlanta, USA, November 2003.
- [Black01] D. Black, S. Brim, Carpenter and Faucheur, "Per Hop Behavior Identification Codes," *RFC 3140*, June 2001.
- [Blake98] S. Blake, D. Black, M. Carlson, et al., "An Architecture for Differentiated Services," *RFC 2475*, December 1998.
- [Bouch00] A. Bouch, M. A. Sasse and Demeer, "Of Packets and People: A User Centered Approach to Quality of Service," *International Workshop on Quality of Service 2000*, Monterey, CA, USA.
- [Boyan94] Boyan, Justin and Littman, "Packet Routing in Dynamically Changing Networks: A Reinforcement Learning Approach," *Advances in Neural Information Processing Systems*, 1994.
- [Boyle00] J. Boyle, Cohen, Durham, Herzog, Rajan and Sastry, "The Common Open Policy Service (COPS) Protocol," *RFC 2748*, January 2000.
- [Boyle02] J. Boyle, V. Gill, A. Hannan, D. Cooper, Awduche, Christian and Lai, "Applicability Statement for Traffic Engineering with MPLS," *IETF RFC 3346*, 2002.
- [Brahim00] Brahim, B. Gleeson, G. Wright, et al. "Network based IPVPN Architecture using Virtual Routers," *Internet Draft* (work in progress), July 2000.
- [Brown00] S. Brown, "Implementing Virtual Private Networks," *Tata McGraw-Hill*, 2000.
- [Calle01] E. Calle, T. Jove, P. Vila and Marzo, "A Dynamic Multilevel MPLS Protection Domain," *3rd International Workshop on Design of Reliable Communication Networks*, DRCN, Budapest (Hungary), 2001.
- [Callon99] R. Callon, Doolan, Feldman, Fredette, Swallow and Viswanathan, "A Framework for Multiprotocol Label Switching," *Internet Draft*, July 1999.

- [Calvert98] K. Calvert, S. Bhattacharjee, E. Zegura and J. Sterbenz, "Directions in Active Networks," *IEEE Communications Magazine*, 1998.
- [Cao99] Z. Cao and E. Zegura, "Utility Max-Min: An Application-Oriented Bandwidth Allocation Scheme," in *Proceedings of IEEE INFOCOM'99*, March 1999.
- [Chan01] Chan, et al., "A Measurement Based Congestion Alarm for Self Similar Traffic," *ICC* 2001.
- [Chandra02] Chandra and Anupam, "Building Edge Failure Resilient Networks," *9th IPCOC*, Cambridge, MA, 2002.
- [Choi96] Choi and Yeung, "Predictive Q-routing: A Memory-based Reinforcement Learning Approach to 1996 Adaptive Traffic Control," *Advances in Neural Information Processing Systems*, 945-951, Cambridge, 1996.
- [Cohen00] R. Cohen and G. Kaempfer, "On the Cost of Virtual Private Networks," *IEEE Transactions on Networking*, Vol.8, No.6, Dec 2000.
- [Crawley98] E. Crawley, R. Nair, B. Rajagopalan and H. Sandick, "A Framework for QoS Based Routing in the Internet," *RFC 2386*, August 1998.
- [Davis01] Carlton R. Davis, "IPSec Securing VPNs," *Tata McGraw-Hill Edition*, 2001
- [Duffield02] N.G. Duffield, P. Goyal and A. Greenberg, "Resource Management with Hoses: Point to Cloud Services for Virtual Private Networks," *IEEE Transactions on Networking*, Vol.10, No.5, Oct 2002.
- [Duffield99] N. G. Duffield, P. Goyal, A. Greenberg, P. Mishra and K. K. Ramakrishnan, "A Flexible Model for Resource Management in Virtual Private Networks," in *Proceedings of ACM SIGCOMM*, September 1999.
- [Dziong96] Z. Dziong, Xiong and Mason, "Virtual Network Concept and its Applications for Resource Management in ATM based Networks," *International IFIP/IEEE Conference on Broadband Communications*, Chapman & Hall, 1996.
- [Erlebach04] Thomas Erlebach and Maurice Ruegg, "Optimal Bandwidth Reservation in Hose-Model VPNs with Multipath Routing," in *Proceedings of IEEE INFOCOM*, 2004.

- [Fei01] A. Fei, J. Cui, M. Gerla and D. Cavendish, "A Dual-Tree Scheme for Fault-Tolerant Multicast," in *Proceedings of IEEE ICC*, Helsinki, Finland, June 2001.
- [Floyd95] S. Floyd and Van Jacobson, "Link-Sharing and Resource Management Models for Packet Networks," *IEEE/ACM Transactions on Networking*, August 1995.
- [Fox99] B. Fox and B. Gleeson, "Virtual Private Network Identifier," *RFC 2685*, September 1999.
- [Friesen96] V. J. Friesen, Harms and Wong, "Resource Management with Virtual Paths in ATM networks," *IEEE Network*, Vol. 10 No. 5, September/October 1996.
- [Gabow86] H. N. Gabow, Z. Galil, T. Spencer and Tarjan, "Efficient Algorithms for Finding Minimal Spanning Trees in Undirected and Directed Graphs," *Combinatorica*, Vol.6-2, pp. 109-122, 1986.
- [Garg00] Garg and Saran, "Fair Bandwidth Sharing among Virtual Networks: A Capacity Resizing Approach," *IEEE INFOCOM 2000*.
- [Glesson00] B. Glesson, Heinanen, Armitage and Malis, "A framework for IP based VPNs," *RFC 2764*, February 2000.
- [Gleeson99] B. Gleeson, A. Lin, J. Heinanen, Armitage and Malis, "A Framework for IP Based Virtual Private Networks," *Internet Draft*, 1999.
- [Goetz98] Goetz, Kumar and Miikkulainen, "On-line Adaptation of a Signal Predistorter Through Dual Reinforcement Learning," in *Machine Learning: Proceedings of the 13th Annual Conference*, Bari, Italy, 1998.
- [Goyal99] P. Goyal, A. Greenberg, Kalmanek, Marshall, Mishra, Nortz and Ramakrishnan, "Integration of Call Signaling and Resource Management for IP Telephony," *IEEE Network*, Vol. 13, May/June 1999.
- [Grover87] W. Grover, "The self-healing network: A fast distributed restoration technique for networks using digital cross-connected machines," in *Proceedings of IEEE GLOBECOM*, Tokyo, Japan, November 1987.
- [Hac99] Anna Hac and Zhou, "A New Heuristic Algorithm for Finding Minimal Cost Multicast Trees with Bounded Path Delays," *IJONM*, pp. 265-278, 1999.

- [Hakimi71] Hakimi, "Steiner Problem in Graphs and its' Implications," *Networks*, Vol. 1, 1971.
- [Hamzeh99] K. Hamzeh, G. Pall, W. Verthein, J. Taarud, W. Little and G. Zorn, "Point to Point Tunneling Protocol (PPTP)," *RFC 2637*, July 1999.
- [Harel84] D. Harel and R.E. Tarjan, "Fast algorithms for finding nearest common ancestors," *SIAM Journal of Computing*, Vol. 13-2, 1984.
- [Hassan00] M. Hassan and M. Atiquzzaman, "Performance of TCP/IP over ATM Networks," *Artech House*, 2000.
- [Heinanan99] J. Heinanan, T. Finland, F. Baker, W. Weiss and J. Wroclawski, "Assured Forwarding PHB Group," *RFC 2597*, June 1999.
- [Herzberg95] M. Herzberg, S. Bye and A. Utano, "The Hop-Limit Approach for Spare-Capacity Assignment in Survivable Networks," *IEEE/ACM Transactions on Networking*, 3(6), December 1995.
- [Hota03a] C. Hota and G. Raghurama, "A Heuristic Algorithm for QoS Path Computation in VPNs," *International Conference on Information Technology*, Bhubaneswar, India, December 2003.
- [Hota03b] C. Hota and G. Raghurama, "Building Virtual Private Networks to Support Mobile VPN Users in a Group with Quality of Service Guarantees," *International Conference on Electronic Business*, NUS, Singapore, December 2003.
- [Hurley99] P. Hurley and J. L. Boudec, "A proposal for an Asymmetric Best-Effort Service," *7th International Workshop on QoS*, London, June 1999.
- [Hyong03] Hyong and Konstantopoulos, "Dynamic Capacity Resizing for Fair Bandwidth Sharing in VPNs," *IEICE Transactions on Communications*, Vol. E86-B, No.5, May 2003.
- [Iraschko96] R. Iraschko, M. MacGregor and Grover, "Optimal Capacity Placement for Path Restoration in Mesh Survivable Networks," in *Proceedings of IEEE ICC*, Dallas, USA, June 1996.
- [Italiano02] G. F. Italiano, R. Rastogi and B. Yener, "Restoration algorithms for Virtual Private Networks in the Hose Model," *IEEE Transactions on Networking*, 2002.

- [Jacobson99] V. Jacobson, K. Nichols and K. Poduri, "An Expedited Forwarding PHB," *RFC 2598*, June 1999.
- [Jha02] S. Jha and M. Hassan, "Engineering QoS in the Internet," *Artech House*, USA, July 2002.
- [Juttner00] A. Juttner, et al., "On demand optimization of label switched paths in MPLS networks," *IEEE ICCCN 2000*.
- [Juttner03] Juttner, Szabo and Szentesi, "On Bandwidth Efficiency of the Hose Resource Management Model in VPNs," *IEEE INFOCOM 2003*.
- [Kar03] Kar, Kodialam and Lakshman, "Routing Restorable Bandwidth Guaranteed Connections using maximum 2-route flows," *IEEE/ACM transactions on Networking*, October 2003.
- [Karnouskos00a] S. Karnouskos, "Supporting Nomadic Users within Virtual Private Networks," in *Proceedings of GLOBECOM workshop on service portability*, 1st December, 2000, San Fran, USA.
- [Karnouskos00b] S. Karnouskos, I. Busse and Covaci "Place Oriented Virtual Private Networks," *International Conference on System Sciences*, Hawaii, 2000.
- [Kawamura99] R. Kawamura and H. Ohta, "Architectures for ATM Network Survivability and their Field Deployment," *IEEE Communications Magazine*, August 1999.
- [Kel97] F. Kelly, "Charging and Rate Control for Elastic Traffic," *European Transactions on Telecommunications*, Vol. 8, 1997.
- [Kennington01] Kennington and M Lewis, "The Path Restoration version of spare capacity allocation problem with modularity restrictions: Models, algorithms, and an empirical analysis," *INFORMS Journal on Computing*, Vol.13, No.3, 2001.
- [Kent98] S. Kent and Atkinson, "Security Architecture for Internet Protocol," *RFC 2401*, November 1998.
- [Key90] P. Key, "Optimal Control and Trunk Reservation in Loss Networks," *Probability in Engineering and Informational Sciences*, Vol. 4, 1990.
- [Kodialam03] M. Kodialam and T. V. Lakshman, "Dynamic Routing of Restorable Bandwidth-Guaranteed Tunnels Using Aggregated Network

Resource Usage Information,” *IEEE Transactions on Networking*, Vol. 11, No.3, June 2003.

- [Korzeniowski00] P. Korzeniowski, “VPNs become Key Part of Enterprise Networks,” *Business Communications Review*, March 2000, pp. 28-32.
- [Kumar01] A. Kumar, Rastogi, Silberschatz and Yener, “Algorithms for Provisioning VPNs in the Hose Model,” in *Proceedings of ACM SIGCOMM* 2001.
- [Kumar02] A. Kumar, R. Rastogi and A. Silberchatz, “Algorithms for Provisioning VPNs in the Hose Model,” *IEEE/ACM Transactions on Networking*, August 2002.
- [Kumar83] Kumar and J. Jaffe, “Routing to multiple destinations in computer networks,” *IEEE Transactions on Communication*, Vol. 31, March 1983.
- [Kumar94] Kumar, Grama, Gupta and Karypis, “Introduction to VPNs,” *The Benjamin/Cummings Publishing Company, Inc.*, 1994.
- [Kumar97] Kumar and Miikkulainen, “Dual Reinforcement Q-Routing: An Online Adaptive Routing Algorithm,” *Artificial Neural Networks in Engineering*, 1997.
- [Lau04a] W. Lau and S. Jha, “Failure Oriented Path Restoration Algorithms for Survivable Networks,” *NOMS*, Korea, 2004.
- [Lau04b] W. Lau and S. Jha, “Joint Path Computation algorithms for restoration networks,” *ICC*, France, 2004.
- [Leon-Garcia03] A. Leon-Garcia and L.G. Mason, “Virtual Network Resource Management for Next-Generation Networks,” *IEEE Communications Magazine*, Vol.41, No.7, July 2003.
- [Leslie96] Leslie, Michael L. Littman and Moore, “Reinforcement Learning: A Survey,” *Journal of Artificial Intelligence Research*, 237-285, 1996.
- [Li03] G. Li, D. Wang, C. Kalmanek and R. Doverspike, “Efficient Distributed Restoration Path Selection for Shared Mesh Restoration,” *IEEE/ACM Transactions on Networking*, 11(5), October 2003.
- [Li99] B. Li and K. Nahrstedt, “Dynamic reconfiguration for Complex Multimedia Applications,” in *Proceedings of IEEE International Conference on Multimedia Computing and Systems (ICMCS 99)*, Florence, Italy, June 1999.

- [Lin96] Lin, Su and Lo, "Virtual Path Management in ATM Networks," *Proceedings of ICC*, Dallas, USA, June 1996.
- [Littman93] Littman and Boyan, "A distributed reinforcement learning scheme for network routing," in *Proceedings of the First International Workshop on Applications of Neural Networks to Telecommunications*, 45-51, Hillside, New Jersey, 1993.
- [Liu01] Y. Liu, Tipper and Siripingwutikorn, "Approximating Optimal Spare Capacity Allocation by Successive Survivable Routing," in *Proceedings of IEEE INFOCOM*, Anchorage, USA, April 2001.
- [Maresca02] R. Maresca, Arienzo, M. Esposito, Romano and G. Ventre, "An Active Network approach to Virtual Private Networks," *IEEE International Symposium on Computers and Communications*, Taormina, Italy, July 2002.
- [Markopoulou04] A. Markopoulou, Iannaccone, Bhattacharyya, Chuah and Diot, "Characterization of Failures in an IP Backbone," in *Proceedings of IEEE INFOCOM*, Hong Kong, China, March 2004.
- [Marzo03] J.L. Marzo, E. Calle, C. Scoglio and T. Anjali, "QoS On-Line Routing and MPLS Multilevel Protection: a Survey," *IEEE Communication Magazine*, Vol.41 No.10, pp. 126-132, October 2003.
- [McDysan00a] David McDysan, "VPN Applications Guide," *John Wiley & Sons, Inc*, 2000.
- [McDysan00b] David McDysan, "QoS and Traffic Management in IP and ATM Networks," *Mc-GrawHill*, 2000.
- [McKenny90] P. E. McKenny, "Stochastic Fairness Queuing," *IEEE INFOCOM'90*, June 1990.
- [Mitra96] Debasis Mitra and I. Ziedins, "Virtual Partitioning by Dynamic Priorities: Fair and Efficient Resource Sharing by Several Services," *Proceedings of Zurich Seminar on Digital Communications*, LNCS, Springer, 1996.
- [Mitra97] Debasis Mitra and Ziedins, "Hierarchical Virtual Partitioning: Algorithms for Virtual Private Networking," *IEEE GLOBECOM97*.

- [Mitra99] Debasis Mitra, J.A.Morrison and K. G. Ramakrishnan, "Virtual Private Networks: Joint Resource Allocation and Routing Design," *IEEE Transactions on Networking*, 1999.
- [Modiano01] Modiano and Narula, "Survivable Routing of logical topologies in WDM networks," *IEEE INFOCOM 2001*.
- [Murch98] R. Murch and T. Johnson, "Intelligent Software Agents," *Prentice Hall PTR*, 1998.
- [Nadeau02] T. Nadeau, J. Boyle, K. Kompella, W. Townsend and Skalecki, "Protocol extensions for support of Diff-Serv-aware MPLS traffic engineering," *IETF Internet Draft*, August 2002.
- [Newman98] P. Newman, G. Minshall and T. Lyon, "IP Switching: ATM under IP," *IEEE/ACM Transactions on Networking*, April 1998.
- [Nichols98] K. Nichols, Blake, Baker and D. Black, "Definition of the Differentiated Services Field (DS field) in the IPv4 and IPv6 Headers," *RFC 2474*, December 1998.
- [Nichols99] K. Nichols, Van Jacobson and Zhang, "A Two-bit Differentiated Services Architecture for the Internet," *RFC 2638*, July 1999.
- [Patton00] Patton, Smith, D. Doss and Yurcik, "A Layered Framework for Deploying High Assurance VPNs," in *Proceedings of HASE 2000*.
- [Perguson98] Paul Ferguson and Geoff Huston, "What is a VPN?," *Cisco White Paper*, April 1998.
- [Perlmutter00] Bruce Perlmutter, "Virtual Private Networks: A View from Trenches," *PH PTR*, 2000.
- [Pitsillides02] A. Pitsillides, C. Patichis, A. Sekercioglu, Stylianou and Vasilakos, "Bandwidth Allocation for VP (BAVP): An Investigation of Performance of Classical Constrained and Evolutionary Programming Optimization Techniques," *Computer Communications* 25(2002) 1443-1453.
- [Pitts00] J. M. Pitts and Schormans, "Introduction to ATM and IP Design and Performance," *J. Wiley & Sons*, 2000.
- [Psounis99] K. Psounis, "Active Networks: Applications, Security, Safety, and Architectures," *IEEE Communications Surveys*, First Quarter 1999.
- [Quiggle01] A. Quiggle, "Implementing CISCO VPNs," *McGraw-Hill*, 2001.

