Dynamic Inter-Domain Distributed Computing

Yiran Gao

SUBMITTED FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

School of Electronic Engineering and Computer Science,

Queen Mary, University of London, United Kingdom

December 2008

Acknowledgements

I would like to thank my supervisor Dr Chris Phillips for all his help and continuous encouragement. I cannot complete this thesis without him. I would also like to thank my Masters supervisor Dr Jintong Lin for his love and care during the years of my study in London and Beijing.

In addition, I would to use this opportunity to say thank you to the academic members of staff in the Department of Electronic Engineering, Queen Mary, whom I have been interacting with, for their comments and suggestions.

In addition, I am thankful to Dr Liwen He, who has given me a lot of help both on an academic and personal level. My PhD has been partly supported by British Telecom (BT), which I would like to acknowledge.

I am also grateful to everyone in the Department for making my days at Queen Mary memorable, both during working hours and when relaxing.

Finally, my love and gratitude goes to my family, especially my parents, Xiaoming Gao and Jinping Kang, and my wife Shan Yang. They are the most important persons in my life; all my success also belongs to them.

Abstract

This research proposes a new service architecture that extends the use of a Multi-Protocol Label Switching (MPLS) infrastructure for the formation and operation of dynamic communities, incorporating value-added resources allied to the tasks being performed. The approach introduces a new operator-owned control-plane entity called the Dynamic Virtual Private Network Manager (DVM) for managing customer communities and liaising with the operator's infrastructure. The communities take the form of dynamic VPNs that can be used to support extranet-based on-demand services compatible with "grid computing" and other distributed, collaborative activities, enabling automated business processes to dynamically request communication infrastructure and processing resources.

A test-bed has been built to confirm the satisfactory operation of the dynamic VPN (DVPN) processes involved with the on-demand creation of Label Switched Paths based on different performance criteria. In addition, a novel Genetic Algorithm (GA) based dynamic resource allocation scheme is proposed to assign resources within the DVPN system. Its performance is critically assessed through simulation, being compared with a standard random assignment scheme as well as other GA variants. The results demonstrate that this algorithm is feasible, effective and robust in the DVPN scenarios considered.

In addition to the inter-domain negotiation mechanism provided by the DVPN architecture, a credit-based resource allocation mechanism is proposed to stimulate cooperation among autonomous systems participating in the DVPN framework. Several policy distributions are put forward and compared in a simulation environment, using different traffic models. Results demonstrate that the usage efficiency of the overall multi-domain system is improved greatly by introducing this mechanism.

Table of Contents

ACKNOWLEDGEMENTS **2**

ABSTRACT **3**

TABLE OF CONTENTS **4**

LIST OF FIGURES **7**

LIST OF TABLES **11**

GLOSSARY **12**

CHAPTER 1 INTRODUCTION **14**

1.1 MOTIVATION AND OBJECTIVES 14
1.2 CONTRIBUTIONS OF THE THESIS 14
1.3 STRUCTURE OF THE THESIS 15

CHAPTER 2 BACKGROUND **17**

2.1 MULTIPLE-PROTOCOL LABEL SWITCHING 17
2.2 BORDER GATEWAY PROTOCOL 19
2.3 DISTRIBUTION OF MPLS LABELS WITH BGP 21
2.4 VIRTUAL PRIVATE NETWORKS 22
2.5 BGP/MPLS VPNS 24
2.6 MPLS VPN SECURITY 26
2.7 INTER-DOMAIN CONNECTIVITY 28
2.7.1 Inter-domain Issues and Shortcomings of BGP *28*
2.7.2 Related Research on BGP *29*
2.7.3 Trusting Routing in Internet *31*
2.7.4 Routing in Mobile, Ad Hoc Network *32*
2.8 RESEARCH ON DYNAMIC VPNS 32
2.9 RESOURCE SCHEDULING 37
2.10 OPTIMISATION ALGORITHMS FOR RESOURCE SCHEDULING 41
2.10.1 Genetic Algorithm *41*
2.10.2 Particle Swarm Optimisation *44*
2.10.3 Comparison between PSO and GA *46*
2.11 SUMMARY 46

CHAPTER 3 MOTIVATION AND REQUIREMENTS FOR INTER-PROVIDER DYNAMIC VPNS **48**

3.1 NEW APPLICATIONS 48
3.2 EXISTING VPN APPROACHES AND LIMITATIONS 52
3.3 DYNAMIC RESOURCE SCHEDULING 53
3.4 INTER-DOMAIN COOPERATION 55

3.5 AIMS OF THE PROPOSED INTER-PROVIDER DYNAMIC VPN ARCHITECTURE 56

CHAPTER 4 INTER-PROVIDER DYNAMIC VPN FRAMEWORK 58

4.1 MOTIVATING EXAMPLE: GLOBAL GRID COMPUTING 58
4.2 THE INTER-PROVIDER DYNAMIC VPN ARCHITECTURE 60
4.3 DYNAMIC VPN MANAGER 62
4.3.1 DVM Functions 62
4.3.2 The Internal Architecture of the DVM 63
4.3.3 DVM Connectivity Management 66
4.3.4 DVM Resource Management 67
4.3.5 Inter-Domain VPN Support 70
4.3.6 Inter-domain Path Selection 71
4.4 DYNAMIC VPN PROCESSES 73
4.5 VPN BASIC ACTIONS 75
4.5.1 VPN Building Action 75
4.5.2 Join VPN Action 78
4.5.3 Leave VPN Action 79
4.5.4 Remove VPN Action 80
4.5.5 Resource Booking Action 81
4.6 ORCHESTRATED COMPUTING 82
4.6.1 Ticket Identifiers 82
4.6.2 DVPN Message Types 82
4.6.3 VPN Link Setup 86
4.6.4 User Task Allocation to Resources 87
4.6.5 Scenario Illustration 90
4.7 SEPARATION OF VPN CONTROL PLANES 94
4.8 MOTIVATING EXAMPLE REVISITED: DVPN-ENABLED GLOBAL GRID COMPUTING 95
4.9 NOVELTY AND CONTRIBUTION 96

CHAPTER 5 DYNAMIC VPN PROTOTYPE 98

5.1 THE OBJECT OF THE TEST-BED 98
5.2 MPLS FORWARDING AND AUTOMATIC CONTROL 98
5.3 TEST-BED DEMONSTRATION 100
5.4 SUMMARY 103

CHAPTER 6 DYNAMIC JOB SCHEDULING 104

6.1 DYNAMIC RESOURCE SCHEDULING ALGORITHM ARCHITECTURE 104
6.2 OPTIMISATION WITH GA AND PSO 108
6.3 GA OPTIMISATION ENGINE IMPLEMENTATION 109
6.3.1 GA Chromosome 109
6.3.2 Computing the Fitness of Chromosome Fitness 109
6.3.3 Crossover Process 110

6.3.4 Mutation Process 111
6.3.5 Evolution Process 111
6.3.6 Simulation Scenarios 111
6.3.7 Simulation Results Analysis 112
6.3.8 Comparison with a Random Scheduler 120
6.4 ALTERNATIVE PSO OPTIMISATION ENGINE IMPLEMENTATION 123
6.4.1 Particle Architecture 123
6.4.2 Performance Evaluation 124
6.4.3 Swarm Parameters 127
6.5 INTER-DOMAIN RESOURCE ALLOCATION 130
6.5.1 Simulation Scenario 131
6.5.2 Fitness Function 131
6.5.3 Simulation Results 132
6.6 VERIFICATION 134
6.6.1 Verification of the Algorithm Architecture 134
6.6.2 Behaviour Verification of the Algorithm for Particular Scenarios 136
6.6.3 Verification of the Random Number Generator 139
6.7 SUMMARY 140

CHAPTER 7 THE CREDIT-BASED INTER-DOMAIN MPLS MECHANISM 141

7.1 THE CREDIT BASED INTER-AS MANAGEMENT MECHANISM 141
7.2 SIMULATION SCENARIO 144
7.3 DECISION POLICY 145
7.4 PERFORMANCE ANALYSIS 150
7.5 PAYMENT/CHARGE RATE 152
7.6 SUMMARY 154