- [Ramanathan96a] S. Ramanathan, "An algorithm for multicast tree generation in networks with asymmetric links," *IEEE INFOCOM*, 1996.
- [Ramanathan96b] S. Ramanathan, "Multicast Tree Generation in Networks with Asymmetric Links," *IEEE/ACM Transactions on Networking*, August 1996.
- [Rosen00] Rosen, Rekhter, Bogovic, et al. "BGP/MPLS VPNs," *Internet Draft* work in progress), July 2000.
- [Rosen99] Rosen, Arun Viswanathan and R. Callon, "Multiprotocol Label Switching Architecture," *Internet Draft*, August 1999.
- [Sakauchi90] H. Sakauchi, Nishimura and Hasegawa, "A Self-Healing Network with an Economical Spare-Channel Assignment," in *Proceedings of IEEE GLOBECOM*, San Diego, USA, December 1990.
- [Sato90] K. Sato, Ohta and Tokizawa, "Broad-Band ATM Network Architecture Based on Virtual Paths," *IEEE Transactions on Communications*, Vol. 38 No. 8, August 1990.
- [Sharma02] V. Sharma, Crane, S. Makam, Owens, Huang, Hellstrand, J. Weil, L. Anderson, B. Jarroussi, B. Cain, S. Civanlar and Chiu, "Framework for MPLS based recovery," *Internet Draft*, July 2002.
- [Simpson95] W. Simpson, "IP in IP Tunneling," *RFC 1853*, Oct. 1995.
- [Singhal03a] N. K. Singhal, L. H. Sahasrabudhe and B. Mukherjee, "Provisioning of Survivable Multicast Sessions Against Single Link Failures in Optical WDM Mesh Networks," *Journal of Lightwave Technology*, Vol. 21, No.11, Nov 2003.
- [Singhal03b] N. K. Singhal and B. Mukherjee, "Protecting a multicast session against single link failures in mesh networks," *IWDC*, 2003, Calcutta, INDIA.
- [Singhal03c] N. K. Singhal and B. Mukherjee, "Protecting Multicast Sessions in WDM Optical Mesh networks," *Journal of Light Wave Technology*, Vol. 21, No.4, April 2003.
- [Stern99] T. Stern and Bala, "Multiwavelength Optical Networks: A Layered approach," *Prentice Hall*, 1999.
- [Stiliadis97] D. Stiliadis and Varma, "A General Methodology for Designing Efficient Traffic Scheduling and Shaping Algorithms," *Proc. IEEE INFOCOM'97*, 1997.

- [Stoica97] Ion Stoica, Hui Zhang, et al. "A Hierarchical Fair Service Curve Algorithm for Link-Sharing, Real Time and Priority Queues," *SIGCOMM'97*, September 1997.
- [Sutton94] R. S. Sutton and Barto, "Reinforcement learning: an introduction," *MIT Press*, Chp.1-3, 1994.
- [Swallo99] G. Swallow, "MPLS Advantages for Traffic Engineering," *IEEE Communications Magazine*, 37(12), December 1999.
- [Takahashi80] Takahashi and Matsuyama, "An Approximate Solution for the Stenier Problem in Graphs," *Math Japonica*, Vol. 24, pp. 573-577, 1980.
- [Tekiner04] F. Tekiner, Ghassemlooy and Srikanth, "Comparison of the Q-Routing and Shortest Path Routing Algorithms," *PGNET04*, Liverpool, UK, June 2004.
- [Townesley99] W. Townesley, Valencia, Rubens, G. Pall, G. Zorn and B. Palter, "Layer two Tunneling Protocol (L2TP)," *RFC 2661*, August 1999.
- [Tuna04] Tuna Guven, Christopher Kommareddy, Richard La, Mark Shayman and Samrat Bhattacharjee, "Measurement Based Optimal Multi-path Routing," *IEEE INFOCOM 2004*, Hong Kong, March 2004.
- [Wang02] Wang and Nahrstedt, "Hop by Hop Routing Algorithms for Premium Class Traffic in DiffServ Networks," *IEEE INFOCOM 2002*.
- [Watkins89] Watkins and Dayan, "Q-learning," *Machine Learning*, 279-292, 1989.
- [Wu93] Wu, H. Kobrinski, Ghosal and T. Lakshman, "A service restoration time study for distributed control SONET digital cross-connect system self-healing networks," in *Proceedings of IEEE ICC*, Geneva, Switzerland, 1993.
- [Xiong99] Y. Xiong and L. Mason, "Restoration Strategies and Spare Capacity Requirements in Self-Healing ATM Networks," *IEEE/ACM Transactions on Networking*, 7(1), February 1999.
- [Yavatkar00] R. Yavatkar, R. Guerin and D. Pendarakis, "A Framework for Policy-based Admission Control," *RFC 2753*, January 2000.
- [Yahara97] Yahara and Kawamura, "Virtual Path self-healing scheme based on multi-reliability ATM network concept," *IEEE GLOBECOM'97*, November 1997.

- [Yuan01a] R. Yuan and Strayer, "Virtual Private Networks – Technologies and Solutions," *Addison Wesley*, 2001.
- [Yuan01b] Yuan, Tham and Ananda, "A Probing Approach for Effective Distributed Resource Reservation," *2nd International Workshop on QoSIP*, Milan, Italy, Springer LNCS 2601, pp. 672-688.
- [Yuan02] X. Yuan, "Heuristic Algorithms for Multi-constrained Quality of Service Routing," *IEEE Transactions on Networking*, Vol. 10, April, 2002.
- [Zeng03] Zeng and Ansari, "Toward IP Virtual Private Network Quality of Service: A Service Provider Perspective," *IEEE Communications Magazine*, April 2003.
- [Zhang93] L. Zhang, S. Deering and D. Estrin, "RSVP: A New Resource Reservation Protocol," *IEEE Network Magazine*, September 1993.
- [Zhang95] H. Zhang, "Service Disciplines for Guaranteed Performance Service in Packet Switching Networks," *Proceedings of the IEEE*, Oct. 1995.

Appendix A

IPSec VPN Implementation

This part describes an IPSec VPN testbed implementation using available Cisco 7100 series routers. The testbed over which IPSec tunnel is established is as shown in figure A.1

IPSec Overview:

IPSec is a framework of open standards developed by IETF. It provides security for transmission of sensitive information over unprotected networks such as Internet. Security can be provided at various layers, including the network layer, the transport layer and the application layer. Providing security at the Internet layer has the following advantages:

- Implemented once, in a consistent manner, it can be used in multiple applications
- Centrally-controlled access policy
- Enables multi-level layered approach to security.

IPSec is designed to provide interoperable, high quality, cryptographically based security for IPv4 and IPv6. The set of security services offered includes:

- Access Control
- Connectionless Integrity
- Data Origin Authentication
- Replay Protection (a form of partial sequence integrity)
- Confidentiality (encryption), and
- Limited Traffic Flow Confidentiality

These objectives are met through the use of two traffic security protocols,

- Authentication Header (AH)
- Encapsulating Security Payload (ESP)

Security Services Provided by IPSec:

Access Control: Access control is a security service that prevents unauthorized use of a resource, including the prevention of use of a resource in an unauthorized manner. In the IPSec context, the resource to which access is being controlled is often:

- For a host, computing cycles or data
- For a security gateway, a network behind the gateway or bandwidth on that network (The term "security gateway" refers to an intermediate system that implements IPSec protocols. For example, a router or a firewall implementing IPSec is a security gateway.)

Connectionless Integrity: Integrity is a security service that ensures that modifications to data are detectable. Connectionless integrity is a service that detects modification of an individual IP datagram, without regard to the ordering of the datagram in a stream of traffic.

Data Origin Authentication: Authentication refers to mechanisms, which are used to verify the identity of a user. Data origin authentication is a security service that verifies the identity of the claimed source of data. This service is usually bundled with connectionless integrity service.

Replay Protection: The form of partial sequence integrity offered in IPSec is referred to as anti-replay integrity, and it detects arrival of duplicate IP datagrams.

Confidentiality: Confidentiality is the security service that protects data from unauthorized disclosure. The primary confidentiality concern in most instances is unauthorized disclosure of application level data.

Traffic Flow Confidentiality: Traffic flow confidentiality is the service that addresses disclosure of the external characteristics of communication by concealing source and destination addresses, message length, or frequency of communication. In the IPSec context, using ESP in tunnel mode, especially at a security gateway, can provide some level of traffic flow confidentiality.

IPSec Specifications:

Security Association (SA):

A Security Association (SA) is a simplex "connection" that affords security services to the traffic carried by it. Security services are afforded to an SA by the use of

AH, or ESP, but not both. If both AH and ESP protection is applied to a traffic stream, then two (or more) SAs are created to afford protection to the traffic stream. To secure typical, bi-directional communication between two hosts, or between two security gateways, two Security Associations (one in each direction) are required.

A security association is uniquely identified by a triple consisting of a *Security Parameter Index (SPI)*, *IP Destination Address* and a *security protocol (AH or ESP) identifier*.

Types of SA

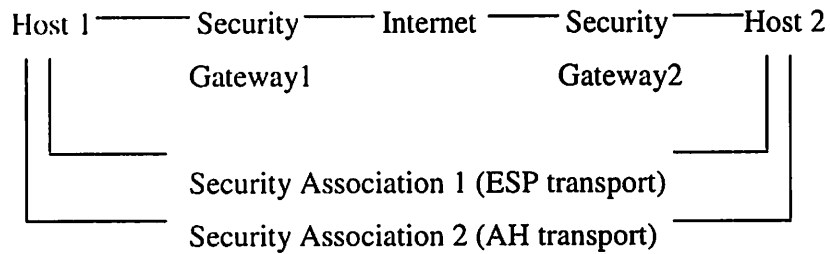
- **Transport Mode:** A transport mode SA is a security association between two hosts. In the case of ESP, a transport mode SA provides security services only for the higher layer protocols, not for the IP header or any extension headers preceding the ESP header. In the case of AH, the protection is also extended to selected portions of the IP header, selected portions of extension headers, and selected options.
- **Tunnel Mode:** A tunnel mode SA is essentially an SA applied to an IP tunnel. Whenever either end of a security association is a security gateway, the SA must be tunnel mode. Thus an SA between two security gateways is always a tunnel mode SA, as is an SA between a host and a security gateway. For a tunnel mode SA, there is an "outer" IP header that specifies the IPSec processing destination, plus an "inner" IP header that specifies the (apparently) ultimate destination for the packet. The security protocol header appears after the outer IP header, and before the inner IP header. If AH is employed in tunnel mode, portions of the outer IP header are afforded protection (as above), as well as all of the tunneled IP packet (i.e., all of the inner IP header is protected, as well as higher layer protocols). If ESP is employed, the protection is afforded only to the tunneled packet, not to the outer header.

Combining Security Associations:

Security associations may be combined in two ways:

Transport Adjacency: Transport adjacency refers to applying more than one security protocol to the same IP datagram, without invoking tunneling. This approach

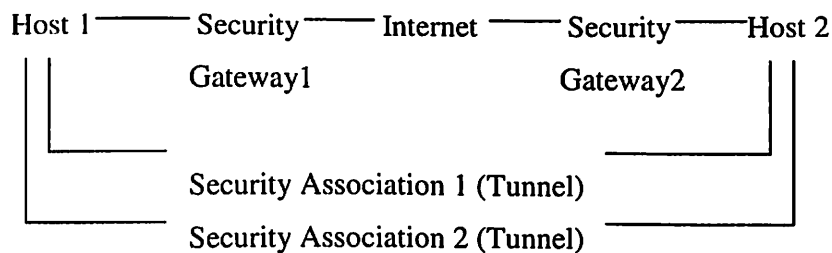
to combining AH and ESP allows for only one level of combination; further nesting yields no added benefit (assuming use of adequately strong algorithms in each protocol) since the processing is performed at one IPSec instance at the (ultimate) destination.



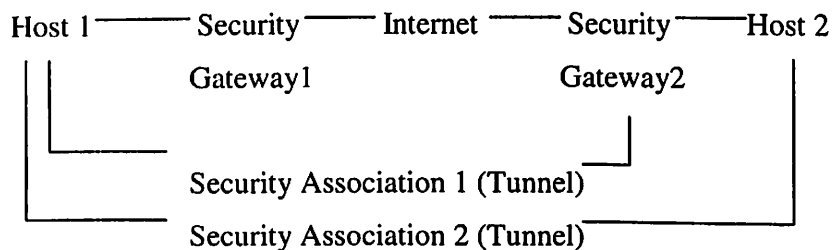
Iterated Tunneling: Iterated tunneling refers to the application of multiple layers of security protocols affected through IP tunneling. This approach allows for multiple levels of nesting, since each tunnel can originate or terminate at a different IPSec site along the path.

There are 3 basic cases of iterated tunneling.

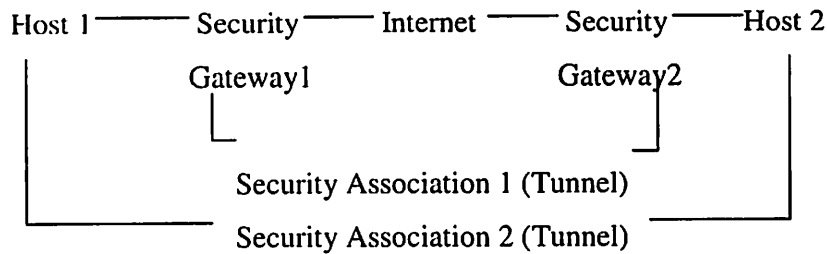
1. Both endpoints for the SAs are the same: The inner and outer tunnels could each be either AH or ESP, though it is unlikely that Host 1 would specify both to be the same. i.e., AH inside of AH or ESP inside of ESP.



2. One endpoint of the SAs is the same: The inner and outer tunnels could each be either AH or ESP.



3. Neither endpoint is the same: The inner and outer tunnels could each be either AH or ESP.



Security Association Database (SAD)

In each IPSec implementation there is a nominal Security Association Database, in which each entry defines the parameters associated with one SA. Each SA has an entry in the SAD. For outbound processing, entries are pointed to by entries in the SPD.

Security Parameter Index (SPI)

It is a 32-bit value used to distinguish among different SAs terminating at the same destination and using the same IPSec protocol.

Security Policy Database (SPD)

Security Policy Database (SPD) specifies what services are to be offered to IP datagrams and in what fashion. The SPD must be consulted during the processing of all traffic (inbound and outbound), including non-IPSec traffic. If no policy is found in the SPD that matches the packet (for either inbound or outbound traffic), the packet is discarded. The SPD has distinct entries for inbound and outbound traffic to support this.

The SPD is used to control the flow of all traffic through an IPSec system, including security and key management traffic (e.g., ISAKMP) from/to entities behind a security gateway.

Authentication Header:

The IP Authentication Header (AH) provides the following services:

- Connectionless integrity
- Data origin authentication for IP datagrams
- Replay protection

AH provides authentication for as much of the IP header as possible, as well as for upper level protocol data. However, some IP header fields may change in transit and the value of these fields, when the packet arrives at the receiver, may not be predictable by the sender. The values of such fields cannot be protected by AH. Thus the protection provided to the IP header by AH is somewhat piecemeal. AH may be applied alone, in combination with the IP Encapsulating Security Payload, or in a nested fashion through the use of tunnel mode.

Authentication Header Format:

0	1	2	3	4
Next Header	Payload Len	RESERVED		
Security Parameters Index (SPI)				
Sequence Number Field				
Authentication Data (variable)				

Next Header: The Next Header is an 8-bit field that identifies the type of the next payload after the Authentication Header.

Payload Length: This 8-bit field specifies the length of AH in 32-bit words.

Reserved: This 16-bit field is reserved for future use. It must be set to "zero."

Security Parameters Index (SPI): The SPI is an arbitrary 32-bit value that, in combination with the destination IP address and security protocol (AH), uniquely identifies the Security Association for this datagram.

Sequence Number: This unsigned 32-bit field contains a monotonically increasing counter value. It is mandatory and is always present even if the receiver does not elect to enable the anti-replay service for a specific SA. Processing of the Sequence Number field is at the discretion of the receiver, i.e., the sender must always transmit this field, but the receiver need not act upon it.

Authentication Data: This is a variable-length field that contains the Integrity Check Value (ICV) for this packet. The field must be an integral multiple of 32 bits in length and may include explicit padding to ensure that the length of the AH header is an integral multiple of 32 bits for IPv4 or 64 bits for IPv6.

The authentication algorithm employed for the ICV computation is one-way hash function MD5 generating 128-bit ICV key.