CHAPTER 8 DISCUSSION, CONCLUSIONS AND FURTHER WORK 155

8.1 DISCUSSION 155
8.2 SCALABILITY ANALYSIS 157
8.2.1 DVPN Control Messages 157
8.2.2 VPN Action Performance for Various Scenarios 160
8.2.3 Number of Tasks versus Computation Time 162
8.2.4 Summary 163
8.3 CONCLUSIONS 163
8.4 FURTHER WORK 164

REFERENCES 166

List of Figures

FIGURE 2.1 THE MPLS LABEL FORMAT 18
FIGURE 2.2 LSP SETUP SIGNALLING 19
FIGURE 2.3 MPLS FORWARDING EXAMPLE 19
FIGURE 2.4 EXAMPLE OF EBGP AND IBGP [22] 20
FIGURE 2.5 HOW BGP PROPAGATES A ROUTE [22] 21
FIGURE 2.6 HOW MPLS WORKS WHEN TWO BGP PEERS ARE NOT DIRECTLY ADJACENT 22
FIGURE 2.7 VPN EXAMPLE [31] 23
FIGURE 2.8 FULLY MESH VPN EXAMPLE [32] 24
FIGURE 2.9 BGP/MPLS/VPN ARCHITECTURE 25
FIGURE 2.10 AN EXAMPLE OF IPSEC OVER AN MPLS VPN 27
FIGURE 2.11 THE SPRITE ARCHITECTURE [69] 32
FIGURE 2.12 THE CUG ARCHITECTURE [77] 33
FIGURE 2.13 DYNAMIC CUG BASED ON OVERLAY NETWORK [78] 34
FIGURE 2.14 DVC DYNAMIC VPN ARCHITECTURE [80] 36
FIGURE 2.15 DRDC DVC NETWORK SET-UP [81] 36
FIGURE 2.16 CM MANAGED DYNAMIC VPN ARCHITECTURE 37
FIGURE 2.17 A DIRECTED ACYCLIC GRAPH EXAMPLE 38
FIGURE 2.18 THE HYBRID REMAPPER ALGORITHM ARCHITECTURE [96] 39
FIGURE 2.19 EXAMPLE OF RESOURCE ESTIMATION [93] 40
FIGURE 2.20 THE CROSSOVER PROCESS 42
FIGURE 2.21 GA FLOW CHART 43
FIGURE 2.22 PARTICLE MOVEMENT AT ITERATION T+1 WITH THE INFLUENCE OF PBEST AND GBEST 45
FIGURE 2.23 PSO FLOW CHART 45
FIGURE 3.1 GRID COMPUTING EXAMPLE [114] 49
FIGURE 3.2 AN SOA E-BUSINESS SCENARIO [120] 51
FIGURE 3.3 LOAN PROCESS 51
FIGURE 3.4 DVPN JOBS SCENARIO EXAMPLE 54
FIGURE 3.5 DYNAMIC VPN ENVIRONMENT EXAMPLE 55
FIGURE 3.6 INTER-DOMAIN CONNECTIVITY 56
FIGURE 4.1 GLOBAL GRID COMPUTING EXAMPLE 58
FIGURE 4.2 DYNAMIC VPN ARCHITECTURE WITH DVM 60
FIGURE 4.3 UNDERLYING NETWORK 61
FIGURE 4.4 THE FUNCTIONS OF THE DVM 62
FIGURE 4.5 THE DVM ARCHITECTURE 63
FIGURE 4.6 THE USER DATABASE 63
FIGURE 4.7 THE JOB DATABASE 64
FIGURE 4.8 THE RESOURCE DATABASE 64
FIGURE 4.9 THE DVM DATABASE 65
FIGURE 4.10 THE VPN DATABASE 65

FIGURE 4.11 RESOURCE MANAGEMENT ARCHITECTURE.......68
FIGURE 4.12 HOW THE DVM MAKES VPN POLICY.......70
FIGURE 4.13 INTER-DOMAIN LSP CREATION.......71
FIGURE 4.14 INTER-AS ROUTING MANAGEMENT.......72
FIGURE 4.15 EXAMPLE MULTI-DOMAIN DVPN SCENARIO.......73
FIGURE 4.16 THE SERVICE APPLICATION MESSAGE FORMAT.......73
FIGURE 4.17 VPN BUILDING PROCESS.......75
FIGURE 4.18 THE DVM'S SERVICE REQUEST MESSAGE FORMAT.......76
FIGURE 4.19 THE VPN ID FORMAT.......76
FIGURE 4.20 DVM COMMAND MESSAGE FORMAT.......76
FIGURE 4.21 JOIN VPN ACTION.......78
FIGURE 4.22 LEAVE VPN ACTION.......79
FIGURE 4.23 REMOVE VPN ACTION.......80
FIGURE 4.24 DVM BOOKING NETWORK RESOURCE FOR USERS.......81
FIGURE 4.25 THE VPN LINK SETUP.......87
FIGURE 4.26 INTRA-DOMAIN.......88
FIGURE 4.27 INTER-DOMAIN.......89
FIGURE 4.28 DAG PROCESSING IN THE DVPN SYSTEM.......90
FIGURE 4.29 THE DVPN CONTROL PLANES.......95
FIGURE 4.30 A DYNAMIC VPN EXAMPLE.......96
FIGURE 5.1THE ARCHITECTURE OF THE TEST-BED.......98
FIGURE 5.2 THE ARCHITECTURE OF MPLS CONTROLLING TOOL.......99
FIGURE 5.3 LABEL ADVERTISEMENT EXAMPLE.......100
FIGURE 5.4 DYNAMIC VPN TEST SCENARIO AND THE ACTUAL TEST-BED.......101
FIGURE 5.5 TEST-BED TOPOLOGY.......101
FIGURE 5.6 THE PROTOTYPING TEST-BED.......102
FIGURE 6.1 THE ORCHESTRATED COMPUTING EXAMPLE.......105
FIGURE 6.2 THE GA-BASED RESOURCE SCHEDULING ALGORITHM.......106
FIGURE 6.3 A NODE-STANDARD DAG.......107
FIGURE 6.4 EXAMPLE JOB SCHEDULING ARRANGEMENT.......108
FIGURE 6.5 ORIGINAL PARENT CHROMOSOMES.......110
FIGURE 6.6 CROSSOVER.......110
FIGURE 6.7 NEW CHILD CHROMOSOMES.......110
FIGURE 6.8 JOB SCENARIO EXAMPLE.......112
FIGURE 6.9 OVERALL COMPLETION TIME PERFORMANCE.......113
FIGURE 6.10 IMPACT OF POPULATION SIZE JOB ON COMPLETION TIME PERFORMANCE-WITH A FIXED CHROMOSOME POPULATION SIZE.......114
FIGURE 6.11 ADAPTIVE CHROMOSOME POPULATION SCHEME.......115
FIGURE 6.12 IMPACT OF GA PARAMETERS ON OVERALL JOB COMPLETION TIME PERFORMANCE.......116
FIGURE 6.13 RELATIONSHIP BETWEEN OVERALL JOB COMPLETION TIME PERFORMANCE AND COMPUTATION TIME.......118
FIGURE 6.14 COMPLETION TIME WITH DIFFERENT SURVIVAL RATES.......119