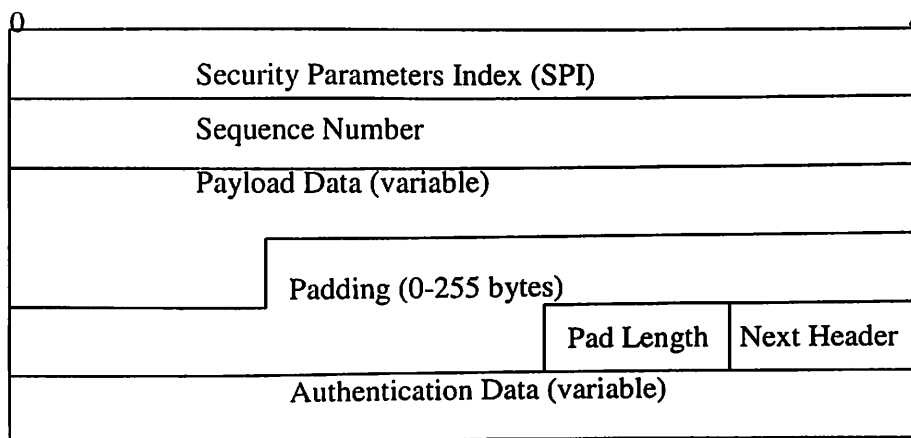
Encapsulating Security Payload:

ESP Header provides the following services:

- Confidentiality,
- Data origin authentication,
- Connectionless integrity,
- Anti-replay service, and
- Limited traffic flow confidentiality (in tunnel mode only)

ESP may be applied alone, in combination with the IP Authentication Header (AH), or in a nested fashion, e.g., through the use of tunnel mode. Security services can be provided between a pair of communicating hosts, between a pair of communicating security gateways, or between a security gateway and a host.

Encapsulating Security Payload Packet Format:



Security Parameters Index: Same as defined in AH

Sequence Number: Same as defined in AH

Payload Data: Payload Data is a variable-length field containing data described by the Next Header field.

Padding: Following factors motivate use of the Padding field.

- If an encryption algorithm is employed that requires the plaintext to be a multiple of some number of bytes, e.g., the block size of a block cipher, the Padding field is used to fill the plaintext (consisting of the Payload Data, Pad Length and Next Header fields, as well as the Padding) to the size required by the algorithm.
- Padding also may be required, irrespective of encryption algorithm requirements, to ensure that the resulting cipher text terminates on a 4-byte boundary. Specifically, the Pad Length and Next Header fields must be right aligned within a 4-byte word, as illustrated in the ESP packet format figure above, to ensure that the Authentication Data field (if present) is aligned on a 4-byte boundary.
- Padding beyond that required for the algorithm or alignment reasons cited above, may be used to conceal the actual length of the payload, in support of (partial) traffic flow confidentiality. However, inclusion of such additional padding has adverse bandwidth implications and thus its use should be undertaken with care.

Pad Length: The Pad Length field indicates the number of pad bytes immediately preceding it.

Next Header: The Next Header is an 8-bit field that identifies the type of data contained in the Payload Data field.

Authentication Data: The Authentication Data is a variable-length field containing an Integrity Check Value (ICV) computed over the ESP packet minus the Authentication Data. The Authentication Data field is optional, and is included only if the authentication service has been selected for the SA

The encryption algorithm employed is 32-bit RSA algorithm.

If authentication is selected along with encryption, encryption is performed first, before the authentication, and the encryption does not encompass the Authentication Data field. This order of processing facilitates rapid detection and rejection of replayed or bogus packets by the receiver, prior to decrypting the packet, hence potentially reducing the impact of denial of service attacks. It also allows for the possibility of parallel processing of packets at the receiver, i.e., decryption can take place in parallel with authentication.

Features of Cisco 7100 Series Routers:

The Cisco 7100 series router is a single box solution, providing VPN services along with integrated firewall and QoS features. It also supports multicast and multiprotocol traffic, and has a rich routing functionality. One more property of this router, where it scores over others is its high uptime, which is facilitated by high component redundancy. Also, the router performs constant monitoring of the environment, and takes adequate measures in case of adverse conditions like excessive heat. For dynamic allocation of bandwidth, the router uses NBAR (Network Based Application Recognition). This provides for efficient use of the available bandwidth. The router also performs Policing and Trafficking operations. The 7100 series router offers high scalability features, wherein it can support a maximum of 140 Mbps 3DES IPsec throughput for 3000 tunnels. The common architectural features of the 7100 series are:

- ❑ Two Fixed LAN Ports – Each of these ports support 10BaseT/100BaseTX Ethernet/Fast Ethernet
- ❑ Modular Adapter Slot–Supports Online Insertion and Removal. This allows a user to add, replace, or remove a modular port adapter without interrupting the system.
- ❑ Console & Auxiliary ports - For local & remote administering respectively.

The comprehensive VPN features offered are:

- ❑ Tunneling: IPsec, GRE, L2TP, PPTP, L2F
- ❑ Encryption: IPsec DES/3DES, MPPE
- ❑ Firewall & Intrusion detection: Cisco IOS Firewall / PIX Firewall
- ❑ Packet Authentication: MD5, SHA, User Authentication – RADIUS, CA

The testbed implemented is as shown below (Figure A.1). It includes formation of IPsec tunnel using 7100 series routers and installation of CA server. We used the components in the testbed for the following purposes:

- 1: IPSEC initiator: Initiates FTP/Telnet to 6.
- 2: Tunnel initiator for IPSEC tunnels.
- 3: 7120 to have NAS connectivity in future.
- 4: 7120 to separate OSPF/EIGRP regions.
- 5: VPN gateway to terminate all site-to-site (Intranet VPN) connections.

6: Windows 2000 CA server: Certificate/RSA verification, Terminator of FTP/telnet request from 1 or 2.

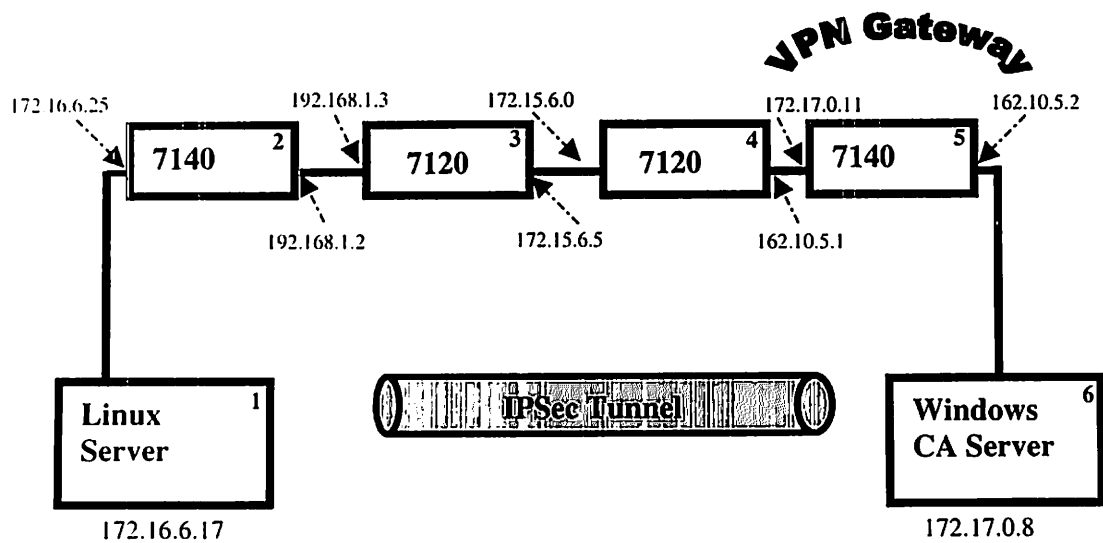


Figure A.1 Generic Illustration of the Configured IPSec Tunnel

Configuring IPSec with IKE:

The following steps cover minimal IPSec configuration where the IPSec security associations will be established via IKE.

Step 1: Enable the debug crypto ipsec command to capture important IPSec related messages that only display when this command is enabled.

debug crypto ipsec

Step 2: Create an access list to define the traffic to protect.

access-list access-list-name {deny | permit} ip source source-netmask destination destination-netmask

Step 3: Configure a transform set that defines how the traffic will be protected. Multiple transform sets can be configured, and then one or more of these transform sets can be specified in a crypto map entry.

crypto ipsec transform-set transform-set-name transform1 [transform2, transform3]

Step 4: Create a crypto map entry by performing the following steps.

- a. IPsec ISAKMP mode:

crypto map map-name seq-num ipsec-isakmp

- b. Assign an access list to a crypto map entry.

crypto map map-name seq-num match address access-list-name

- c. Specify the peer to which the IPsec protected traffic can be forwarded.

crypto map map-name seq-num set peer ip-address

- d. Specify which transform sets are allowed for this crypto map entry. List multiple transform sets in order of priority. Up to six transform sets can be specified.

crypto map map-name seq-num set transform-set transform-set-name1 [transform-set-name2, transform-set-name6]

- e. (Optional) SA lifetimes for the crypto map entry are specified, if the security associations for this entry are to be negotiated using different IPsec security association lifetimes other than the global lifetimes.

crypto map map-name seq-name set security-association lifetime {seconds seconds | kilobytes kilobytes}

Step 5: (Optional) Create a crypto dynamic map entry by performing the following steps:

- a. (Optional) Assign an access list to a dynamic crypto map entry, which determines which traffic should be protected and which not.

crypto dynamic-map dynamic-map-name dynamic-seq-num match address access-list-name

- b. (Optional) Specify the peer to which the IPsec protected traffic can be forwarded. This is rarely configured in dynamic crypto map entries because dynamic crypto map entries are often used for unknown peers.

crypto dynamic-map dynamic-map-name dynamic-seq-num set peer ip-address

- c. Specify which transform sets are allowed for this dynamic crypto map entry. List multiple transform sets in order of priority.

crypto dynamic-map dynamic-map-name dynamic-seq-num set transform-set transform-set-name1 [transform-set-name2, transform-set-name9]

- d. (Optional) Specify SA lifetime for the crypto dynamic map entry, if the security associations for this entry are to be negotiated using different IPSec security association lifetimes other than the global lifetimes.

crypto dynamic-map dynamic-map-name dynamic-seq-num set security-association lifetime {seconds seconds | kilobytes kilobytes}

- e. Add the dynamic crypto map set into a static crypto map set. Be sure to set the crypto map entries referencing dynamic maps to be the lowest priority entries (highest sequence numbers) in a crypto map set.

crypto map map-name seq-num ipsec-isakmp dynamic dynamic-map-name

Step 6: Apply a crypto map set to an interface on which the IPSec traffic will be evaluated.

crypto map map-name interface interface-name

IPSec with Certificate Authority (CA) Server:

Certification Authority (CAs) are responsible for managing certificate requests and issuing certificates to participating IPSec network peers. These services provide centralized key management for the participating peers and thus simplify the administration of IPSec network devices.

Without a CA, if IPSec services such as encryption are to be enabled between two peers, then one must first ensure that each peer has the other's key such as a pre-shared key. This requires that the key be manually configured in the two peers. However, every time a new peer is added to the IPSec network, keys need to be configured between the new peer and each of the existing peers. Consequently, more are the keys, more becomes the key administration. To relieve from this, CAs are used along with IPSec. In an IPSec with a CA, there is no need to configure keys between all the encrypting devices. Instead, each participating device needs to be dynamically enrolled with the CA, requesting a

certificate for the device. When this has been accomplished, each participating device can dynamically authenticate all of the other participating peers.

To add a new IPsec device to the network, only that the new device needs to be configured to request a certificate from the CA, instead of making multiple key configurations with all the other existing IPsec peers.

When two IPsec peers want to exchange IPsec-protected traffic passing between them, they must first authenticate each other. The authentication is done with IKE. Without a CA, a device authenticates itself to the remote peer using either RSA-encrypted nonces or pre-shared keys. With a CA, a peer authenticates itself to the remote peer by sending a certificate to the remote peer and performing some public key cryptography. Each peer must send its own unique certificate which was issued and validated by the CA. This process works because each peers' certificate encapsulates the peer's public key, each certificate is authenticated by the CA, and all participating peers recognize the CA as an authenticating authority. This is called IKE with an RSA signature.

The peer can continue sending its own certificate for multiple IPsec sessions, and to multiple IPsec peers until the certificate expires. CAs can also revoke certificates for peers that will no longer participate in IPsec. Revoked certificates are listed in a Certificate Revocation List (CRL), which each peer may check before accepting another peers' certificate.

Enrollment of a 7100 VPN Gateway with a CA server:

The following steps were carried out in setting up the CA server and enrolling the 7100 series with it.

1. The CA server and name server were set up
2. The CA identity on the VPN Gateway was created and enrolled with the CA
3. The certificate authority was authenticated by obtaining its public key and certificate
4. RSA keys were generated on the Gateways

5. The VPN Gateway was enrolled with the CA server by sending a certificate request

These above steps are described in detail as below:

1. Windows 2000 certificate authority should be installed from the additional component list.
2. mscep (Microsoft Cisco enrollment protocol).dll should be installed as an add-on to CA for supporting enrollment of Cisco devices.
3. LDAP and HTTP services should be running and the url for enrolling would be `http://<ca-name>/certsrv/mscep/mscep.dll`
4. mscep passphrase can be disabled for better facility on the first installation screen of msceo.dll. It avoids the necessity to enter one time password needed in case of its check. This password is to be entered during the enrollment process.

Declaring a Certification Authority:

A Certification Authority should be declared for use by the router. To declare a CA, the following tasks need to be performed in global configuration mode:

1. Declare a CA (*crypto ca identity* name)
2. Specify the url of the CA that should include any non-standard cgi-bin script location (*enrollment url url*)
3. If the CA provides an RA and supports the LDAP, then specify the location of LDAP (*query url url*)
4. (Optional) Specify a retry period (*enrollment retry-period minutes*)
5. (Optional) Specify how many times the router will continue to send unsuccessful certificates before giving up (*enrollment retry-count count*)
6. (Optional) Specify that other peers' certificates can still be accepted by the router even if the appropriate CRL is not accessible to the router (*crl optional*)
7. Exit from ca-identity configuration mode (*exit*)

Authenticating the CA:

The router needs to authenticate the CA. It does this by obtaining the CA's self-signed certificate that contains the CA's public key. As the CA's certificate is self signed, the CA's public key should be manually authenticated by contacting the CA

administrator when this step is performed. To get the CA's public key, the following steps need to be performed in global configuration mode.

Step 1: Get the CA's public key. Use the same name that was used when declaring the CA with the *crypto ca identity* command. (*crypto ca authenticate name*)

Step 2: Generate an RSA key pair. The usage-keys keyword is used to specify special usage keys instead of general purpose keys. (*crypto key generate rsa [usage-keys]*)

Step 3: Request certificates for all of the RSA key pairs. (*enroll name*)

By carrying out the above steps, *domino.bits-pilani.ac.in* was configured as the CA server and 7140-2T3 and 7140-2FE were configured to enroll with the CA server and obtain their certificates.

The Windows telnet server and the Linux telnet client were able to communicate successfully with each other through the IPsec channel created in the testbed (Figure B.2) by authenticating themselves by certificate exchange. The configuration steps carried over the 7100 series routers are as given below:

Configuration of IPsec in Cisco 7100 Routers:

Configuration of 7140-2T3 (esp-des) mode transport

```
Running Configuration
Current configuration : 2029 bytes
!
version 12.1
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname 7140-2T3
!
boot system disk1:c7100-is56i-mz.121-9.bin
enable secret 5 $1$UY5W$/z6RTT9stj1w3rV7wkAAN1
!
ip subnet-zero
ip cef
ip domain-name bits-pilani.ac.in
!
crypto ipsec transform-set 7140-2T3-trans esp-des
mode transport
```

```

!
crypto map 7140-2T3-manualmap 35 ipsec-manual
  set peer 192.168.1.3
  set session-key inbound esp 259 cipher 123451234512345123456
  set session-key outbound esp 258 cipher 12341234123412341234123412341234
  set transform-set 7140-2T3-trans
  match address 105
!
!
interface FastEthernet0/0
  ip address 192.168.1.2 255.255.255.0
  duplex auto
  speed auto
  crypto map 7140-2T3-manualmap
!
interface FastEthernet0/1
  ip address 172.16.6.2 255.255.0.0
  duplex auto
  speed auto
!
!
ip classless
ip route 172.17.0.0 255.255.0.0 192.168.1.3
no ip http server
!
access-list 103 permit ip 172.16.0.0 0.0.255.255 172.17.0.0 0.0.255.255
access-list 104 permit ip 172.17.0.0 0.0.255.255 172.16.0.0 0.0.255.255
access-list 105 permit ip host 172.16.6.17 host 172.17.0.8
!
line con 0
line aux 0
line vty 0 4
  password 7140-2T3
  login
!
end
!