FIGURE 6.15 COMPLETION TIME PERFORMANCE AND MUTATION RATE 120
FIGURE 6.16 PERFORMANCE COMPARISON OF PROPOSED GA SCHEME AGAINST A RANDOM SCHEDULER 122
FIGURE 6.17 COMPARISON BETWEEN GA AND RANDOM SCHEDULER IN PARTICULAR SCENARIOS 123
FIGURE 6.18 A PARTICLE EXAMPLE 124
FIGURE 6.19 PERFORMANCE COMPARISON OF STATIC GA AND STATIC PSO RESOURCE SCHEDULING ALGORITHMS 125
FIGURE 6.20 RELATIONSHIP BETWEEN COMPLETION TIME AND COMPUTATION TIME 126
FIGURE 6.21 COMPUTATION TIME COMPARISON BETWEEN GA AND PSO 127
FIGURE 6.22 RELATIONSHIP BETWEEN PARTICLE INERTIA (C0) AND PSO PERFORMANCE 128
FIGURE 6.23 RELATIONSHIP BETWEEN GLOBAL VELOCITY (C1) AND PSO PERFORMANCE 129
FIGURE 6.24 RELATIONSHIP BETWEEN LOCAL VELOCITY (C2) AND PSO PERFORMANCE 130
FIGURE 6.25 TWO-DOMAIN SCENARIO 131
FIGURE 6.26 TASK DISTRIBUTION IN THE SCENARIO WITHOUT INTER-DOMAIN COSTS 133
FIGURE 6.27 TASK DISTRIBUTION IN THE SCENARIO WHERE THE INTER-DOMAIN COST IS CONSIDERED (B=1) 133
FIGURE 6.28 RELATIONSHIP BETWEEN OVERALL COMPLETION TIME AND COST FACTOR “B” 134
FIGURE 6.29 VERIFICATION JOB SCENARIO 135
FIGURE 6.30 SCHEDULING ARRANGEMENT 135
FIGURE 6.31 COMPLETION TIME PERFORMANCE FOR A SIMPLE TANDEM TASK JOB SCENARIO 136
FIGURE 6.32 COMPLETION TIME PERFORMANCE FOR AN INDEPENDENT JOB SCENARIO 137
FIGURE 6.33 COMPLETION TIME PERFORMANCE SCENARIOS WITH DIFFERENT TASKS RELATIONSHIPS BUT THE SAME SCENARIO SIZE 138
FIGURE 6.34 RESOURCE CAPACITY VERIFICATION 139
FIGURE 6.35: DISTRIBUTION OF RANDOM NUMBERS 139
FIGURE 7.1 EXAMPLE OF HOW THE CREDIT BASED INTER-DOMAIN MANAGEMENT MECHANISM WORKS 141
FIGURE 7.2 THE CREDIT BASED INTER DOMAIN MANAGEMENT ARCHITECTURE 142
FIGURE 7.3 THE CREDIT CHARGE/PAYMENT PROCESS 144
FIGURE 7.4 EXAMPLE SIMULATION SCENARIO 144
FIGURE 7.5 RELATIONSHIP BETWEEN TRANSIT SERVICE ACCEPTANCE PROBABILITY, AVAILABLE BANDWIDTH AND OWNED CREDIT. 146
FIGURE 7.6 RELATIONSHIP BETWEEN TRANSIT SERVICE ACCEPTANCE PROBABILITY AND THE CREDIT FACTOR 146
FIGURE 7.7 RELATIONSHIP BETWEEN TRANSIT SERVICE ACCEPTANCE PROBABILITY AND THE BANDWIDTH FACTOR 147
FIGURE 7.8 RELATIONSHIP BETWEEN TRANSIT SERVICE ACCEPTANCE PROBABILITY, THE CREDIT AND AVAILABLE BANDWIDTH 148
FIGURE 7.9 RELATIONSHIP BETWEEN TRANSIT SERVICE ACCEPTANCE PROBABILITY AND BANDWIDTH LOWER LIMIT 149
FIGURE 7.10 RELATIONSHIP BETWEEN TRANSIT SERVICE ACCEPTANCE PROBABILITY AND CREDIT UPPER LIMIT 150
FIGURE 7.11 RELATIONSHIP BETWEEN THE MEAN AVAILABLE BANDWIDTH AND THE TRANSIT SERVICE ACCEPTANCE PROBABILITY 151
FIGURE 7.12 RELATIONSHIP BETWEEN TRANSIT SERVICE ACCEPTANCE PROBABILITY AND DIRECT CONNECTION PROBABILITY 152
FIGURE 7.13 RELATIONSHIP BETWEEN TRANSIT SERVICE ACCEPTANCE PROBABILITY AND PAYMENT/CHARGE RATE 154
FIGURE 8.1 SCENARIO EXAMPLES 158
FIGURE 8.2 MEAN VPN ACTIONS VERSUS MEAN NUMBER OF DEPENDENT UPPERS 160

FIGURE 8.3 VPN ACTIONS VS TOTAL TASKS NUMBER 161
FIGURE 8.4 MEAN VPN ACTIONS VERSUS NUMBER OF RESOURCES 162
FIGURE 8.5 MEAN COMPUTATION TIME FOR VARIOUS NUMBERS OF TASKS 163

List of Tables

TABLE 2.1 EXAMPLE LIB TABLE 18
TABLE 4.1 SERVICE TYPES AND SERVICE ATTRIBUTES 74
TABLE 4.2 THE COMMAND TYPES AND COMMAND ATTRIBUTES 76
TABLE 4.3 VPN LABEL FORWARDING TABLE EXAMPLE 87
TABLE 6.1 THE CHROMOSOME ARCHITECTURE 109
TABLE 6.2 THE RELATIONSHIP BETWEEN THE TASKS AND THE RESOURCES REPRESENTED BY TABLE 6.1 109
TABLE 6.3 THE RELATIONSHIP BETWEEN THE TASKS AND THE RESOURCES REPRESENTED IN FIGURE 6.18 124
TABLE 6.4 THE RESOURCES 131
TABLE 6.5 ALGORITHM PROCESS TRACKING 135
TABLE 7.1 THE JOB FORMAT 144
TABLE 7.2 THE DECISION SCHEME 148
TABLE 7.3 THE BANDWIDTH UTILIZATION UNDER DIFFERENT TRANSIT SERVICE ACCEPTANCE RATES 151
TABLE 8.1 NUMBER OF CONTROL MESSAGES PRODUCED 159

Glossary

ACK	Acknowledgement
ASBR	AS Border Router
ATM	Asynchronous Transfer Mode
AS	Autonomous System
BW	Bandwidth
BGP	Border Gateway Protocol
BGP-4	Border Gateway Protocol Version 4
BT	British Telecom
BPEL	Business Process Execution Language for Web Services
CA	Certificate Authority
CCS	Credit Clearance Service
CUG	Closed User Group
CM	Community Manager
CR	Computational Resources
CR-LDP	Constraint-Based Label Distribution Protocol
RSVP-TE	Resource Reservation Protocol-Traffic Extension
CME	Connection Management Entity
CPA	Critical Path Analysis
CIM	Credit Based Inter-AS Management
CE	Customer Edge
CRM	Customer Relationship Management
DS	Data Storage
DRDC	Defence R&D Canada
DREnet	Defence Research Establishment Network
DRDC	Defence Research & Development Canada
DoS	Denial of Service
DAG	Directed Acyclic Graph
DNS	Domain Name System
DLAR	Dynamic Load-Aware Routing
DVC	Dynamic Virtual Private Network Controller
DVPN	Dynamic VPN
DVM	Dynamic VPN-Manager
e-Commence	electronic commerce
ERP	Enterprise Resource Planning
eBGP	external/exterior BGP
FEC	Forwarding Equivalence Class
FR	Frame Relay
gbest	global best vector
G-MPLS	Generalise MPLS
GA	Genetic Algorithm
ICSF	Innermost Control Structure First
CIM	Inter-AS Management
iBGP	internal/interior BGP
IETF	The Internet Engineering Task Force
IGP	Interior Gateway Protocol
IP	Internet Protocol
ISP	Internet Service Providers
JNI	Java Native Interface
JS	Jobs Set
LDP	Label Distribution Protocol
LSPs	Label-Switched Paths
LER	Label Edge Router
LIB	Label Information Base
LBAR	Load-Balanced Ad hoc Routing

LS	Local Search
LSR	Label Switch Router
MBGP	Multiprotocol Extension to BGP4
MET	Minimum Execution Time
MANET	Mobile, ad hoc network
MPLS	Multi-Protocol Label Switching
MCT	MPLS Controlling Tool
NHLFE	Next Hop Label Forwarding Entry
NLRI	Network Layer Reachablity Information
NNI	Network-Network Interface
OSPF	Open Shortest Path First
OLB	Opportunistic Load Balancing
P	Provider
PPM	Packet Purse Model
PTM	Packet Trade Model
PSO	Particle Swarm Optimisation
pbest	personal best vector
PRI	Priority
PIFA	Protocol-Independent Fairness Algorithm
PE	Provider Edge
PKI	Public Key Infrastructure
QoS	Quality-of-Service
RTS	Ready Tasks Set
RM	Resource Manager
RSVP	Resource ReSerVation Protocol
RS	Resource Set
RFC	IETF-Request For Comments
RIP	Routing Information Protocol
s-BGP	secure Border Gateway Protocol
SAFI	Subsequent Address Family Identifiers
SLA	Service Level Agreement
SOA	Service Oriented Architecture
SLA	Simple Load-balancing Approach
SNMP	Simple Network Management Protocol
SOAP	Simple Object Access Protocol
SA	Simulated Annealing
SSL	Secure Sockets Layer
TS	Tabu Search
TCP	Transmission Control Protocol
TSP	Trust Service Providers
TTL	Time To Live
UNI	User Network Interface
VC	Virtual Circuit
VPN	Virtual Private Network
VRF	Virtual Routing and Forwarding Table
WSDL	Web Service Definition Language

Chapter 1 Introduction

1.1 Motivation and Objectives

A Virtual Private Network (VPN) is a communications environment in which access is controlled to permit peer connections only within a defined community of interest, and is constructed through some form of partitioning of an underlying communications medium, where this medium provides services to the network on a non-exclusive basis[1][2]. With the worldwide success of IP-MPLS network deployment, inter-connecting multiple provider IP-MPLS networks for global reachability becomes the next important step. Many providers have implemented MPLS [3] inter-provider connections. The main drivers for inter-provider IP-MPLS services with QoS (quality-of-service) guarantees are inter-provider IP VPNs [4][5][6]. RFC2547 [5] defines the most mature MPLS VPN solution, commonly referred to as BGP/MPLS VPNs [4]. They allow a service provider to use its Internet Protocol (IP) backbone to provide an IP VPN service to customers. The service provider uses BGP to exchange VPN routes. An MPLS label is assigned to each VPN route, the process of label distribution is carried out using BGP (by piggybacking the label information on the BGP messages). Whilst distributing reachability information, BGP also distributes MPLS labels for that path. RFC2547 provides a scalable approach to support inter-domain VPNs. Draft-ieft-ppvpn-rfc2547bis-01 [7] describes three approaches to establish inter-provider connectivity.

VPNs [8] are typically established using manual intervention to configure the various network resources. This is acceptable for long-term connectivity – such as inter-connection between enterprise campuses. However, new forms of e-business [8][9] and "grid computing" [10][11][12] can have resource usage demands that do not fit well with this arrangement. In these cases it is desirable to establish community relationships that may last minutes, or hours through to weeks. Moreover, these new applications require QoS guarantees when the VPN spans AS. Also the VPN processes and network resources should be allocated on demand. It is necessary that a VPN can be configured based on a customer's requirements and the network status. Although recent research has been devoted to intra-domain protection and restoration mechanisms, limited work has addressed these factors in an inter-domain context [13]. There remain key challenges for VPN services to provide security, QoS guarantees, reliability, dynamic operation and efficient inter-domain operations.

1.2 Contributions of the Thesis

To extend the flexibility of VPNs in terms of connectivity with QoS support, a new functional element is proposed named the Dynamic VPN-Manager (DVM) [14]. By locating DVMs in separate AS, inter-AS dynamic VPNs become possible. With the help of DVMs, activities that are hitherto performed manually are augmented and even replaced by an automated infrastructure capable of inter-operator negotiation as well as supervising local configuration management.

The DVM is responsible for scheduling and reserving communications and processing resources for complex jobs expressed as a workflow. One of the key challenges remaining is how to schedule job allocation to the underlying resources. The author carried out an in-depth study of this problem. A dynamic resource scheduling algorithm is proposed to perform resource scheduling in DVPN job scenarios where either Genetic Algorithm (GA) or Particle Swarm Optimisation (PSO) are employed as optimisation algorithms. The performance of the algorithms is studied through simulations and the results demonstrate that both algorithms are feasible and efficient in the DVPN job scenarios considered.

VPNs spanning multiple-domains face the challenge of reliability and QoS constraints. The DVPN architecture also exhibits similar challenges. For this reason, the author also proposes a novel credit-based Inter-AS Management (CIM) mechanism to make the DVPN more reliable and efficient by stimulating cooperation among ASes.

Relevant publications by the author are listed as follows:

1. Yiran Gao, Chris Phillips, Liwen He, "DVM Based Dynamic VPN Architecture for Group Working and Orchestrated Distributed Computing", Third IEEE International Conference on Digital Information Management (ICDIM 2008), London, November 2008.
2. Y. Gao, C. Phillips, "A GA Based Real-time Resource Scheduling Algorithm", IEEE ICTTA08, Syria, April 2008.
3. Yiran Gao, Chris Phillips, John Bigham, "Dynamic VPN Architecture for Group Working and Orchestrated Distributed Computing", London Communications Symposium, September 2006
4. Yiran Gao, Chris Phillips, "Dynamic MPLS/IP VPN with DVM", WGN5: Workshop in G/MPLS Networks, Spain, March 2006. (Invited paper)
5. Yiran Gao, Chris Phillips, "Inter-Provider Dynamic VPNs", The 6th Annual PostGraduate Symposium on The Convergence of Telecommunications, Networking and Broadcasting, Liverpool, UK, June 2005.

1.3 Structure of the Thesis

The rest of the thesis is organised as follows:
Chapter 2 provides the background in relation to the key technologies of inter-domain MPLS VPNs. After introducing MPLS (Multi-Protocol Label Switching) and VPNs (Virtual Private Networks), this chapter describes the RFC 2457 BGP/MPLS/VPN and the existing solutions for inter-domain connectivity. Finally, the resource scheduling problem is studied. Two heuristic optimisation mechanisms- GA and PSO are introduced in detail as candidate optimisation tools for the proposed resource scheduling algorithm.

Chapter 3 provides an explanation of the motivation for the work by placing it in the context of new services. Emerging applications such as e-business and Global Grid Computing are introduced. The limitations of existing technologies and approaches are analysed.

Chapter 4 describes the new service architecture that extends the use of a Multi-Protocol Label Switching (MPLS) infrastructure for the formation and operation of dynamic communities, incorporating value-added resources allied to the tasks being performed. The approach introduces a new operator-owned control-plane entity called the Dynamic VPN Manager (DVM) for managing customer communities and liaising with the operator's infrastructure. The communities take the form of dynamic VPNs that can be used to support extranet-based on-demand services compatible with "grid computing" and other distributed, collaborative activities. The DVPN architecture is introduced in detail; the basic DVPN processes (build/remove, join/leave) are also analysed.

Chapter 5 details the DVPN prototype. A small test scenario was successfully implemented on eight PCs, representing VPN user entities, CE, PE and P routers as well as a DVM. The details of the test-bed arrangement are given in this chapter.

Chapter 6 proposes a novel resource scheduling mechanism using two alternative optimisation engines. GA and PSO are employed as the optimisation algorithms in this scheme. Its performance is studied through simulations with various randomly generated job scenarios. The simulation results show that this mechanism is both feasible and efficient for use in dynamic VPN (DVPN) scenarios. The comparative performance of PSO and GA approaches are also examined through simulations.

Chapter 7 describes a novel credit-based inter-domain management (CIM) mechanism to stimulate cooperation among AS operators without requiring sensitive domain information to be divulged. The architecture of this proposed CIM mechanism is introduced and its performance is studied through simulations. Results demonstrate that the overall inter-network resource utilisation rate is improved after introducing CIM.