```

7140-2T3#sh crypto map

```

Crypto Map "7140-2T3-manualmap" 35 ipsec-manual
  Crypto Engine = (0)
  Peer = 192.168.1.3
  Extended IP access list 105
    access-list 105 permit ip host 172.16.6.17 host 172.17.0.8
  Current peer: 192.168.1.3
  Transform sets = {7140-2T3-trans,}
  Inbound esp spi:259,
    cipher key:12345123451234512345123456
    auth_key:,
  Inbound ah spi:0,
    key:,
  Outbound esp spi: 258
    cipher key: 123412341234123412341234123412341234
    auth key:,
  Outbound ah spi:0,
    key:,

```

Interfaces using crypto map 7140-2T3-manualmap:
FastEthernet0/0

7140-2T3#sh ip route

Codes: C-connected, S-static, I-IGRP, R-RIP, M-mobile, B-BGP, D-EIGRP, EX-EIGRP external,
O-OSPF, IA-OSPF inter area, N1-OSPF NSSA external type 1, N2-OSPF NSSA external type 2,
E1-OSPF external type 1, E2-OSPF external type 2, E-EGP, i-IS-IS, L1-IS-IS
level-1, L2-IS-IS level-2, ia -IS-IS inter area, *-candidate default, U-per-user static route,
o-ODR, P-periodic downloaded static route

Gateway of last resort is not set

S 172.17.0.0/16 [1/0] via 192.168.1.3
C 172.16.0.0/16 is directly connected, FastEthernet0/1
C 192.168.1.0/24 is directly connected, FastEthernet0/0

Configuration of 7140-2FE (esp-des) mode: transport

7140-2FE# sh crypto map

Crypto Map "7140-2FE-manualmap" 35 ipsec-manual
Crypto Engine = (0)
Peer = 192.168.1.2
Extended IP access list 105
access-list 105 permit ip host 172.17.0.8 host 172.16.6.17
Current peer: 192.168.1.2
Transform sets = {7140-2FE-trans,}
Inbound esp spi:258,
cipher key: 12341234123412341234123412341234123412341234
auth_key:.
Inbound ah spi:0,
key:.
Outbound esp spi: 259
cipher key: 12345123451234512345123456
auth key:.
Outbound ah spi:0,
key:.
Interfaces using crypto map 7140-2FE-manualmap:
FastEthernet0/1

7140-2FE#sh ip route

Codes: C-connected, S-static, I-IGRP, R-RIP, M-mobile, B-BGP, D-EIGRP, EX-EIGRP external,
O-OSPF, IA-OSPF inter area, N1-OSPF NSSA external type 1, N2-OSPF NSSA external type 2,
E1-OSPF external type 1, E2-OSPF external type 2, E-EGP, i-IS-IS, L1-IS-IS
level-1, L2-IS-IS level-2, ia -IS-IS inter area, *-candidate default, U-per-user static route,
o-ODR, P-periodic downloaded static route

Gateway of last resort is not set

S 172.17.0.0/16 is directly connected, FastEthernet0/0
C 172.16.0.0/16[1/0] via 192.168.1.2
C 192.168.1.0/24 is directly connected, FastEthernet0/1

!
7140-2FE#sh crypto ipsec transform-set
Transform set 7140-2FE-trans: {esp-des}

will negotiate = {Transport, },
!

7140-2FE#sh cdp neighbors detail

Device ID:7140-2T3.bits-pilani.ac.in
Entry address(es):
 IP address: 192.168.1.2
Platform: cisco 7140-2T3, Capabilities: Router
Interface: FastEthernet0/1, Port ID (outgoing port): FastEthernet0/0
Holdtime : 157 sec

Version:
Cisco Internetwork Operating System Software
IOS (tm) EGR Software (C7100-IS56I-M), Version 12.1 (9), RELEASE SOFTWARE (fc1)
Copyright (c) 1886-2001 by cisco Systems, Inc.
Compiled Wed 13-Jun-01 19:26 by kellythw

advertisement version:2
Duplex: full

Configuration of 7140-2FE (esp-des & ah-sha) mode: tunnel

```
crypto ipsec transform-set 7140-2FE-man ah-sha-hmac esp-des
!  
crypto map 7140-2FE-manualmap 35 ipsec-manual  
set peer 192.168.1.2  
set session-key inbound esp 258 cipher 12341234123412341234123412341234  
set session-key outbound esp 259 cipher 1234512345123451234512345123456  
set session-key inbound ah 261  
1234567890123456789012345678901234567890123456789012345678901234567890  
set session-key outbound ah 260  
123412341234123412341234123412341234123412341234123412341234123412341234  
set transform-set 7140-2FE-man  
match address 105  
!
```

7140-2FE#sh crypto map

```
Crypto Map "7140-2FE-manualmap" 35 ipsec-manual  
Crypto Engine = (0)  
Peer = 192.168.1.2  
Extended IP access list 105  
  access-list 105 permit ip host 172.17.0.8 host 172.16.6.17  
Current peer: 192.168.1.2  
Transform sets = {7140-2FE-man,}  
Inbound esp spi: 258,  
  cipher key: 123412341234123412341234123412341234  
  auth_key:.  
Inbound ah spi:261,  
  key: 123456789012345678901234567890123456789012345678901234567890  
Outbound esp spi: 259  
  cipher key: 12345123451234512345123456
```


Device ID	Local Intrfce	Holdtme	Capability Platform	Port ID
7140-2FE	bits-pilFas 0/0	171	R	7120-2FE Fas
0/1				

7140-2FE using isakmp and esp-des, esp-md5-hmac, ah-md5-hmac)

```

crypto ipsec transform-set 7140-2FE-trans ah-md5-hmac esp-des esp-md5-hmac
!
crypto key pubkey-chain rsa
  named-key 7140-2T3.bits-pilani.ac.in
  address 192.168.1.2
  key-string
    305C300D 06092A86 4886F70D 01010105 00034B00 30480241 00A98DB3 B93D6EFC
    A6D71A51 10546F97 3292846E 5E420B2B 10F9C711 D5887E01 3B72C8A6 2A7C5848
    FE7351F3 90D41551 4D28A480 9CAD2B0B 93D8F808 BA1C50CA C7020301 0001
  quit
!
crypto map 7140-2FE-isakmp 36 ipsec-isakmp
set peer 192.168.1.2
set transform-set 7140-2FE-trans
match address 105
!

```

7140-2FE#sh crypto map

```

Crypto Map "7140-2FE-isakmp" 36 ipsec-isakmp
Crypto Engine = (0)
Peer = 192.168.1.2
Extended IP access list 105
  access-list 105 permit ip host 172.17.0.8 host 172.16.6.17
Current peer: 192.168.1.2
Security association lifetime: 4608000 kilobytes/3600 seconds
PFS (Y/N): N
Transform sets = {7140-2FE-trans,}
Interfaces using crypto map 7140-2FE-isakmp:
  FastEthernet0/1

```

7140-2FE#sh crypto isakmp policy

```

Protection suit of priority 1
  encryption algorithm: DES- Data Encryption Standard (56 bit keys)
  hash algorithm:      Message Digest 5
  authentication method: Rivest-Shamir-Adleman Encryption
  Diffie-Hellman group: #1 (768 bit)
  lifetime:           86400 seconds, no volume limit
Default Protection suite
  encryption algorithm: DES- Data Encryption Standard (56 bit keys)
  hash algorithm:      Secure Hash Standard
  authentication method: Rivest-Shamir-Adleman Signature
  Diffie-Hellman group: #1 (768 bit)
  lifetime:           86400 seconds, no volume limit

```

7140-2FE#sh crypto ipsec transform-set

```

Transform set 7140-2FE-trans: {ah-md5-hmac}
  will negotiate = {Tunnel,},
  {esp-des esp-md5-hmac}
  will negotiate = {Tunnel,},

```

7140-2T3 using isakmp and esp-des, esp-md5-hmac, ah-md5-hmac)

```
crypto key pubkey-chain rsa
  named-key 7140-2FE.bits-pilani.ac.in
  address 192.168.1.3
  key-string
    305C300D 06092A86 4886F70D 01010105 00034B00 30480241 00A98DB3 B93D6EFC
    A6D71A51 10546F97 3292846E 5E420B2B 10F9C711 D5887E01 3B72C8A6 2A7C5848
    FE7351F3 90D41551 4D28A480 9CAD2B0B 93D8F808 BA1C50CA C7020301 0001
  quit
!
crypto map 7140-2T3-isakmp 36 ipsec-isakmp
set peer 192.168.1.3
set transform-set 7140-2T3-trans
  match address 105
!
```

7140-2T3#sh crypto ipsec transform-set

```
Transform set 7140-2T3-trans: {ah-md5-hmac}
  will negotiate = {Tunnel,},
  {esp-des esp-md5-hmac}
  will negotiate = {Tunnel,},
```

7140-2T3#sh crypto map

```
Crypto Map "7140-2T3-isakmp" 36 ipsec-isakmp
Crypto Engine = (0)
Peer = 192.168.1.3
Extended IP access list 105
  access-list 105 permit ip host 172.16.6.17 host 172.17.0.8
Current peer: 192.168.1.3
Security association lifetime: 4608000 kilobytes/3600 seconds
PFS (Y/N): N
Transform sets = {7140-2T3-trans,}
Interfaces using crypto map 7140-2T3-isakmp:
  FastEthernet0/0
```

7140-2T3#sh crypto isakmp policy 1

```
Protection suit of priority 1
  encryption algorithm: DES- Data Encryption Standard (56 bit keys)
  hash algorithm: Message Digest 5
  authentication method: Rivest-Shamir-Adleman Encryption
  Diffie-Hellman group: #1 (768 bit)
  lifetime: 86400 seconds, no volume limit
Default Protection suite
  encryption algorithm: DES- Data Encryption Standard (56 bit keys)
  hash algorithm: Secure Hash Standard
  authentication method: Rivest-Shamir-Adleman Signature
  Diffie-Hellman group: #1 (768 bit)
  lifetime: 86400 seconds, no volume limit
```

7140-2FE using isakmp with tunnel interface (ah-md5-hmac, esp-des, esp-md5-hmac)

```
sh run
Building configuration...
Current configuration: 1676 bytes
!
version 12.1
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname 7140-2FE
!
boot system flash disk0:c7100-is56i-mz.121-9.bin
enable secret 5 $1SSvCK$ZQ4haArFS0Vul8oM3Khp90
!
ip subnet-zero
ip cef
ip domain-name bits-pilani.ac.in
!
crypto isakmp policy 1
hash md5
authentication rsa-encr
crypto isakmp key moti address 192.168.1.2
!
crypto ipsec transform-set 7140-2FE-trans ah-md5-hmac esp-des esp-md5-hmac
!
crypto key pubkey-chain rsa
  named-key 7140-2T3.bits-pilani.ac.in
  address 192.168.1.2
  key-string
    305C300D 06092A86 4886F70D 01010105 00034B00 30480241 00A98DB3 B93D6EFC
    A6D71A51 10546F97 3292846E 5E420B2B 10F9C711 D5887E01 3B72C8A6 2A7C5848
    FE7351F3 90D41551 4D28A480 9CAD2B0B 93D8F808 BA1C50CA C7020301 0001
  quit
!
crypto map 7140-2FE-isakmp 36 ipsec-isakmp
set peer 192.168.1.2
set transform-set 7140-2FE-trans
  match address 105
!
interface Tunnel1
ip address 10.0.0.2 255.0.0.0
tunnel source 192.168.1.3
tunnel destination 192.168.1.2
tunnel mode ipip
crypto map 7140-2FE-isakmp
!
interface FastEthernet0/0
ip address 172.17.0.11 255.255.0.0
duplex auto
speed auto
!

interface FastEthernet0/1
ip address 192.168.1.3 255.255.255.0
duplex auto
speed auto
!
ip classless
ip route 10.0.0.0 255.0.0.0 192.168.1.2
```



```

ip route 172.16.0.0 255.255.0.0 10.0.0.1
no ip http server
!
access-list 103 permit ip 172.17.0.0 0.0.255.255 172.16.0.0 0.0.255.255
access-list 103 permit ip 172.16.0.0 0.0.255.255 172.17.0.0 0.0.255.255
access-list 105 permit ip host 172.17.0.8 host 172.16.6.17
!
line con 0
line aux 0
line vty 0 4
    password 7140-2FE
    login
!
end

```

7140-2FE#sh crypto isakmp policy

```

Protection suit of priority 1
    encryption algorithm: DES- Data Encryption Standard (56 bit keys)
    hash algorithm:          Message Digest 5
    authentication method:   Rivest-Shamir-Adleman Encryption
    Diffie-Hellman group: #1 (768 bit)
    lifetime:                86400 seconds, no volume limit
Default Protection suite
    encryption algorithm: DES- Data Encryption Standard (56 bit keys)
    hash algorithm:        Secure Hash Standard
    authentication method: Rivest-Shamir-Adleman Signature
    Diffie-Hellman group: #1 (768 bit)
    lifetime:              86400 seconds, no volume limit
!

```

7140-2FE#sh crypto map

```

Crypto Map "7140-2FE-isakmp" 36 ipsec-isakmp
Crypto Engine = (0)
Peer = 192.168.1.2
Extended IP access list 105
    access-list 105 permit ip host 172.17.0.8 host 172.16.6.17
Current peer: 192.168.1.2
Security association lifetime: 4608000 kilobytes/3600 seconds
PFS (Y/N): N
Transform sets = {7140-2FE-trans,}
Interfaces using crypto map 7140-2FE-isakmp:
    Tunnel1
!

```

7140-2FE#sh ip route

```

Codes: C-connected, S-static, I-IGRP, R-RIP, M-mobile, B-BGP, D-EIGRP, EX-EIGRP external,
O-OSPF, IA-OSPF inter area, N1-OSPF NSSA external type 1, N2-OSPF NSSA external type 2,
E1-OSPF external type 1, E2-OSPF external type 2, E-EGP, i-IS-IS, L1-IS-IS
level-1, L2-IS-IS level-2, ia -IS-IS inter area, *-candidate default, U-per-user static route,
o-ODR, P-periodic downloaded static route

```

Gateway of last resort is not set

```

C 172.17.0.0/16 is directly connected, FastEthernet0/0
S 172.16.0.0/16[1/0] via 10.0.0.1
C 10.0.0.0/8 is directly connected, Tunnel1
C 192.168.1.0/24 is directly connected, FastEthernet0/1
!

```

7140-2FE#sh cdp neighbors

Capability Codes: R-Router, T-Trans Bridge, B-Source Route Bridge, S-Switch,

H-Host, I-IGMP, r-Repeater

Device ID	Local Intrfce	Holdtme	Capability	Platform	Port ID
7140-2T3.bits-pilFas	0/1	179		R	7140-2T3 Fas

0/0

7140-2FE#sh crypto ipsec transform-set?

WORD Transform set tag

| Output modifiers

<cr>

7140-2FE#sh crypto ipsec transform-set

Transform set 7140-2FE-trans: {ah-md5-hmac}

will negotiate = {Tunnel.},

{esp-des esp-md5-hmac}

will negotiate = {Tunnel.},

7140-2FE using isakmp with tunnel interface (ah-md5-hmac, esp-des, esp-md5-hmac)

#sh run

Building configuration...

Current configuration: 1915 bytes

!

version 12.1

service timestamps debug uptime

service timestamps log uptime

no service password-encryption

!

hostname 7140-2T3

!

boot system disk1:c7100-is56i-mz.121-9.bin

enable secret 5 \$1\$UY5W\$/z6RTT9stjlw3rV7wkAAN1

!

ip subnet-zero

ip cef

ip domain-name bits-pilani.ac.in

!

!

crypto isakmp policy 1

hash md5

authentication rsa-encr

crypto isakmp key moti address 192.168.1.3

!

!

crypto ipsec transform-set 7140-2T3-trans ah-md5-hmac esp-des esp-md5-hmac

!

crypto key pubkey-chain rsa

named-key 7140-2FE.bits-pilani.ac.in

address 192.168.1.3

key-string

305C300D 06092A86 4886F70D 01010105 00034B00 30480241 00B11C9C 66C7AFB9

2B849D71 2C3726BF 6039FDF5 16165ECB C53C3709 ACD9D68B A25213FB EE797046

A7E7AB44 F2C50A28 E8A999B8 5A452811 786AFFCB 0CBC7DAD 79020301 0001

quit

!

crypto map 7140-2T3-isakmp 36 ipsec-isakmp

set peer 192.168.1.3

set transform-set 7140-2T3-trans

```

    match address 105
    !
    !
    !
interface Tunnel0
ip address 10.0.0.1 255.0.0.0
tunnel source 192.168.1.2
tunnel destination 192.168.1.3
tunnel mode ipip
crypto map 7140-2T3-isakmp
!
interface FastEthernet0/0
ip address 192.168.1.2 255.255.255.0
duplex auto
speed auto
!
interface FastEthernet0/1
ip address 172.16.6.2 255.255.0.0
duplex auto
speed auto
!
interface Serial1/0
no ip address
shutdown
framing c-bit
cablelength 10
dsu bandwidth 44210
serial restart-delay 0
!
interface Serial1/1
no ip address
shutdown
framing c-bit
cablelength 10
dsu bandwidth 44210
serial restart-delay 0
!
ip classless
ip route 10.0.0.0 255.0.0.0 192.168.1.3
ip route 172.17.0.0 255.255.0.0 10.0.0.2
no ip http server
!
access-list 103 permit ip 172.16.0.0 0.0.255.255 172.17.0.0 0.0.255.255
access-list 104 permit ip 172.17.0.0 0.0.255.255 172.16.0.0 0.0.255.255
access-list 105 permit ip host 172.16.6.17 host 172.17.0.8
!
line con 0
line aux 0
line vty 0 4
password 7140-2T3
login
!
end

```

7140-2T3#sh ip route

Codes: C-connected, S-static, I-IGRP, R-RIP, M-mobile, B-BGP, D-EIGRP, EX-EIGRP external,
O-OSPF, IA-OSPF inter area, N1-OSPF NSSA external type 1, N2-OSPF NSSA external type 2,
E1-OSPF external type 1, E2-OSPF external type 2, E-EGP, i-IS-IS, L1-IS-IS
level-1, L2-IS-IS level-2, ia -IS-IS inter area, *-candidate default, U-per-user static route,

o-ODR, P-periodic downloaded static route

Gateway of last resort is not set

```
S 172.17.0.0/16 [1/0] via 10.0.0.2
C 172.16.0.0/16 is directly connected, FastEthernet0/1
C 10.0.0.0/8 is directly connected, Tunnel0
C 192.168.1.0/24 is directly connected, FastEthernet0/0
!
```

7140-2T3#sh crypto isakmp policy

Protection suit of priority 1

```
encryption algorithm: DES- Data Encryption Standard (56 bit keys)
hash algorithm:      Message Digest 5
authentication method: Rivest-Shamir-Adleman Encryption
Diffie-Hellman group: #1 (768 bit)
lifetime:           86400 seconds, no volume limit
```

Default Protection suite

```
encryption algorithm: DES- Data Encryption Standard (56 bit keys)
hash algorithm:      Secure Hash Standard
authentication method: Rivest-Shamir-Adleman Signature
Diffie-Hellman group: #1 (768 bit)
lifetime:           86400 seconds, no volume limit
```

Crypto Map "7140-2T3-isakmp" 36 ipsec-isakmp

```
Crypto Engine = (0)
Peer = 192.168.1.3
Extended IP access list 105
  access-list 105 permit ip host 172.16.6.17 host 172.17.0.8
Current peer: 192.168.1.3
Security association lifetime: 4608000 kilobytes/3600 seconds
PFS (Y/N): N
Transform sets = {7140-2T3-trans,}
Interfaces using crypto map 7140-2T3-isakmp:
  Tunnel0
```

7140-2T3#sh crypto ipsec transform-set

```
Transform set 7140-2T3-trans: {ah-md5-hmac}
will negotiate = {Tunnel,},
{esp-des esp-md5-hmac}
will negotiate = {Tunnel,},
```

7140-2T3#sh crypto ipsec transform-set

```
sa          IPSEC SA table
security-association-lifetime Show this router's security association
lifetime info
transform-set      Crypto transform sets
```

7140-2T3#sh crypto ipsec security-association-lifetime

```
Security association lifetime: 4608000 kilobytes/3600 seconds
```

Appendix B

Traffic Engineering in MPLS VPN

MPLS is rapidly emerging as a core technology for next-generation networks, in particular optical networks. It also provides a flexible and elegant VPN solution based on the use of LSP tunnels to encapsulate VPN data. VPNs give considerable added value to the customer over and above a basic best effort IP service, so this represents a major revenue-generating opportunity for Service Providers. In this part an implementation of MPLS VPN traffic engineering is described.