Chapter 8 discusses the thesis and provides the conclusions and pointers to future work.

Chapter 2 Background

BGP/MPLS VPS [15][16] has received a lot of attention, and it is designed to provide QoS guaranteed, reliable and secure inter-domain connectivity service over IP network. In this chapter, the technologies related to the inter-domain VPNs are analysed. In addition, new applications (i.e. grid-computing) to harness distributed resources in a network for solving problems or supporting group-working activities are arising. However resource scheduling in a heterogonous environment (i.e. grid computing) is a well-known NP-complete problem [17][18]. The author reviews the research works on this field in this chapter.

2.1 Multiple-Protocol Label Switching

Multi-protocol Label Switching (MPLS) [3] is an architecture for fast packet switching and routing. It provides the designation, routing, forwarding and switching of traffic flows through the network. More specifically, MPLS has mechanisms to manage traffic flows of various granularities. MPLS is independent of the layer 2 and layer 3 [16] such as ATM (Asynchronous Transfer Mode) and IP respectively. MPLS provides a means to map IP addresses onto simple, fixed-length labels used by different forwarding and switching technologies. MPLS interfaces to existing routing and switching protocols, such as IP, ATM, Frame Relay, Resource ReSerVation Protocol (RSVP) and Open Shortest Path First (OSPF) [19].

In MPLS, data transmission occurs along Label-Switched Paths (LSPs). LSPs are a sequence of labels at each and every node along the path from the source to the destination [3]. A label in the simplest form identifies the path a packet should traverse. A label may be carried in a layer-2 header[1] along with the packet. The receiving label switching router examines the packet for its label to determine the next hop, which means that a labelled packet is forwarded based on label switching along a LSP. In MPLS labels are bound to a FEC (Forwarding Equivalence Class) which is a representation of a group of packets with the same requirements for transport (i.e. the destination, traffic engineering, virtual private network, QoS). All the packets of a same FEC are treated in the same way and the assignment of a packet to a particular FEC is done just once as the packet enters the MPLS network. The label can be embedded in the header of the data link layer (ATM VCI/VPI) or in the shim between the layer 2 header and layer 3 header). The MPLS shim consists of 4 segments [3]:

- A 20-bit label value.
- A 3-bit field for QoS priority.
- A 1-bit bottom of stack flag representing whether the current label is the last in the stack.
- An 8-bit TTL (time to live) field.

Figure 2.1 illustrates the label format of the MPLS shim.

[1] Not necessary, it is generally carried in a layer 2.5 shim header located partially of fully within the L2 header/payload.

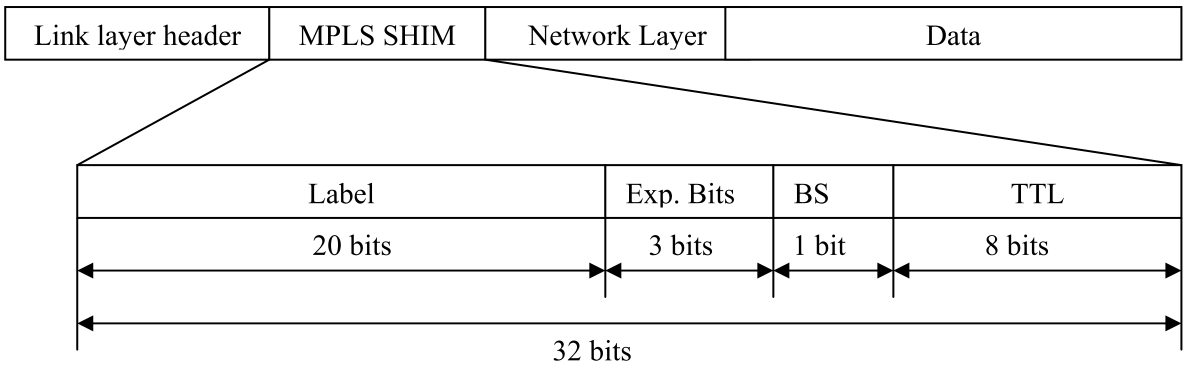


Figure 2.1 The MPLS Label Format

The MPLS domain consists of MPLS-devices. In an MPLS domain, an MPLS Path –LSP (Label Switch Path) is created for a packet to travel in accordance with an FEC. The LSP is unidirectional, so the return traffic must travel along another LSP. There are several label distribution protocols being used today, such as Label Distribution Protocol (LDP) or RSVP-TE [19] or piggybacked on routing protocols like Border Gateway Protocol (BGP) [20] and OSPF.

The MPLS router at the network edges is called a LER (Label Edge Router) and the MPLS routers within the MPLS domain are called LSRs (Label Switch Routers). An MPLS router routes the labelled packets according to the LIB (Label Information Base). The LIB is created in accordance with the label distribution signalling. Table 2.1 is an example LIB. In this instance, if the packets are coming in through incoming port 1 with label 1, the label will be swapped with label 6 and the packets will be sent out to out via port 3. If the incoming packets arrive via port 2 with label 9, the label will be popped and the packets will be forwarded as pure IP packets; in this case, this router is the egress of the LSP.

Table 2.1 Example LIB Table

Input Port	Incoming Port Label	Output Port	Outgoing Port label
1	1	3	6
2	9	--	--

An MPLS router typically requests a label from its downstream hop to bind to a specific FEC and the MPLS router responds to a label request by sending a label to the upstream initiator. For example, in the scenario shown in Figure 2.2, the LER router A initiates a label request to the LSR router B for a particular FEC, the router B sends a label request to LER router C which is the egress of the this LSP. After receiving the label request from B, router C assigns a label for this LSP according to the FEC and sends this label back to B. The LER C also creates an entry of this label binding in the LIB. Once B receives the label from C, it also assigns a label for this LSP and sends it back to A while creating an entry of this label binding in its LIB. Finally, this LSP is created after a receiving label from B and updating its LIB.

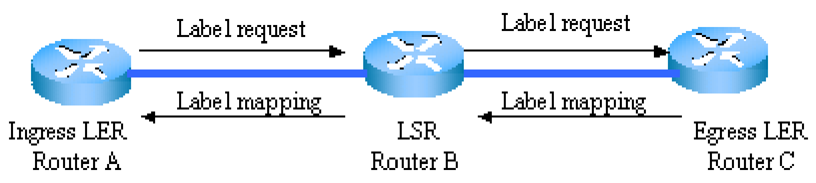


Figure 2.2 LSP Setup Signalling

Figure 2.3 demonstrates an example of labelled packet forwarding. The ingress IP packets will have label 3 attached to them and be sent out to via interface 1 once they enter the MPLS domain through the ingress LER A. At LSR B, all the packets with label 3 coming in through interface 2 will be sent out to interface 3 after the label 3 is swapped with label 17. Finally, at LER C the egress of this LSP, if the packets are with label 17 and coming in through interface 3, the MPLS label will be popped and the packets will be routed as pure IP packets to their final destination.

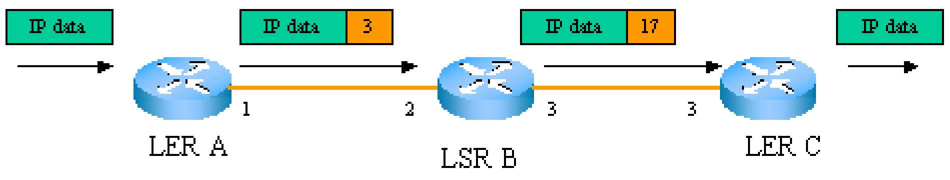


Figure 2.3 MPLS forwarding example

In MPLS, high-speed switching of data is possible because the fixed-length labels are inserted at the very beginning of the packet or cell and can be used by hardware to switch packets quickly between links instead of a lookup in the IP table, which avoids the longest prefix match normally associated with IP packet forwarding.

Currently MPLS is designed to address the network problems such as network-speed, scalability, QoS, and traffic engineering. MPLS has also become a solution to meet the bandwidth-management and service requirements for next-generation IP-based backbone networks. In its generalised form (G-MPLS), it is also used as a connection management protocol for optical channels and so forth.

2.2 Border Gateway Protocol

The Border Gateway Protocol (BGP) [20] is an inter-autonomous system routing protocol. An autonomous system is a network or a group of networks under a common administration and with common routing policies. BGP is used to exchange routing information for the Internet and is the protocol used between Internet Service Providers (ISP). Customer networks, such as universities and corporations, usually employ an Interior Gateway Protocol (IGP) such as RIP [21] or OSPF [19] for the exchange of routing information within their networks. Customers connect to ISPs, and ISPs use BGP to exchange customer and ISP routes. When BGP is used between autonomous systems (AS), the protocol is referred to as external BGP (eBGP). If a service provider is using BGP to exchange routes within an AS, then the protocol is referred to as interior BGP (iBGP) [22]. In Figure 2.4, routes are advertised by eBGP between AS1, AS2 and AS3, while the iBGP is employed inside AS to distribute these routes internally.

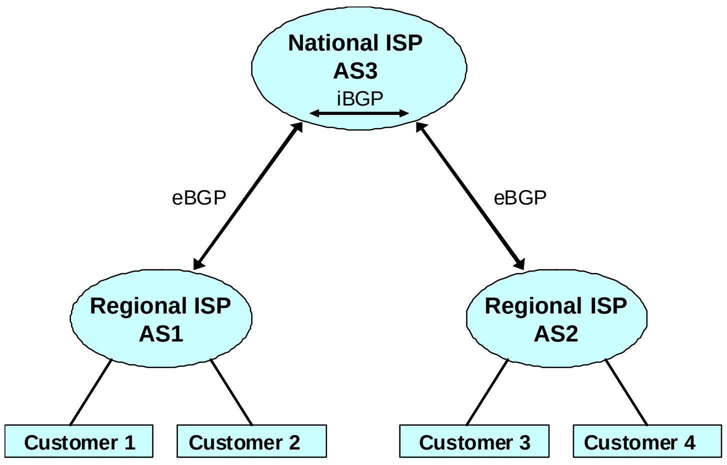


Figure 2.4 Example of eBGP and iBGP [22]

BGP is currently a very robust and scalable routing protocol, as evidenced by the fact that BGP is the routing protocol employed on the Internet [22]. Border Gateway Protocol Version 4 (BGP-4) (RFC 1771) [20] is the current inter-AS routing protocol used for the global Internet. BGP is an impure distance-vector algorithm; several path attributes carried in BGP messages affect the routing policy. Other BGP-related documents are RFC 1772 (BGP Application) [23], RFC 1773 [24] (BGP Experience), RFC 1774 [25] (BGP Protocol Analysis), and RFC 1657 [26] (BGP MIB).

BGP's basic information unit is the BGP path, which is a route[2] to certain IP address prefixes. BGP paths are tagged with various path attributes, the most important attributes are AS_PATH and NEXT_HOP. AS_PATH lists the ASes the advertisement message has bypassed. The NEXT_HOP tells the BGP speaker the interface IP address to the next AS. Every time a BGP path advertisement crosses an Autonomous System boundary, the NEXT_HOP attribute is changed to the IP address of the boundary router. Conversely, as a BGP path advertisement is passed among BGP speakers in the same AS, the NEXT_HOP attribute is left untouched [27]. Figure 2.5 describes how the route is propagated in BGP. In this example, Router C advertises network 172.16.1.0/24 of AS 200 with a next hop of 10.1.1.1. Router A received this route through the interface 10.1.1.2. When router A propagates this route to routers (i.e. Router B) within its own AS, AS 100, the eBGP next-hop information is preserved unchanged. In this example, B received the route to 172.16.1.0/24 from A through interface 10.1.2.2, and the next hop is still 10.1.1.1.

[2] Not a full route-just a list of ASes, providing reachability information

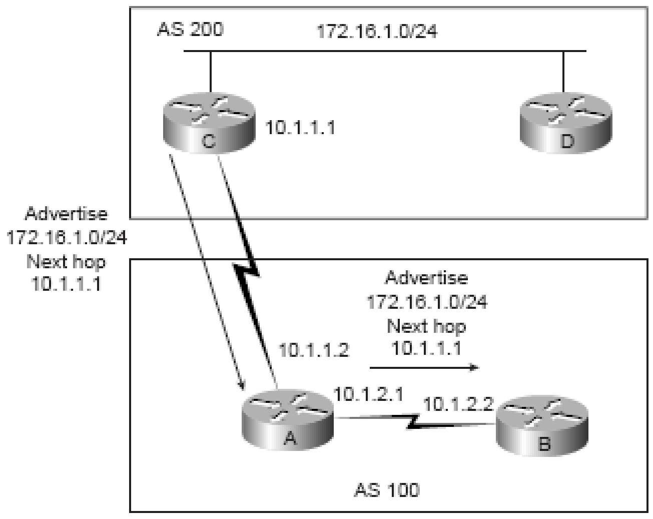


Figure 2.5 How BGP Propagates a Route [22]

2.3 Distribution of MPLS Labels with BGP

Since the BGP can distribute particular routes, it may also be used to distribute MPLS labels. RFC 3107 [28] describes how the MPLS label information is distributed in the same update message which is used to distribute the route itself. Label distribution can be piggybacked in the BGP Update message by using the BGP-4 Multiprotocol Extensions attribute [29]. The label is encoded into the NLRI (Network Layer Reachablity Information) field of the attribute, and the SAFI (Subsequent Address Family Identifier) field is used to indicate that the NLRI contains a label. A BGP speaker may not use BGP to send labels to a particular BGP peer unless that peer indicates, through BGP Capability Advertisement, that it can process Update messages with the specified SAFI.

If two immediately adjacent Label Switch Routers (LSRs) are also BGP speakers, the label distribution can be done without any other label distribution mechanism. BGP speakers are fullmeshed in a same AS (without BGP Route Reflection [30]), which means that these BGP routers are adjacent to each other (there is no 3rd BGP speaker between any pair) [30]. However, most of BGP speakers in different domains are not adjacent. [28] describes how MPLS operates when two BGP peers are not directly adjacent. Considering the LSR topology in Figure 2.6, suppose that LSR D distributes a Label (Ld) to A, B. C must not see the Ld at the top of the label stack during the packet forwarding process from A to D. To meet this requirement, A cannot simply push Ld to the label stack and send the resulting packet to B (B can see the Ld as a result). Before A sends the packet labelled Ld, it must push another label (Lb) distributed by B onto the top of the label stack. B must replace the Lb with Lc distributed by LSR C at the top of the label stack and send the packet with Lc to LSR C. Lc is popped at C and sent to D with the Ld at the top of the stack. There must be LSPs between A and

D, otherwise A cannot make use of Ld. This is true any time labels are distributed between none-adjacent LSRs, whether the label distribution is achieved by BGP or some other method [28].

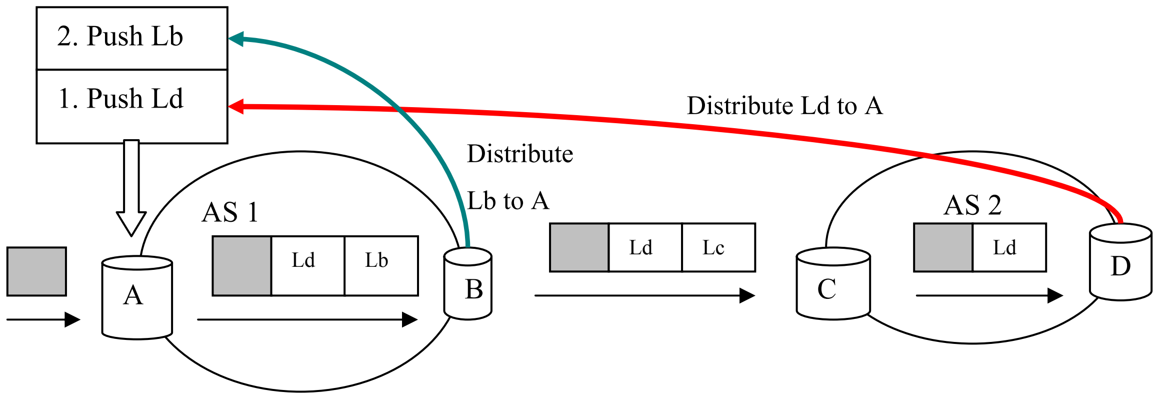


Figure 2.6 How MPLS Works When Two BGP Peers Are Not Directly Adjacent

2.4 Virtual Private Networks

A Virtual Private Network (VPN) is a network perceived to be private, constructed across a public or shared network infrastructure such as the public Internet. A more formal definition is given in [31] that "A VPN is a communications environment in which access is controlled to permit peer connections only within a defined community of interest, and is constructed though some form of partitioning of a common underlying communications medium, where this underlying communications medium provides services to the network on a non-exclusive basis".