Elements of an MPLS VPN solution

Let us consider how MPLS can provide a VPN solution by examining how it would work at several different levels. We start with the data forwarding mechanics and work our way up to the network management considerations. Different implementation models for MPLS-based VPNs imply different interactions between these elements of a VPN solution.

LSP Tunnels

The basis of any MPLS solution for VPNs is the use of LSP tunnels for forwarding data between SP edge routers that border on a given VPN. By labeling the VPN data as it enters such a tunnel, the LSR neatly segregates the VPN flows from the rest of the data flowing in the SP backbone.

- Multiple protocols on the VPN can be encapsulated by the tunnel ingress LSR since the data traversing an LSP tunnel is opaque to intermediate routers within the SP backbone.

- Multiplexing of traffic for different VPNs onto shared backbone links can be achieved by using separate LSP tunnels (and hence separate labels) for each data source.

- Authentication of the LSP tunnel endpoint is provided by the label distribution protocols.

- QoS for the VPN data can be assured by reserving network resources for the LSP tunnels. MPLS supports both Intserv and Diffserv.

- Protection switching and automatic re-routing of LSP tunnels ensures that failure of a link or router that affects a VPN can be corrected without management intervention. These protection mechanisms operate at several different levels, including refresh/keep-alive messages on a hop-by-hop basis within the label distribution protocols, re-routing of LSP tunnels, pre-provisioning of alternative routes, and wavelength failure detection and management for optical networks.

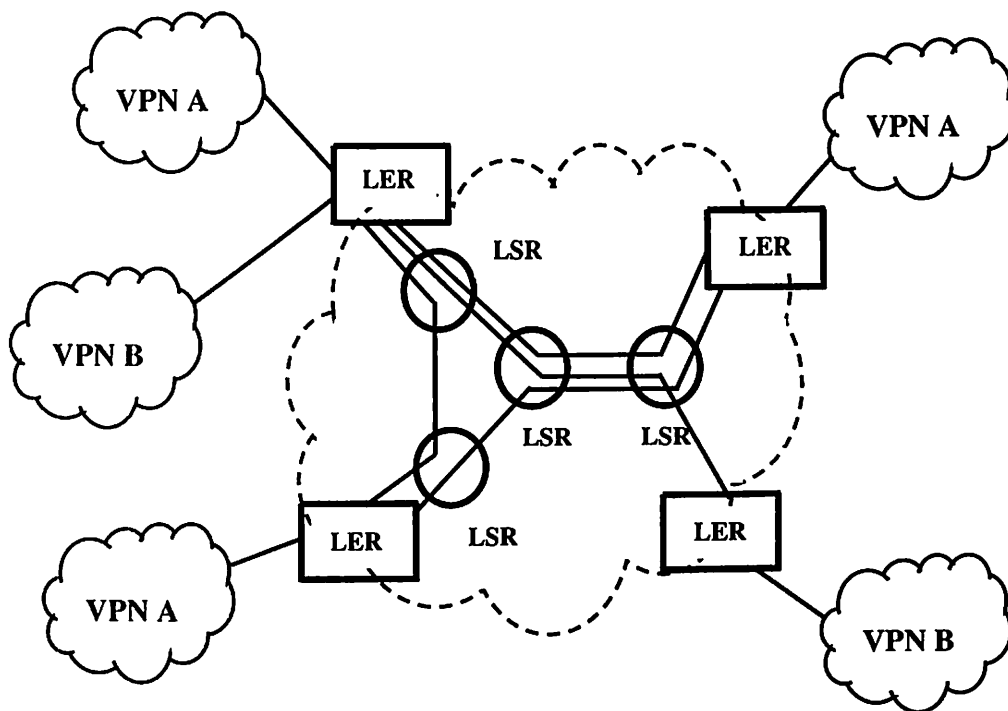


Figure B.1 VPN Connectivity Using LSP Tunnels

Figure B.1 shows simple interconnection between five VPN sites belonging to two different VPNs. A total of four LSPs are required in this topology, one to connect the two sites in VPN B, and three to connect the three sites in VPN A.

VPN Traffic Engineering

An LSP tunnel forms an excellent encapsulation scheme for VPN data flowing between two LSRs. These are complex questions that do not have a single “right” answer.

There is a number of factors that determine what VPN Traffic Engineering (TE) scheme best suits the performance and scalability requirements of a particular customer and their SP.

- **Identifying VPN peers**

This is the first problem facing an LSR that has been configured to support a VPN. The simplest scheme is to use explicit manual configuration of the VPN peers. This is the traditional solution providing obvious and deterministic control of resources and security, but it does not scale well as the size and complexity of the VPN increases.

Alternative schemes automate the process of discovering VPN peers using a directory or by overlaying VPN membership information on one or more routing protocols used on the SP network. This greatly simplifies the configuration task for a VPN since it means that each SP edge router need only be configured with information about the VPNs serviced by each of its customer interfaces. There is clearly a potential security trade-off here as rogue routers can pretend to give access to a VPN.

In comparison, an IPSEC-based solution requires that each SP edge router also be configured with security attributes for each peer in the VPN, which greatly increases the configuration complexity.

- **Multiplexing VPNs on an LSP**

Although LSRs in the core of the SP network do not have to examine the data flowing on VPN LSP tunnels, they are still aware of the existence of these tunnels. This can represent a scalability problem if a separate mesh of LSP tunnels is used for each VPN, because the core LSRs must at least maintain a forwarding table entry and associated resource reservation for each tunnel. If the SP supports thousands of VPN customers, the core LSRs could be required to maintain millions of LSPs. This is the same problem faced by VPN solutions based on ATM or Frame relay technology. Depending on the network topology, this large number of labels may also be beyond the capacity of the LSR switching hardware.

An alternative approach is to multiplex the traffic from multiple VPNs that share the same ingress and egress SP edge routers within a single LSP tunnel between those LSRs. This is achieved using label stacks, with a single outer tunnel set up across the core and an inner LSP that identifies the VPN for which the data is bound. The lower label in the stack is known only to the ingress and egress LSRs.

This use of label stacks reduces the number of LSP tunnels exposed to the network core, but it ties VPNs together. The multiplexed VPNs cannot be routed separately or given different prioritization or drop priority by the core LSRs. The VPNs must also share a single network resource reservation within the network core, which may make it harder for the SP to guarantee the SLA for each individual customer.

Figure B.2 shows two VPNs connected across the MPLS network between a pair of LERs. The traffic for each VPN is carried on a distinct LSP shown as a red and a green line in the diagram. These two VPNs are nested within an outer LSP shown in blue.

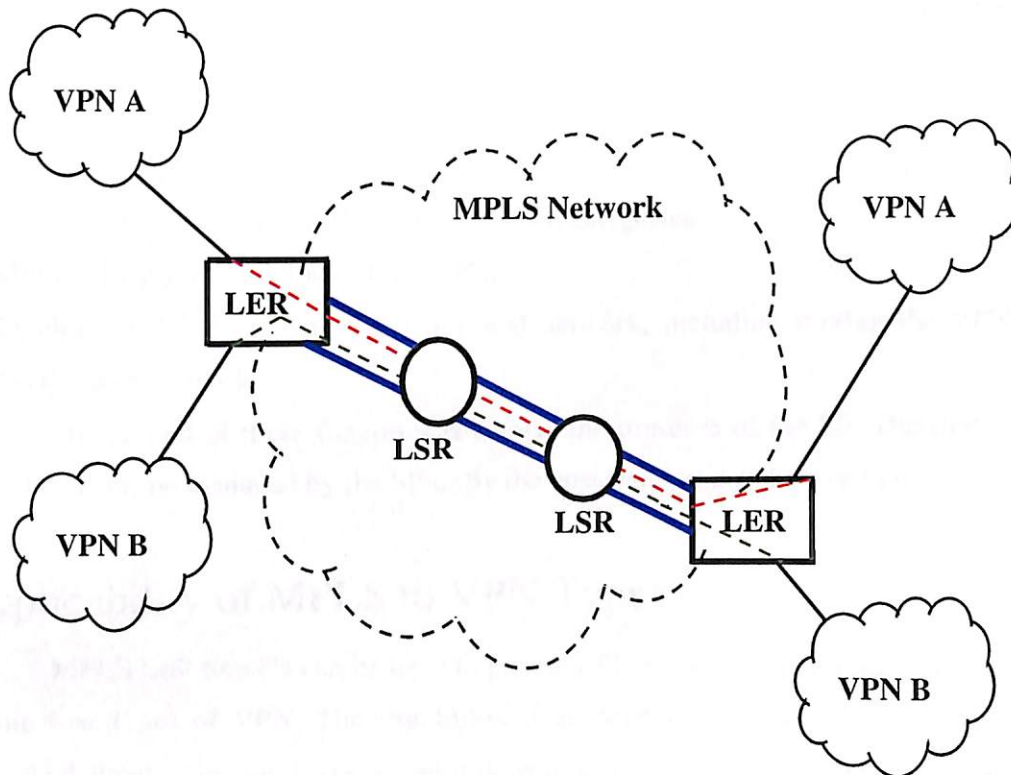


Figure B.2 Nested LSPs Providing VPN Connectivity

- Separating QoS classes

Multiplexing VPNs within a single tunnel helps to reduce the signaling load and forwarding table size in the core LSRs as the number and size of the VPNs increase. However, once the data for multiple streams has been clustered together in a single LSP, it is hard to provide distinct management of the different flows. The encoding of an MPLS label allows three bits to encode the Differentiated Services Code Point (DSCP). Thus a total of eight classes of service (CoS) can be set for packets within any one LSP. These bits can define queuing rules and drop priorities for packets carried on the LSP. In the case of an ATM-based network there is just one bit available to encode the DSCP and this is usually used to indicate the drop preference. If a customer or SP needs to be able to differentiate more than eight DSCPs across the core, multiple outer LSP tunnels must be set up. Each outer tunnel carries a different CoS range and can be routed separately across the core.

- TE across the backbone

MPLS TE can be used to distribute the load within a network and to guarantee bandwidth and QoS by controlling the routing of the outer VPN LSP tunnels across the SP backbone network.

Network Management

The management of a VPN falls into two categories.

- Defining the logical topology of the VPN.
- Mapping the VPN onto the SP's physical network, including routing the VPN data across the core network.

The second of these functions is always the preserve of the SP. The first feature may, however, be managed by the SP or by the customer on a self-serve basis.

Applicability of MPLS to VPN Types

MPLS LSP tunnels can be used to provide all or part of an implementation of any of the four types of VPN. The suitability of an MPLS solution to each VPN type is described below, including the scalability and management challenges such solutions present.

MPLS for VLL

Conceptually, this is the easiest application of MPLS to VPNs. Each point-to-point VLL is provisioned as an LSP tunnel between the appropriate customer sites. The customer is explicitly looking for the equivalent of leased lines, so it is very important that the SP meets any bandwidth guarantees in the SLA. This means that the LSP tunnels used in a VLL solution may have to be dedicated to that specific customer rather than multiplexing the VLL traffic with other VPNs. It is also possible to subdivide the resources of an outer tunnel to provide the QoS for inner LSPs. The point-to-point connectivity of a VLL means that each VLL is most easily provisioned at the edge LSRs by manual configuration rather than an automatic scheme for detecting the VLL peers.

MPLS for VPLS

The most immediately obvious means of implementing a VPLS is to map the LAN segment to a unique IP multicast address perhaps using IP encapsulation of the VPN traffic. Such a solution could use existing IP multicast technologies, rather than MPLS. Indeed, such approaches are offered by many ISPs today. However, technologies such as RSVP do not provide the full TE capabilities of MPLS, so the SP has less control over how the VPLS traffic is routed across the backbone network.

Very large SPs with many VPLS customers may also eventually find that there are too few administratively scoped IPv4 multicast addresses to represent each of the VPN LAN segments that they need to support, forcing them either to move to IPv6 or to multiplex several VPLSs on one multicast address. There are 224 administratively scoped IP multicast addresses (239./8), but an SP may well wish to reserve only a portion of this address space for VPN services. Current MPLS label distribution protocols are specified for unicast destination IP addresses only. This means that an MPLS-based implementation of a VPLS is, necessarily for now, based on one of the following network topologies.

- A full mesh of LSP tunnels connecting the customer sites, with each SP edge LSR responsible for the fan-out to all peers.
- Point-to-point or multipoint-to-point LSP tunnel connections to a “hub” LSR that handles fan-out to all sites using point-to-point LSP tunnels.

But especially for the mesh of LSP tunnels, the MPLS-based topology may use more network bandwidth in total than the IP-multicast based solution. This is because multiple copies of each packet may be sent across any given link, each copy carried within one of several different LSP tunnels for the VPLS that transit that link. However, Service Providers may still choose to implement a VPLS using MPLS in order to exploit the TE capabilities of MPLS to give them better control of how the VPLS traffic is routed between SP edge LSRs.

Future standardization work on MPLS may extend the TE capabilities to cover point-to-multipoint or multipoint-to-multipoint LSP tunnels. Such an extension would allow MPLS-based implementations of a VPLS to avoid the bandwidth overhead compared to an IP-multicast based implementation.

MPLS for VPRN

LSP tunnels provide an excellent solution to VPRNs. A VPRN is routed, rather than requiring point-to-multipoint connectivity. This means that even if the SP edge routers set up a full mesh of LSP tunnels to all the other SP edge routers for a given VPRN, they can route each packet onto a single LSP tunnel according to the destination address for that packet rather than fanning out copies to all peers for that VPRN. This avoids the bandwidth wastage that can occur when using an MPLS-based VPLS, as described in the previous section. Note that the routing protocols used on a VPRN are independent of the routing protocols used on the SP backbone. It is perfectly possible for an SP to use OSPF and BGP4 but for a VPN customer to use a much simpler protocol such as RIP.

MPLS for VPDN

MPLS could be used as the underlying transport mechanism between the LAC and LNS in an L2TP-based VPDN. This is no different from using MPLS to transport any other data that uses public IP addresses. The essential function of a VPDN is provided by L2TP.