Previously, if an organisation wished to set up a private WAN they needed to employ dedicated leased lines between their different sites. Service providers would provide such leased lines by layer 2 technologies such as ATM or FR (frame relay). However, leased lines are both expensive and restrictive; their bandwidth is usually fixed making it difficult to meet the short-term peaks in demand and wasting bandwidth at off-peak times. VPN are more cost-effective than private WAN because customers in different VPNs can share common resources. Service providers can provide more flexible service by sharing resources (such as bandwidth and CPU processors). At the same time, customers do not have to manage their geographically distributed networks themselves, this is done by the service provider. Figure 2.7 [31] shows an example of VPN. The red lines depict the VPN A across the service provider's backbone network. VPN B is completely unaware of the A's existence. A and B coexist on the same backbone infrastructure harmoniously, not being aware of the existence of each other.

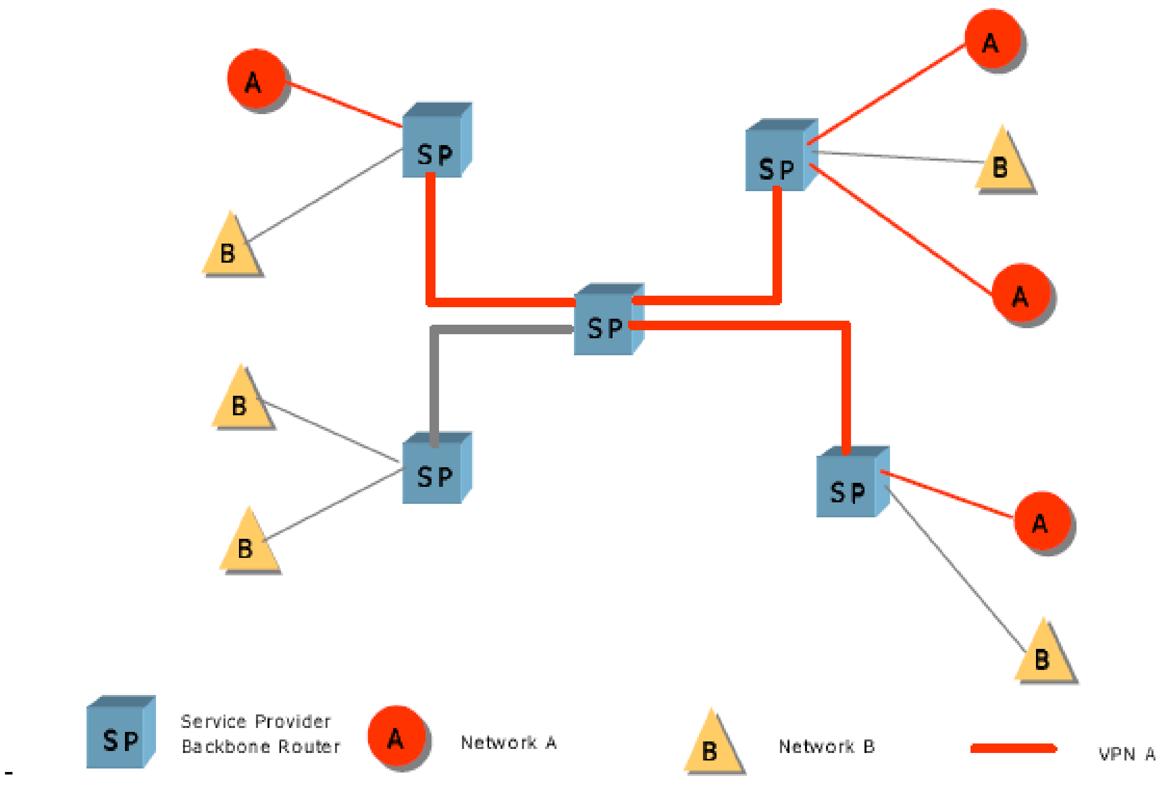


Figure 2.7 VPN Example [31]

Currently most of these VPN infrastructures are built on Frame-Relay or ATM networks connecting customer sites via Virtual Circuits (VCs.) In the overlay VPN model [31], the enterprise IP network is overlaid on top of the Service Provider backbone (Figure 2.7), the enterprise network is the higher layer network (layer 3) while the backbone network is the lower layer (layer 2). Both networks exist relatively independent of each other. The enterprise establishes router-to-router communication using an IGP whose signalling messages are merely regarded as data by the service providers [31].

For an enterprise network to be routed optimally in this model, it is necessary that the network (VPN) is fully meshed (Figure 2.8), which means that each site must have a link to each other site. This increases the number of VCs to a total of n*(n-1)/2 (n = number of sites). That increase in the number of VCs leads to significant increase in the complexity of the network and the routing information. Traffic engineering has also made things more difficult in this model because knowledge of site-to-site traffic is necessary to properly provision the VCs. All of these issues mean that a fully-meshed model does not scale well for large networks.

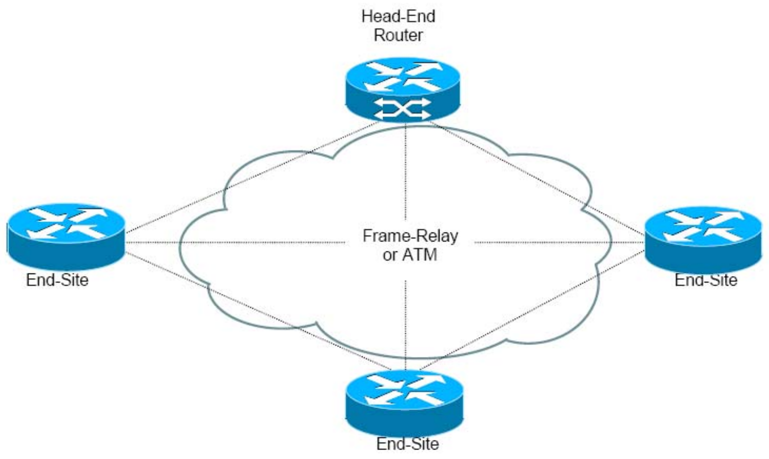


Figure 2.8 Fully Mesh VPN Example [32]

A peer model can be employed to tackle the scalability problem. In the peer model both the service provider and the customer use the same network protocol. The Provider Edge (PE) router directly exchanges routing information with the Customer Edge (CE) router. This simplifies the routing from the customer's perspective, as they no longer need to peer with every other end-site but only with one PE-router. Routing is optimal among customer's sites, as the provider routers know the customer's network topology. At the same time the addition of a new site is simpler due to the service provider not having to provide a whole new set of VCs [32].

The shared router approach and the dedicated router approach are the two implementation options for the peer model prior to MPLS based VPNs. The shared router approach is where several VPN customers share the same PE-router. This approach has to provide access control to ensure there is no crossover between different customers' traffic. The dedicated router utilises a separate PE router for each VPN customer, causing scalability concerns for the provider. Neither approach allows for the use of private IP addresses (RFC 1918 [33]), as each customer would have to have a unique public IP address [32].