Implementation of MPLS VPN Traffic Engineering in C

The code implements LDP (Label Distribution Protocol) and MPLS taking the following constraints into consideration for traffic engineering:

1. Bandwidth.
2. Hop count.
3. Delay.

Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<malloc.h>
#include<string.h>

struct packet{
int seqno;
char srcip[16];
char dstip[16];
char data[1024];
int label;
int VPN_no;
} packet;

struct contpck{ //control packet
int seqno;
char srcip[16];
char dstip[16];
int label;
int delay;
int tot_delay;
int capacity;
int VPN_no;
int hop;
int tot_hop;
char ingress[16];
char egress[16];
};

struct router{ //router
FILE *lib;
FILE *lsp;
int no_link;
int index;
int ing_eg; //0=ingress 1=egress
struct link *lk;
char addr[16];
char host[10][16];
};

struct link
{
int capacity;
int delay;
struct router *dst;
} link;
```

```

struct router *lsrouter;
FILE *label_file;

int flood(struct router *,struct contpck*);           //function for flooding
int createlsp(struct router *,struct contpck);       //function for creating LSP
void send(struct router *,struct packet);

void lsrinit(struct router *lsr, int no)
{
    int i,cap;
    char src[16];
    char dst[16];
    char filename[16];
    lsr->index=no;
    lsr->ing_eg=-1;
    printf("enter the ip for lsr %d: ",no);
    scanf("%s",src);
    strcpy(lsr->addr,src);
    sprintf(filename,"%s%s","lib",src);
    lsr->lib=fopen(filename,"w+");
    sprintf(filename,"%s%s","lsp",src);
    lsr->lsp=fopen(filename,"w+");
    printf("enter no on links connected to lsr %d: ",no);
    scanf("%d",&(lsr->no_link));
    lsr->lk=(struct link *)malloc(lsr->no_link*sizeof(struct link));
}

void linkinit(struct link *lk,struct router *dstl,int cap,int del)
{
    lk->capacity=cap;
    lk->dst=dstl;
    lk->delay=del;
}

void connect(struct router *lsr,int no)
{
    int i,cap,lsrno;

    for(i=0;i<lsr->no_link;i++)
    {
        printf("Enter dest lsr and capacity for link %d from lsr %d(ip=%s): ",i,lsr->index,lsr->addr);
        scanf("%d %d",&lsrno,&cap);
        linkinit(&(lsr->lk[i]),&(lsrouter[lsrno]),cap,50);
    }
    return;
}

int getLabel()
{
    int temp_label,label,cont;
    do
    {
        cont=0;
        fseek(label_file,0,SEEK_SET);
        label=rand()%1000;
        while(!feof(label_file))
        {
            fscanf(label_file,"%d\n",&temp_label);
            if(label==temp_label)
            {
                cont=1;
                break;
            }
        }
    }while(cont);
    return label;
}

```

```

}

int flood(struct router *lsr,struct contpck *cpck)
{
char ip[16]="",temp[16]="";
struct contpck cpck_send;
int seq=0,seq1=0,i,exists=0,ret=0,count=0,delay,delay1,vpn_no,hop;
int ret1;

fseek(lsr->lsp,0,SEEK_SET);
while(!feof(lsr->lsp))
{
fscanf(lsr->lsp,"%s\t%d\t%d\t%d\t%d\n",ip,&seq,&delay1,&hop,&vpn_no);
if((seq==cpck->seqno)&&(!strcmp(ip,cpck->srcip))) //duplicate LSP REQ packet from some node
{
if((cpck->tot_delay>=delay1)&&(cpck->tot_hop>=hop)&&(cpck->VPN_no==vpn_no)) //if delay is >= the
existing delay and the hop count exceeds
return 0;
}
}

// if REQ pckt from some other node
// or from same node but with less delay or less hopcount
fseek(lsr->lsp,0,SEEK_END);
fprintf(lsr->lsp,"%s\t%d\t%d\t%d\t%d\n",cpck->srcip,cpck->seqno,cpck->tot_delay,cpck->tot_hop,cpck->VPN_no);
if(!strcmp(lsr->addr,cpck->egress)) //if egress
{
printf("\n%o: LSP REQ %d REACHED\n",lsr->addr,cpck->seqno);
cpck->delay=-1;
return 1;
}
else
{
strcpy(temp,cpck->srcip);
strcpy(cpck->srcip,lsr->addr);
delay=cpck->tot_delay;
hop=cpck->tot_hop;
for(i=0;i<lsr->no_link;i++)
{
strcpy(cpck->srcip,lsr->addr);
cpck->tot_delay=delay;
cpck->tot_hop=hop;
if((strcmp(temp,((lsr->lk[i]).dst)->addr)!=0)&&(strcmp(((lsr->lk[i]).dst)->addr,cpck->ingress)!=0))
//dont send to ingress and the sending node
{
if((lsr->lk[i]).capacity>=cpck->capacity)
{
if((delay+(lsr->lk[i]).delay<=cpck->delay)&&((1+cpck->tot_hop)<=cpck->hop))
{
cpck->tot_delay=delay+(lsr->lk[i]).delay;
++(cpck->tot_hop);
ret=flood(((lsr->lk[i]).dst),cpck);
if(ret==1) //flood succeeds
count++;
if(cpck->delay==1) //Packet has reached the egress
break;
}
}
}
}
}
if(cpck->delay!=-1){
cpck->tot_delay=delay;
cpck->tot_hop=hop;
}
}

```

```

        if(count>0) //atleast one path through this node exists
            return 1;
        else //no path through this node
            return 0;
    } //end of else
}

int createlsp(struct router *lsr,struct contpck cpck)
{
    char ip[16]="",src[15]="",dst[15]="";
    int seq=-1,in_label=-1,out_label=-1,label=-1,cont=0,lsrno=-1,i,delay,vpn_no,hop=-1;
    if(!strcmp(lsr->addr,cpck.ingress)) //ingress
    {
        fseek(lsr->lib,0,SEEK_END);
        fprintf(lsr->lib,"%s\t%d\t%s\t%d\t%d\n",cpck.dstip,-1,cpck.srcip,cpck.label,cpck.VPN_no);
        printf("*****LSP CREATED*****\n\n");
        return 1;
    }

    fseek(lsr->lsp,0,SEEK_SET);

    while(!feof(lsr->lsp)) //find any path which satisfies delay constraint
    {
        fscanf(lsr->lsp,"%s\t%d\t%d\t%d\t%d\n",ip,&seq,&delay,&hop,&vpn_no);
        if((delay<=cpck.tot_delay)&&(seq==cpck.seqno)&&(vpn_no==cpck.VPN_no)&&(hop<=cpck.tot_hop))
            break;
    }

    do
    {
        cont=0;
        fseek(lsr->lib,0,SEEK_SET);
        label=rand()%100;
        while(!feof(lsr->lib))
        {
            fscanf(lsr->lib,"%s\t%d\t%s\t%d\t%d\n",src,&in_label,dst,&out_label,&vpn_no);
            if((label==in_label)&&(strcmp(src,ip)==0)) //if entry with same label and same interface is there
            {
                cont=1;
                break;
            }
        }
    } while(cont);
    fseek(lsr->lib,0,SEEK_END);

    if(!strcmp(lsr->addr,cpck.egress)) //if egress
        fprintf(lsr->lib,"%s\t%d\t%s\t%d\t%d\n",ip,label,"-1",-1,cpck.VPN_no);
    else
        fprintf(lsr->lib,"%s\t%d\t%s\t%d\t%d\n",ip,label,cpck.srcip,cpck.label,cpck.VPN_no);

    cpck.label=label;
    strcpy(cpck.srcip,lsr->addr);
    for(i=0;i<lsr->no_link;i++)
    {
        if(strcmp(((lsr->lk[i]).dst)->addr,ip)==0)
        {
            lsrno=((lsr->lk[i]).dst)->index;
            break;
        }
    }
    if(lsrno<0)
    {
        printf("ERROR IN LSPCREATE\n");
        return 0;
    }
}

```

```

    }
    else
    {
        printf("%s: sent label to %s\n",lsr->addr.ip);
        cpck.tot_delay+=(lsr->lk[i]).delay;
        cpck.tot_hop+=1;
        createlspt(&(lsrouter[lsrno]),cpck);
    }
    return 1;
}

void send(struct router *lsr,struct packet pck)
{
    char in_ip[16]="",out_ip[16]="",dest[16]="";
    int i,in_lab=-1,out_lab=-1,lsrno=-1,vpn_no=-1;
    fseek(lsr->lib,0,SEEK_SET);

    if(lsr->ing_eg==0) //ingress
    {
        while(!feof(lsr->lib))
        {
            fscanf(lsr->lib,"%s\t%d\t%s\t%d\t%d\n",in_ip,&in_lab,out_ip,&out_lab,&vpn_no); //in_ip contains the
            dest ip(egress)
            if(!strcmp(pck.dstip,in_ip) && (vpn_no==pck.VPN_no))
            {
                pck.label=out_lab;
                strcpy(dest,out_ip);
                break;
            }
        }
    }

    else if((lsr->ing_eg)==1) //egress
    {
        printf("%s: packet %d reached\n",lsr->addr.pck.seqno);
        return;
    }

    else //if not egress
    {
        while(!feof(lsr->lib))
        {
            fscanf(lsr->lib,"%s\t%d\t%s\t%d\t%d\n",in_ip,&in_lab,out_ip,&out_lab,&vpn_no);
            if((strcmp(pck.srcip,in_ip)==0)&&(pck.label==in_lab))
            {
                pck.label=out_lab;
                strcpy(dest,out_ip);
                break;
            }
        }
        for(i=0;i<lsr->no_link;i++)
        {
            if(strcmp(((lsr->lk[i]).dst)->addr,dest)==0)
                lsrno=((lsr->lk[i]).dst)->index;
        }
        strcpy(pck.srcip,lsr->addr);
        printf("%s: packet %d sent to %s\n",lsr->addr.pck.seqno,dest);
        send(&(lsrouter[lsrno]),pck);
    }
}

struct contpck cpck_init(int seq, int del,int cap,int vpn_no,int hop,char *ing,char* eg)
{
    struct contpck cpck;

```



```

cpck.seqno=seq;
cpck.tot_delay=0;
cpck.delay=del;
cpck.capacity=cap;
cpck.VPN_no=vpn_no;
cpck.hop=hop;
cpck.tot_hop=0;
strepyp(cpck.srcip,ing);
strepyp(cpck.dstip,eg);
strepyp(cpck.ingress,ing);
strepyp(cpck.egress,eg);
cpck.label=-1;
return cpck;
}

int main()
{
    int no_lsr,i,j,ret=0,cap=0,delay=0,seq=0,vpn_no=-1,hopcount=-1;
    char temp,answer='n';
    char ingress[16]="",egress[16]="";
    struct contpck cpck;
    struct packet pck;
    printf("Enter no of lsrs: ");
    scanf("%d",&no_lsr);
    lsrouter=(struct router *)malloc(no_lsr*sizeof(struct router));

    for(i=0;i<no_lsr;i++)
        lsrintit(&lsrouter[i],i);

    for(i=0;i<no_lsr;i++)
        connect(&lsrouter[i],i);

    pck.seqno=2;
    strepyp(pck.srcip,ingress);
    strepyp(pck.dstip,egress);
    pck.label=-1;
    do
    {
        seq++;
        printf("enter the ingress: ");
        scanf("%s",ingress);
        printf("enter the egress: ");
        scanf("%s",egress);
        printf("enter the VPN No : ");
        scanf("%d",&vpn_no);

        printf("\nenter the bandwidth required: ");
        scanf("%d",&cap);

        printf("\nenter the delay required: ");
        scanf("%d",&delay);
        printf("\nenter the hopcounts required: ");
        scanf("%d",&hopcount);
        cpck=cpck_init(seq,delay,cap,vpn_no,hopcount,ingress,egress);
        for(j=0;j<no_lsr;j++)
        {
            lsrouter[j].ing_eg=-1;
        }
        for(i=0;i<no_lsr;i++)
        {
            if(!strcmp(lsrouter[i].addr,ingress))
            {
                lsrouter[i].ing_eg=0;
                break;
            }
        }
    }
}

```

```

for(j=0;j<no_lsr;j++)
{
    if(!strcmp(lsrouter[j].addr,egress))
    {
        lsrouter[j].ing_eg=1;
        break:
    }
}
printf("Ingress= %d and Egress= %d",i,j);
printf("\n");
ret=floodt(&(lsrouter[i]),&cpck);

strcpy(cpck.srcip,ingress);
strcpy(cpck.dstip,egress);
pck.seqno=seq++;
pck.VPN_no=vpn_no;
strcpy(pck.srcip,ingress);
strcpy(pck.dstip,egress);
pck.label=-1;

if(ret==0)
    printf("no path with desired QoS\n");
else
{
    if(createlsp(&(lsrouter[j]),cpck))
        send(&(lsrouter[i]),pck);
}
printf("\nWant to enter more(y/n): ");
scanf("%c",&answer);
scanf("%c",&answer);
}while(answer=='Y'\answer=='y');
}

```

Sample Outputs:

```

Enter no of lsrs: 6
enter the ip for lsr 0: 10
enter no on links connected to lsr 0: 3
enter the ip for lsr 1: 20
enter no on links connected to lsr 1: 2
enter the ip for lsr 2: 30
enter no on links connected to lsr 2: 3
enter the ip for lsr 3: 40
enter no on links connected to lsr 3: 2
enter the ip for lsr 4: 50
enter no on links connected to lsr 4: 3
enter the ip for lsr 5: 60
enter no on links connected to lsr 5: 3
Enter dest lsr and capacity for link 0 from lsr 0(ip=10): 1 3
Enter dest lsr and capacity for link 1 from lsr 0(ip=10): 4 4
Enter dest lsr and capacity for link 2 from lsr 0(ip=10): 5 6
Enter dest lsr and capacity for link 0 from lsr 1(ip=20): 0 3
Enter dest lsr and capacity for link 1 from lsr 1(ip=20): 2 5
Enter dest lsr and capacity for link 0 from lsr 2(ip=30): 1 5
Enter dest lsr and capacity for link 1 from lsr 2(ip=30): 3 6
Enter dest lsr and capacity for link 2 from lsr 2(ip=30): 4 5
Enter dest lsr and capacity for link 0 from lsr 3(ip=40): 2 6
Enter dest lsr and capacity for link 1 from lsr 3(ip=40): 5 4
Enter dest lsr and capacity for link 0 from lsr 4(ip=50): 0 4
Enter dest lsr and capacity for link 1 from lsr 4(ip=50): 2 5
Enter dest lsr and capacity for link 2 from lsr 4(ip=50): 5 5
Enter dest lsr and capacity for link 0 from lsr 5(ip=60): 0 6

```

Enter dest lsr and capacity for link 1 from lsr 5(ip=60): 4 5
Enter dest lsr and capacity for link 2 from lsr 5(ip=60): 3 4
enter the ingress: 10
enter the egress: 40
enter the VPN No : 1

enter the bandwidth required: 5

enter the delay required: 200

enter the hopcounts required: 4
Ingress= 0 and Egress= 3

40: LSP REQ 1 REACHED
40: sent label to 30
30: sent label to 50
50: sent label to 60
60: sent label to 10
*****LSP CREATED*****

10: packet 1 sent to 60
60: packet 1 sent to 50
50: packet 1 sent to 30
30: packet 1 sent to 40
40: packet 1 reached

Want to enter more(y/n): y
enter the ingress: 20
enter the egress: 60
enter the VPN No : 2

enter the bandwidth required: 3

enter the delay required: 100

enter the hopcounts required: 2
Ingress= 1 and Egress= 5

60: LSP REQ 3 REACHED
60: sent label to 10
10: sent label to 20
*****LSP CREATED*****

20: packet 3 sent to 10
10: packet 3 sent to 60
60: packet 3 reached

Want to enter more(y/n): y
enter the ingress: 10
enter the egress: 40
enter the VPN No : 3

enter the bandwidth required: 7

enter the delay required: 250

enter the hopcounts required: 6
Ingress= 0 and Egress= 3
no path with desired QoS

Appendix C

Simulation of IPSec VPN in Tunnel and Transport Modes

In this part IPSec Client-Server communication in the TUNNEL and TRANSPORT modes using Authentication Header (AH) and Encrypted Security Payload (ESP) security policies are simulated. The client and server programs were run over the testbed as shown in figure C.1.

Components:

- IPSec Gateway
- Legitimate Client
- Illegitimate Client (Snooper)

Inputs to the Simulator:

- Following are the inputs to the simulator at the client end
- Security Level (AH or ESP)
- Communication Mode (Transport or Tunnel)
- Application Data (Configuration file)

Here, a client is trying to write a critical configuration file on the server, server cannot accept that file from any client. Thus for authentication and confidentiality of the data an IPSec connection is opened between them. Connections can be opened in Tunnel and Transport modes.

Following events happen on the client side

Tunnel Mode:

1. Client attaches the application header to the payload data.
2. TCP attaches it's header to the application payload
3. IP Header is attached by the client containing the final destination address

4. The client now sends this packet to the IPSec gateway
5. Depending on the Security Association, the IPSec gateway hashes or encrypts the data and attaches the AH (containing ICV key generated by MD5 hashing) or ESP header (containing the encrypted data generated by RSA algorithm)
6. IPSec attaches the new IP header containing the IP address of the destination IPSec gateway and transmits the packet

Transport Mode:

1. Client attaches the application header to the payload data.
2. TCP attaches its header to the application payload
3. Depending on the Security Association, the client hashes or encrypts the data and attaches the AH (containing ICV key generated by MD5 hashing) or ESP header (containing the encrypted data generated by RSA algorithm)
4. IP Header is attached by the client containing the final destination address and sends the packet to the server on the same LAN

Following events happen on the server side

- Through the Security Association the server knows whether the received packet is an AH or ESP.
- If the packet is AH, it runs the agreed hashing algorithm (MD5 in our case) and finds out the ICV key, which is compared with the ICV key received from the client in the AH header, if they happen to be the same then the client is authenticated successfully else authentication fails.
- If the packet is ESP, it runs the agreed decryption algorithm (RSA in our case) using the agreed keys. If the key is correct, correct data will be obtained else it gets garbled data.

Testbed Scenario:

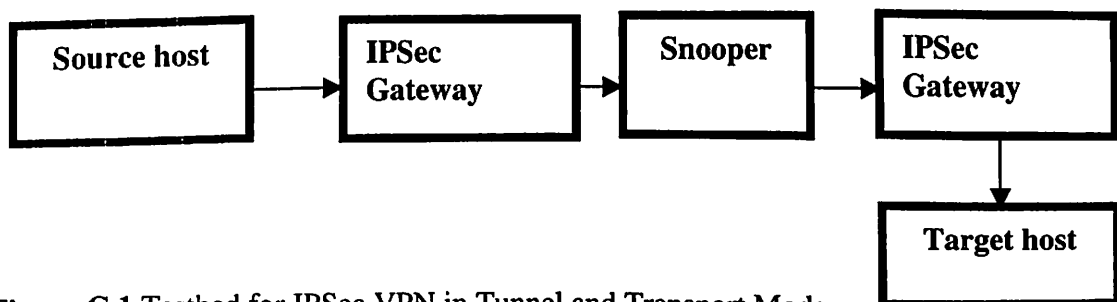


Figure C.1 Testbed for IPSec VPN in Tunnel and Transport Mode

The Simulation for the IPsec Client-Server communication has been successfully tested. This simulation demonstrates the working of the actual IPsec protocol and can be used as a primary design for the full-fledged implementation of IPsec protocol suite. The sniffer was unable to make out the communication that was going on between the source host and the target host. Below are the programs used:

protocol.h : This file describes the packet exchange protocols between the client and the server.

```
#define true 1
#define false 0
#define boolean int
#define TRANSPORT 1
#define TUNNEL 2
#define IP 1
#define TCP 2
#define AP 3
#define AH 4
#define ESP 5
#define AH_ESP 6
typedef struct{
    char next_header;
    char ah_header_length;
    short reserved;
    unsigned SPI;
    unsigned seq_no;
    long icv[4];
}AH_HEADER; //size = 28

typedef struct{
    unsigned SPI;
    unsigned seq_no;
    unsigned payload_length;
    char padding[4]; //max it can be 255
    char pad_length;
    char next_header;
    unsigned int icv;
}ESP_HEADER; //size = 22, actual=24
typedef struct{
    char version_and_headerlength; //4bit each
    char tos;
    short payload_length;
    short identification;
    short flags_and_frag_off; //3bit flag and 13 bit offset
    char TTL;
    char protocol;
    short checksum;
    unsigned source_IP;
    unsigned dest_IP;
}IP_HEADER; //size = 20

typedef struct{
    short source_port;
    short dest_port;
    unsigned seq_no;
    unsigned ack;
    short headerlength_and_flags; //4 bit header-length 6 bits reserved and 6 bits flags
    short window_size;
    short TCP_checksum;
    short urgent_pointer;
```

```

}TCP_HEADER://size = 20
typedef struct{
    short app_data_length;
    short security_level; //1 for AH 2. ESP 3.AH+ESP
    short mode; //1.Intranet(transport) 2.Internet(tunnel)
    char data[100];
}APP_PACKET://size = size = 106, acutal = 108
typedef struct{
    IP_HEADER ip;
    union{
        AH_HEADER ah;
        ESP_HEADER esp;
    }type;
    TCP_HEADER tcp;
    APP_PACKET ap;
}TRANSPORT_MODE_PACKET; // size = 176

typedef struct{
    IP_HEADER new_ip;
    union{
        AH_HEADER ah;
        ESP_HEADER esp;
    }type;
    IP_HEADER old_ip;
    TCP_HEADER tcp;
    APP_PACKET ap; // size = 196
}TUNNEL_MODE_PACKET;

typedef struct{
    short mode;
    short type;
    union{
        TRANSPORT_MODE_PACKET trmp;
        TUNNEL_MODE_PACKET tump;
    }packet_type; //size = 200
}ACTUAL_PACKET;

```

client.c: This file contains the code for the client which generate AH or ESP packets in the TRANASPORT or TUNNEL mode and sends them to the VPN server.

```

#include "myinclude.h"
#include "protocol.h"
#include "wrapper.c"
#include "config.h"
APP_PACKET ap;
IP_HEADER ip;
TCP_HEADER tcp;
AH_HEADER ah;
ESP_HEADER esp;
TRANSPORT_MODE_PACKET trmp;
TUNNEL_MODE_PACKET tump;
ACTUAL_PACKET p;
void make_app_packet(FILE*,char,char);
void add_tcp_header(void);
void add_ip_header(void);
void add_ah_header(short);
void add_esp_header(short);

int main(int argc,char *argv[]){
    int sockfd;
    register i;
    long h[4];

```

```

short security_level,mode;
FILE *fp = fopen("original_data","r");
char *buff,*esp_buff;
struct sockaddr_in servaddr;
int d.n.e_size=0,d_size;
void *dec_msg;
int *enc_msg;

typedef struct{
    short mode;
    short type;
    int size;
}ESP_INFO;

ESP_INFO e,tmp;
system("clear");
printf("\n\n\t\tSimulation of the IPSec standard\n");
printf("\nWelcome to the IPSec client...\n");
printf("\nEnter the Security level (AH=4 or ESP=5): ");
scanf("%hd",&security_level);
printf("\nEnter the mode (TRANSPORT=1, TUNNEL=2): ");
scanf("%hd",&mode);
make_app_packet(fp,security_level,mode);

switch(mode){
    case TRANSPORT:
        if(security_level == 4){
            //ADD the AH Header
            add_tcp_header();
            tmp.tcp = tcp;
            add_ah_header(TRANSPORT);
            tmp.type.ah = ah;
            tmp.ap = ap;
            add_ip_header();
            tmp.ip = ip;
        }
        else{
            //add the ESP header
            add_tcp_header();
            tmp.tcp = tcp;
            add_esp_header(TRANSPORT);
            tmp.type.esp = esp;
            tmp.ap = ap;
            add_ip_header();
            tmp.ip = ip;
        }
        break;
    case TUNNEL:
        if(security_level == 4){
            //ADD the AH Header
            add_tcp_header();
            tmp.tcp = tcp;
            add_ip_header();
            tmp.old_ip = ip;
            add_ah_header(TUNNEL);
            tmp.type.ah = ah;
            tmp.ap = ap;
            add_ip_header();
            tmp.new_ip = ip;
        }
        else{
            //ESP IN Tunnel

```



```

        add_tcp_header();
        tump.tcp = tcp;
        add_ip_header();
        tump.old_ip = ip;
        add_esp_header(TUNNEL);
        tump.type.esp = esp;
        tump.ap = ap;
        add_ip_header();
        tump.new_ip = ip;
    }
    break;

default:
    printf("Illegal packet type\n");
}
//end of switch
sockfd = Socket(AF_INET,SOCK_STREAM,0);
bzero(&servaddr, sizeof(servaddr));
servaddr.sin_family = AF_INET;
servaddr.sin_port = htons(SERV_PORT);
Inet_pton(AF_INET,DEST_IP,&servaddr.sin_addr);
Connect(sockfd, (struct sockaddr *)&servaddr,sizeof(servaddr));
switch(security_level){
    case AH:
        if(mode == TRANSPORT){
            buff = (char*)malloc(sizeof(ACTUAL_PACKET));
            p.mode = TRANSPORT;
            p.type = AH;
            p.packet_type.trmp = trmp;
            memcpy(buff,&p,sizeof(ACTUAL_PACKET));
            printf("\nMD5 Hashing function called to generate the authentication key...\n");
            sleep(1);
            hash(h,buff,sizeof(ACTUAL_PACKET));
            for(i=0;i<4;i++){
                p.packet_type.trmp.type.ah.icv[i] = h[i];
            }
            n = write(sockfd,&p,sizeof(ACTUAL_PACKET));
            printf("\nPacket sent to the server...\n");
        }
        else{
            buff = (char*)malloc(sizeof(ACTUAL_PACKET));
            p.mode = TUNNEL;
            p.type = AH;
            p.packet_type.tump = tump;
            memcpy(buff,&p,sizeof(ACTUAL_PACKET));
            printf("\nMD5 Hashing function called to generate the authentication key...\n");
            sleep(1);
            hash(h,buff,sizeof(ACTUAL_PACKET));
            for(i=0;i<4;i++){
                p.packet_type.tump.type.ah.icv[i] = h[i];
            }
            n = write(sockfd,&p,sizeof(ACTUAL_PACKET));
            printf("\nPacket sent to the server...\n");
        }
        break;
    case ESP:
        if(mode != TRANSPORT){
            buff = (char*)malloc(sizeof(ACTUAL_PACKET));
            p.packet_type.tump = tump;
            p.mode = TUNNEL;
            p.type = ESP;
            memcpy(buff,&p,sizeof(ACTUAL_PACKET));
            printf("\nEncrypting the packet using RSA algorithm...\n");
            sleep(1);
            rsa_encrypt(buff,sizeof(ACTUAL_PACKET),&enc_msg,&e_size,RSA_KEY_E,RSA_KEY_N);
            printf("\nPacket encrypted successfully...\n");
        }

```



```

        ip.payload_length = sizeof(TCP_HEADER) + 4 + sizeof(ap.data);
        ip.identification = 0;
        ip.flags_and_frag_off = 0;
        ip.TTL = 32;
        ip.protocol = 65;
        ip.checksum = 0;
        Inet_pton(AF_INET,SERV_IP, &ip.source_IP);
        Inet_pton(AF_INET,SERV_IP, &ip.dest_IP);
        printf("\nIP header generated and attached...\n");
        sleep(1);
    } //end of add_ip_header

void add_ah_header(short mode){
    register i;
    if(mode == TUNNEL)
        ah.next_header = IP;
    else
        ah.next_header = TCP;
    ah.ah_header_length = sizeof(AH_HEADER);
    ah.reserved = 0;
    ah.SPI = 1; //to be generated
    ah.seq_no = 1;
    for(i=0;i<4;i++)
        ah.icv[i] = 0;
    printf("\nAH header generated and attached...\n");
    sleep(1);
} //end of add_ah_header

void add_esp_header(short mode){
    register i;
    esp.SPI = 1;
    esp.seq_no = 1;
    if(mode == TUNNEL)
        esp.payload_length = sizeof(TCP_HEADER) + 4 + sizeof(ap.data);
    else
        esp.payload_length = sizeof(TCP_HEADER) + 4 + sizeof(ap.data) + sizeof(IP_HEADER);
    for(i=0;i<4;i++)
        esp.padding[i] = 255;
    esp.pad_length = 4;
    if(mode == TUNNEL)
        ah.next_header = IP;
    else
        ah.next_header = TCP;
    esp.icv = 0; //RSA Key
    printf("\nESP header generated and attached...\n");
    sleep(1);
} //end of add_esp_header

```

server.c: This is the IPSec VPN server which gets the packets from the client, checks for the authenticity of clients using RSA for AH mode and MD5 for ESP mode and displays the contents of the packet if the client is authenticated successfully.

```

#include "myinclude.h"
#include "wrapper.c"
#include "protocol.h"
#include "config.h"
#include <time.h>

ACTUAL_PACKET *p,*esp_p;
long h[4],old_h[4];
boolean authenticate(long *,long *);
void success_msg(short);

```

```

void error_msg(void);
int main(int argc, char **argv){
    int listenfd,connfd,n,d_size,*dec_buff;
    boolean valid;
    register i;
    struct sockaddr_in servaddr;
    char *buff;
    typedef struct{
        short mode;
        short type;
        int size;
    }ESP_INFO;
    ESP_INFO e;
    listenfd = Socket(AF_INET, SOCK_STREAM,0);
    bzero(&servaddr,sizeof(servaddr) );
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    servaddr.sin_port = htons(SERV_PORT);
    Bind(listenfd, (struct sockaddr*)&servaddr, sizeof(servaddr));
    Listen(listenfd, LISTENQ);
    buff = (char *)malloc(4);
    system("clear");
    printf("\n\n\tWelcome to IPSec server...\n");
    while(1){
        printf("\nServer waiting for clients to connect...\n");
        connfd = Accept(listenfd,(struct sockaddr*)NULL, NULL);
        system("clear");
//        n = read(connfd,buff,sizeof(ACTUAL_PACKET));
        n = read(connfd,buff,4);
        memcpy(&e,buff,4);
        printf("\nClient arrived with mode=%s and
type=%s\n",e.mode==TRANSPORT?"TRANSPORT":"TUNNEL",e.type==ESP?"ESP":"AH");
        if(e.type == ESP){
            switch(e.mode){
                case TRANSPORT:
                    n = read(connfd,buff,4);
                    memcpy(&e.size,buff,4);
                    free(buff);
                    buff = (char*)malloc(e.size);
                    n = read(connfd,buff,e.size);
                    dec_buff = (int*)malloc(e.size);
                    memcpy(dec_buff,buff,e.size);
                    printf("\nRSA decryption algorithm called...\n");
                    sleep(1);
                    rsa_decrypt(dec_buff,e.size,&esp_p,&d_size,RSA_KEY_D,RSA_KEY_N);
                    printf("\nPacket decrypted successfully...\n");
                    printf("\nData in the packet is:\n%s\n",esp_p->packet_type.ttmp.ap.data);
                    break;
                case TUNNEL:
                    n = read(connfd,buff,4);
                    memcpy(&e.size,buff,4);
                    free(buff);
                    buff = (char*)malloc(e.size);
                    n = read(connfd,buff,e.size);
                    dec_buff = (int*)malloc(e.size);
                    memcpy(dec_buff,buff,e.size);
                    printf("\nRSA decryption algorithm called...\n");
                    sleep(1);
                    rsa_decrypt(dec_buff,e.size,&esp_p,&d_size,RSA_KEY_D,RSA_KEY_N);
                    printf("\nPacket decrypted successfully...\n");

                    printf("\nData in the packet is:\n%s\n",esp_p->packet_type.tump.ap.data);

```

```

                break;
            default: printf("illegal esp\n");
        } //end of switch
    } //end of ESP
    else{
free(buff);
        buff = (char *)malloc(sizeof(ACTUAL_PACKET)-4);
        n = read(connfd,buff,sizeof(ACTUAL_PACKET)-4);
        p = (ACTUAL_PACKET *)malloc(sizeof(ACTUAL_PACKET));
        p->mode = c.mode;
        p->type = c.type;
        switch(p->mode){
            case TRANSPORT:
                memcpy(&p->packet_type.trmp,buff,sizeof(ACTUAL_PACKET)-4);
                free(buff);
                buff = (char*)malloc(sizeof(ACTUAL_PACKET));
                if(p->type == AH){
                    for(i=0;i<4;i++){
                        old_h[i] = p->packet_type.trmp.type.ah.icv[i];
                        p->packet_type.trmp.type.ah.icv[i] = 0;
                    }
                    memcpy(buff,p,sizeof(ACTUAL_PACKET));
                    printf("\nMD5 Hashing function called to generate the authentication key...\n");
                    sleep(1);
                    hash(h,buff,sizeof(ACTUAL_PACKET));
                    valid = authenticate(h,old_h);
                    if(valid)
                        success_msg(p->mode);
                }
                else
                    error_msg();
            } //end of if
            break;

            case TUNNEL:
                memcpy(&p->packet_type.tump,buff,sizeof(ACTUAL_PACKET)-4);
                free(buff);
                buff = (char*)malloc(sizeof(ACTUAL_PACKET));
                if(p->type == AH){
                    for(i=0;i<4;i++){
                        old_h[i] = p->packet_type.tump.type.ah.icv[i];
                        p->packet_type.tump.type.ah.icv[i] = 0;
                    }
                    memcpy(buff,p,sizeof(ACTUAL_PACKET));
                    printf("\nMD5 Hashing function called to generate the authentication key...\n");
                    sleep(1);
                    hash(h,buff,sizeof(ACTUAL_PACKET));
                    valid = authenticate(h,old_h);
                    if(valid)
                        success_msg(p->mode);
                }
                else
                    error_msg();
            } //end of if
            break;

            default:
                printf("mode = %hd\n",p.mode);
                printf("Illegal....!\n");
        } //end of switch
    } //end of else
} //end of while(1)
} //end of main

```

```

void success_msg(short m){
    printf("\nClient authenticated successfully...\n");
    printf("\nApplication data recieved is:\n");
    if(m==TRANSPORT)
        printf("%s\n\n",p->packet_type.trmp.ap.data);
    else
        printf("%s\n\n",p->packet_type.tump.ap.data);
/*
    printf("Hash key by server: %u %u %u %u\n", h[0], h[1], h[2], h[3] );*/
}
//end of success_msg
void error_msg(void){
    printf("\nClient could not be authenticated successfully!!\n");
    printf("From client: Hash value is %u %u %u %u\n", old_h[0], old_h[1], old_h[2], old_h[3] );
    printf("From server: Hash value is %u %u %u %u\n", h[0], h[1], h[2], h[3] );

}
//end of error_msg
boolean authenticate(long *val1,long *val2){
    register i;
    for(i=0;i<4;i++){
        if(val1[i] == val2[i])
            continue;
        else
            break;
    }
    if(i!=4)
        return false;
    return true;
}
//end of authenticate

```

rsa.c: This is the file which implements the 32-bit RSA algorithm
#include "rsa.h"

```

void get_prime(int e, int *prime)
{
    int found = 0, p1, p2;
    time_t curr_time;
    srandom((unsigned int) time(&curr_time));

    while (found == 0)
    {
        int temp1, temp, temp2, p, count;
        int y, z, a, b, c;

        temp1 = random();
        temp = (temp1 % 55);

        p = (temp*6)+5;
        temp2 = p-1;
        if (temp2 == 0)
            continue;
        else
        {
            b=0;
            for (count=0; count<32;count++)
            {
                if ((temp2 % 2) == 0)
                    b++;
                else
                    break;
                temp2 = temp2 >> 1;
            } /* for(count=0;..... */
            c = (p-1) / (pow(2, b));
        } /* else */
    }
}

```

```

a = random();
a = a%250;
pow_modn(a, c, &y, p);
temp2 = y;
for (count=0; count<b; count++)
{
    my_square(temp2, p, &z);
    if ((z==1) && (( temp2 == 1) || (temp2 == -1)))
    {
        *prime = p;
        found = 1;
    }
    temp2 =z;
}
} /* while (found == 0) */
}
void my_square(int base, int p, int* result)
{
    (*result) = base % p;
    (*result) = (*result) * (*result);
    (*result) %= p;
    return;
}
void pow_modn(int base, int exp, int *result, int n)
{
    int size_key = 32; /*TBD : This is to be passed as an argument */
    char *num;
    short start_flag;
    int temp1, scan;
    *result = 1;
    num = (char*) malloc(size_key);
    temp1 = exp;
    for (scan = (size_key-1); scan >= 0; scan--)
    {
        if (temp1 % 2 == 0)
            *(num+scan) = '0';
        else
            *(num+scan) = '1';
        temp1 = temp1 >> 1;
    }
    for (scan = 0; scan < size_key; scan++)
    {
        if (start_flag)
            my_square(*result, n, result);

        if (*(num+scan) == '1')
        {
            if (!start_flag)
            {
                start_flag = 1;
                (*result) = base;
            }
            else
                (*result) *= base;
            (*result) %= n;
        }
    }
    free(num);
    return;
}
void est_d_n(int *e, int *d, int *n)
{ int p, q, found_d = 0, c = 0;

```

```

double temp1;
long temp_d;
int o_n;
*e = 3;
rep: get_prime(*e, &p);
if(p>230)
    goto rep;
else
    if((p%5)==0)
        goto rep;
sleep(1);
rep2: get_prime(*e, &q);
if(q>250)
    goto rep2;
else
    if((q%5) == 0)
        goto rep2;
else
    if(p==q)
        goto rep;
*n = p*q;
o_n = (p-1) * (q-1);
while (found_d == 0)
{
    c++;
    temp1 = ((float)((c*o_n)+1))/((float)(*e));
    temp_d = (long)temp1;
    if (((double)temp_d) == temp1)
    {
        found_d = 1;
        *d = (int) temp1;
    }
}

return;
}
void ker_crypt(int msg, int e, int n, int *enc_msg)
{
    pow_modn(msg, e, enc_msg, n);
return;
}
void rsa_encrypt(void *msg, int size_msg, int **enc_msg, int *enc_size, int e, int n)
{
    char mesg;
    char *scan;
    int num, i;
    *enc_msg = NULL;
    *enc_msg = (int*)malloc(size_msg*sizeof(int));
    *enc_size = size_msg*sizeof(int);
    scan = (char*)msg;
    for(i=0; i<size_msg; i++)
    {
        mesg = *scan;
        num = (int)mesg;
        ker_crypt(num, e, n, (*enc_msg+i));
        scan++;
    }
    return;
}
void rsa_decrypt(int *msg, int size_msg, void **dec_msg, int *dec_size, int e, int n)
{
    int *scan=msg;
    int size_m, temp, i;

```



```

char *tempc;
size_m = size_msg/4;
*dec_size = size_m;
(*dec_msg) = malloc(size_m*sizeof(char));
tempc = (char *)(*dec_msg);
for(i=0; i<size_m; i++)
{
    ker_crypt(*(scan+i), e, n, &temp);
    *(tempc+i) = temp;
}
return:
}

```

md5.c: This is the file which implements the MD5 algorithm

```

#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netdb.h>
#include<netinet/in.h>
#define ALGO 4
#define f(x,y,z) ((x&y)|(~x)&z))
#define g(x,y,z) ((x&y)|(y&(~z)))
#define h(x,y,z) (x^y^z)
#define i(x,y,z) (y^(x&(~z)))
#define max 512
typedef struct
{
    int x:int y:int z: }S;

typedef struct
{
    short f1:short f2:short f3:short f4:short l:short s;
    unsigned int t;
}OPER;

typedef struct
{
    long s_no;
    short algo_id;
    char issu[40];
    int from;
    int to;
    char subname[40];
    char pub_key[32];
}CERT;

FILE *fp;
void hash(long *hasharr,void *ip,long leng)
{
    OPER step1[16]={ {0,1,2,3,0,7,0xd76aa478}, {3,0,1,2,1,12,0xe8c7b756},
                    {2,3,0,1,2,17,0x242070db}, {1,2,3,0,3,22,0xc1bdceee},
                    {0,1,2,3,4,7,0xf57c0faf}, {3,0,1,2,5,12,0x4787c62a},
                    {2,3,0,1,6,17,0xa8304613}, {1,2,3,0,7,22,0xfd469501},
                    {0,1,2,3,8,7,0x698098da}, {3,0,1,2,9,12,0x8b44f7af},
                    {2,3,0,1,10,17,0xffff5bb1}, {1,2,3,0,11,22,0x895cd7be},
                    {0,1,2,3,12,7,0x6b901122}, {3,0,1,2,13,12,0xfd987193},
                    {2,3,0,1,14,17,0xa679438e}, {1,2,3,0,15,22,0x49b40821} };
    OPER step2[16]={ {0,1,2,3,1,5,0xf61e2562}, {3,0,1,2,6,9,0xc040b340},
                    {2,3,0,1,11,14,0x265e5a51}, {1,2,3,0,0,20,0xe9b6c7aa},
                    {0,1,2,3,5,5,0xd62f105d}, {3,0,1,2,10,9,0x02441453},

```

```

                {2,3,0,1,15,14,0xd8a1e681}, {1,2,3,0,4,20,0x455a14ed},
                {0,1,2,3,9,5,0x21e1cde6}, {3,0,1,2,14,9,0xc33404b6},
                {2,3,0,1,3,14,0xf4b50d87}, {1,2,3,0,8,20,0x455a14ed},
                {0,1,2,3,13,5,0xa9e3e905}, {3,0,1,2,2,9,0xfcefa3f8},
                {2,3,0,1,7,14,0x676f02d9}, {1,2,3,0,12,20,0x8d2a4c8a} };
OPER step3[16]={ {0,1,2,3,5,4,0xfffa3942}, {3,0,1,2,8,11,0x8771f681},
                {2,3,0,1,11,16,0x6d9d6122}, {1,2,3,0,14,23,0xfde5380c},
                {0,1,2,3,1,4,0xa4beea44}, {3,0,1,2,4,11,0x4bdec5a9},
                {2,3,0,1,7,16,0xf6bb4b60}, {1,2,3,0,10,23,0xbebfb70},
                {0,1,2,3,13,4,0x289b7ee6}, {3,0,1,2,0,11,0xea127fa},
                {2,3,0,1,3,16,0xd4ef3085}, {1,2,3,0,6,23,0x04881d05},
                {0,1,2,3,9,4,0xd9d4d039}, {3,0,1,2,12,11,0xe6db99e5},
                {2,3,0,1,15,16,0x1fa27cf8}, {1,2,3,0,2,23,0xc4ac5665} };
OPER step4[16]={ {0,1,2,3,0,6,0xf4292244}, {3,0,1,2,7,10,0x432aff97},
                {2,3,0,1,14,15,0xab9423a7}, {1,2,3,0,5,21,0xfc93a039},
                {0,1,2,3,12,6,0x655b59c3}, {3,0,1,2,3,10,0x8f0ccc92},
                {2,3,0,1,10,15,0xffef47d}, {1,2,3,0,1,21,0x85845dd1},
                {0,1,2,3,8,6,0x6fa87e4f}, {3,0,1,2,15,10,0xfe2ce6e0},
                {2,3,0,1,6,15,0xa3014313}, {1,2,3,0,13,21,0xe0811a1},
                {0,1,2,3,4,6,0xf7537e82}, {3,0,1,2,11,10,0xbd3af235},
                {2,3,0,1,2,15,0x2ad7d2bb}, {1,2,3,0,9,21,0xeb86d391} };

```

```

unsigned int i,j,pad_bytes;
unsigned int arr[16];
unsigned int con[4]={0x01234567,0x89abcdef,0xfedcba98,0x7654321};
unsigned long len,k;
unsigned op1,op2,op3,op4,temp,res,v;
char string;
char *filler;
char y;
S s;
fp=fopen("message","w");
fwrite(ip,sizeof(char),leng,fp);
fclose(fp);
fp=fopen("message","ab");
len=ftell(fp);
pad_bytes=(len+4)%64;
if(pad_bytes!=0)
{
    pad_bytes=64-pad_bytes;
    filler=(char *)malloc(sizeof(char)*pad_bytes);
    bzero(filler,pad_bytes);
    fwrite(filler,sizeof(char),pad_bytes,fp);
}
fwrite(&len,sizeof(long),1,fp);
fclose(fp);
fp=fopen("message","rb");
while(!feof(fp))
{
    fread(arr,sizeof(int),16,fp);
    for(i=0;i<16;i++)
    {
        op1=con[step1[i].f1]; op2=con[step1[i].f2];
        op3=con[step1[i].f3]; op4=con[step1[i].f4];
        temp=(op1^f(op2,op3,op4)^arr[step1[i].I]^step1[i].t);
        v=temp<<(32-step1[i].s);
        temp=temp>>step1[i].s;
        temp=v;
        res=op2^temp;
        con[step1[i].f1]=res;
    }
    for(i=0;i<16;i++)
    {
        op1=con[step2[i].f1]; op2=con[step2[i].f2];
        op3=con[step2[i].f3]; op4=con[step2[i].f4];
        temp=(op1^g(op2,op3,op4)^arr[step2[i].I]^step2[i].t);
        v=temp<<(32-step2[i].s);

```

```

        temp=temp>>step2[i].s; templ=v; res=op2^temp;
        con[step2[i].f1]=res;
    }
    for(i=0;i<16;i++)
    {
        op1=con[step3[i].f1]; op2=con[step3[i].f2];
        op3=con[step3[i].f3]; op4=con[step3[i].f4];
        temp=(op1^h(op2,op3,op4)^arr[step3[i].l]^step3[i].t);
        v=temp<<(32-step3[i].s);
        temp=temp>>step3[i].s;
        templ=v;
        res=op2^temp; con[step3[i].f1]=res;
    }
    for(i=0;i<16;i++)
    {
        op1=con[step4[i].f1]; op2=con[step4[i].f2];
        op3=con[step4[i].f3]; op4=con[step4[i].f4];
        temp=(op1^i(op2,op3,op4)^arr[step4[i].l]^step4[i].t);
        v=temp<<(32-step4[i].s);
        temp=temp>>step4[i].s;
        templ=v;
        res=op2^temp;
        con[step4[i].f1]=res;
    }
}
hasharr[0]=con[0]; hasharr[1]=con[1]; hasharr[2]=con[2]; hasharr[3]=con[3];
printf("\nHashing completed on the packet...\n");
printf("Hash values generated: %u %u %u %u\n",con[0],con[1],con[2],con[3]);
fclose(fp);
}

```

config.h: This is the configuration file where all the configurations regarding ports, IP addresses and RSA keys are written.

```

#define SERV_PORT 5178
#define DEST_IP "127.0.0.1"
#define SERV_IP "127.0.0.1"
#define LISTENQ 1024
#define RSA_KEY_E 3
#define RSA_KEY_D 3051
#define RSA_KEY_N 4717
#define WRONG_RSA_KEY_E 4

```

makefile: This is the makefile which generates the client, server and snooper executable files

```

prog: server client snooper
server: server.o md5.o rsa.o
    @echo 'SERVER BEING MADE...'
    cc server.o md5.o rsa.o -lm -o server.out
client: client.o md5.o rsa.o
    @echo 'CLIENT BEING MADE...'
    cc client.o md5.o rsa.o -lm -o client.out
snooper:snooper.o md5.o rsa.o
    @echo 'SNOOPER BEING MADE...'
    cc snooper.o md5.o rsa.o -lm -o snooper.out
client.o:client.c
    @cc -c client.c
snooper.o:snooper.c
    @cc -c snooper.c
server.o:server.c
    @cc -c server.c
rsa.o:rsa.c
    @cc -c rsa.c -lm
md5.o: md5.c
    @cc -c md5.c
clean: @rm -f *.o *.out core

```

List of Publications

[Hota03a] Chittaranjan Hota and G. Raghurama, "A Heuristic Algorithm for Quality of Service Path Computation in Virtual Private Networks," *In Proceedings of Sixth International Conference on Information Technology (CIT 2003)*, Bhubaneswar, INDIA, 22nd to 25th December 2003, pp. 19-24.

[Hota03b] Chittaranjan Hota and G. Raghurama, "Building Virtual Private Networks to Support Mobile VPN Users in a Group with Quality of Service Guarantees," *In Proceedings of Third International Conference on Electronic Business (ICEB 2003)*, SINGAPORE, 9th to 13th December 2003, pp.616-618.

[Hota04a] Chittaranjan Hota, Sanjay Kumar Jha and G. Raghurama, "Distributed Dynamic Resource Management in IP Virtual Private Networks to Guarantee Quality of Service (invited paper)," *In Proceedings of 12th IEEE International Conference on Computer Networks (ICON 2004)*, SINGAPORE, 16th to 19th November 2004, pp. 414-419, ISBN 0-7803-8783-X, ISSN 1531-2216, IEEE Xplore.

[Hota04b] Chittaranjan Hota, Sanjay Kumar Jha, G. Raghurama and William Lau, "Bandwidth Guaranteed Restorable Paths in Multicast Virtual Private Networks," *In Proceedings of Asia-Pacific Conference on Parallel and Distributed Computing Technologies (ObComAPC 2004)*, Vellore, INDIA, December 2004, pp.77 to 97.

[Hota04c] Chittaranjan Hota, Sanjay Kumar Jha and G. Raghurama, "Restoration of Virtual Private Networks with Quality of Service Guarantees in the Pipe Model," *Sixth International Workshop on Distributed Computing (IWDC 2004)*, Kolkata, INDIA, 27th to 30th December 2004, Springer-Verlag Lecture Notes on Computer Science, LNCS 3326, pp. 288-301.

[Hota04d] Chittaranjan Hota and G. Raghurama, "Design and Deployment of IP Virtual Private Networks: A Case Study," *Seventh International Conference on Information Technology (CIT 2004)*, Hyderabad, INDIA, 20th to 23rd December 2004, Springer-Verlag Lecture Notes on Computer Science, LNCS 3356, pp. 76-86.

[Hota05a] Chittaranjan Hota, G. Raghurama, Sanjay Kumar Jha and William Lau, "Bandwidth Guaranteed Restorable Multicast Virtual Private Networks," *In Proceedings of Seventh IEEE International Conference on Personal Wireless Communications (ICPWC 2005)*, New Delhi, INDIA, 23rd to 25th January 2005, pp. 9-13.

[Hota05b] Chittaranjan Hota, Sanjay Kumar Jha and G. Raghurama, "Restoration of Virtual Private Networks with Quality of Service Guarantees in the Pipe Model," Published in *GESTS International Transaction on Computer Science and Engineering*, Vol.6 and No.1, Journal ISSN No: 1738-6438, May 30 2005, SunJin Publishing Co.

[Hota05c] Chittaranjan Hota, Sanjay Kumar Jha and G Raghurama, "Distributed Dynamic Management of Bandwidth-Assured IP Virtual Private Networks," Accepted in *International Journal of Network Management*, Wiley Publications, 2005.

Biography of the Supervisor

Prof. Raghurama G. is presently the Dean of Faculty Division-II and Admissions & Placement at the Birla Institute of Technology and Science (BITS), Pilani. Apart from teaching and research, he has a rich experience of educational administration.

Dr. Raghurama obtained his masters degree in Physics from IIT Chennai in 1980 and his doctorate from Indian Institute of Science, Bangalore in 1986. After a year of post doctoral work at IISc, he joined BITS, Pilani in 1987 where he has been involved in teaching, research and administration. He has published more than 20 research papers in national and international journals in the areas of solid state physics, optics, communication networks, web caching, computer networks etc. He presently teaches courses on telecommunications and network management to undergraduate and graduate students at BITS.

Biography of the Candidate

Chittaranjan Hota has his Bachelors in Computer Engineering from Amravati University (MS), India with first division in the year 1990 and M.E in Computer Science and Engineering from Thapar Institute of Engineering and Technology (Deemed), Patiala with first class distinction in the year 1998. He has a teaching experience of over 15 years to undergraduate and graduate computer engineering students at various Indian universities (Utkal, Orissa; Amravati, MS and B.I.T.S, Pilani). Currently he is working as an Assistant Professor in Computer Science and Information Systems group at B.I.T.S, Pilani. His areas of interest include Dynamic Network Provisioning, Quality of Service in VPNs, Distributed Systems Development, Artificial Intelligence, and Systems Programming. He has carried out part of this research work at University of New South Wales, Sydney, Australia from Feb 2004 to June 2004. He has presented an invited research paper at IEEE ICON 2004, Singapore organized by NUS, Singapore during November 2004.