The drawback of the peer models described above is that they cannot provide traffic isolation. The VPN users have to use unique addressing because all routes are placed in the global routing table [32].

2.5 BGP/MPLS VPNs

VPNs are usually categorised according to the OSI layer at which they provide the partitioning element in the public or shared infrastructure in which they operate. Although VPNs can be implemented at the higher layers in this protocol stack from the perspective of service providers and commercial customers, VPN technology is usually implemented at Layer 2 or Layer 3. A recent opportunity in this area is the exploitation of MPLS as a generic tunnelling technology in the implementation of such VPNs.

These virtual private networks are contingent on MPLS LSPs, so they provide the user with all benefits associated with this technology. Hence the performance advantages, along with traffic classification and segmentation inherent to MPLS, provide a solid foundation on which virtual private networks can be constructed. An additional benefit provided by these VPNs is the preservation of the customer's existing IP address scheme. Even if a service provider's customers have overlapping private IP address spaces (i.e. 10.x.x.x networks), these addresses will be effectively segmented so that data and control traffic will remain exclusively within the correct VPNs. All of these features are implemented by using the Layer 3 capabilities of the MPLS-enabled network. RFC2547 [5] defines the most mature MPLS VPN solution, commonly referred to as BGP/MPLS VPNs [34][35][36]. They allow a service provider to use its Internet Protocol (IP) backbone to provide an IP VPN service to customers. The service provider uses BGP to exchange VPN routes. An MPLS label is assigned to each VPN route, the process of label distribution is carried out using BGP (by piggybacking the label information on the BGP messages). While distributing reachability information, BGP also distributes MPLS labels for that path.

In the BGP/MPLS VPN model, the VPNs only exist at the edge of the service provider's network. The SP backbone routers (P routers) do not participate in the actual VPNs, they just forward labelled packets over various LSPs. The customer's routers also do not participate in the VPNs, they simply continue to route IP packets according to the customer's established addressing and routing schemes [37]. Figure 2.9 describes the architecture of a BGP/MPLS VPN.

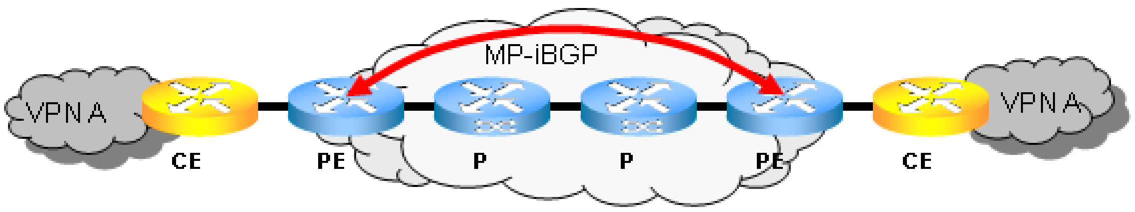


Figure 2.9 BGP/MPLS/VPN Architecture

A PE router is directly connected to the Customer Edge (CE) routers. Typically, a PE device will be connected to multiple CE devices supporting different customers. The PE router will communicate with the CE router via the routing protocol selected by the customer (RIP, OSPF, BGP, static routes, and so on). Thus the PE router will participate in the customers' Layer 3 networks; it will learn all the customers' routes and deliver packets according to that route information. This means that if a PE router is connected to a dozen of different customers, it will need to participate in a dozen of different networks, each of which will have unique addressing plans and routing protocols. The PE router also must communicate with other PE routers to exchange reachability information. If a customer in Los Angeles has a branch office in Dallas, the PE router in Los Angeles, it will need to communicate with the router in Dallas to establish the appropriate VPN tunnel. To accomplish this task, PE routers run Multi-Protocol BGP [38] on their wide area links. The PE routers will also need to initiate and terminate LSPs for MPLS traffic to be routed across the backbone (via the P routers). This means PE routers also implement the MPLS protocol as edge LSRs (ELSRs).

2.6 MPLS VPN Security

The security of the BGP/MPLS IP VPN architecture is the most important concern to service providers and VPN customers. The BGP/MPLS IP VPN architecture needs to address the following key security requirements.

1. It is necessary to have addressing and routing separation.
2. The internal structure of the backbone network must be hidden from the outside.
3. The network must have resistance to attacks, both Denial-of-Service (DoS) and intrusion attacks.

MPLS VPNs perform traffic separation at Layer 3 through the use of separate IP VPN forwarding tables. In RFC 2547, each customer's VPN is assigned a unique VRF (VPN Routing & Forwarding table). Packet forwarding within the service provider backbone is based on labels, label-switched paths (LSPs) begin and terminate at the provider-edge routers, while normal routing, in contrast, is performed by customer-edge routers. Traffic destined for each VRF will carry its own inner label value. The PE router determines which forwarding table to use when handling a packet. Because each incoming interface on a PE router is associated with a particular VPN, a packet can enter a VPN only through an interface that is associated with that VPN. Therefore, data traffic of one VPN will be separated from another.

In an MPLS based VPN, the provider's core infrastructure is invisible to the outside. Extensive packet filtering prevents exposure of any information about the VPN customer internal network or the MPLS core to the outside, making attacks much more difficult. Meanwhile, because only the provider-edge routers contain VPN-specific information, it is not necessary to reveal any internal network topology information to the outside or VPN customers. The service provider only needs to reveal the address of the provider-edge router, which is required by dynamic routing protocols between the provider edge and customer edge. Static routing can be configured between the provider edge and customer edge, which keeps the MPLS core completely hidden. Meanwhile, it is required that customer VPNs advertise their routes to the MPLS core so that they can be reached across the MPLS network. Because the information advertised to the core is about network routes, not specific hosts, the network security is not compromised. In this point, an MPLS VPN is equivalent to the existing layer 2 VPN models, such as Frame Relay or ATM networks, in which routing information about the VPNs can also be seen on the core network.

As it is not possible to directly intrude into MPLS VPNs, attackers might attempt to attack the MPLS core and use it to attack the VPNs [39]. There are two basic ways that the MPLS core can be attacked: by attacking provider-edge routers directly, or by attacking the signalling mechanisms of MPLS. Both types of attacks can be stopped by proper router configuration. To attack an element of an MPLS network, the attacker must know its address, but as stated above, all the addresses of the MPLS network equipment are hidden from the outside world. Meanwhile, even if an attacker can guess the core router IP addresses, the attacking packets cannot reach the targets. With the address separation mechanisms in MPLS, each incoming packet will be treated as belonging to the address space of the VPN customer. Therefore, it is still impossible for a packets coming from the outside of the core network to reach an internal core router even though the attacker knows the IP addresses [39].

On IP networks, IP source address spoofing has been extensively used by hackers and is a major security concern. In an MPLS environment, although it is possible for a VPN customer to do IP source address spoofing, it is impossible to use this mechanism to attack other VPNs or the core because of the strict separation between VPNs and the core network. IP spoofing only remains within the VPN where it originated. Meanwhile label spoofing is also impossible in an MPLS environment. As the interface between customer-edge router and its peering provider-edge router is a pure IP interface, any labelled packet from a customer side will be dropped at the PE. Attackers can also try Denial of Service (DoS) attack to make the resources become unavailable to authorised users. However as described above, in the MPLS VPN, the attack packets can never reach the core networks outside of the configured LSPs; the DoS attack is limited within the VPN where the attack was launched.

Based on the above observations, we can conclude that an MPLS VPN is as secure as a Layer2 VPN such as Frame Relay or ATM when the security requirements are considered, since MPLS VPNs isolate the service provider backbone from the customer [40].

In some cases, a customer does want to prevent its information from leaking to others even the service provider. MPLS cannot meet this security requirement by itself. Additional measures can be implemented to supplement the basic security features associated with RFC 2547 VPNs. Encryption is the most desirable method for ensuring data security. IPSec is a set of protocols developed by the IETF to support secure packet exchanges at the IP layer. To meet the customer's security requirements, IPSec also cooperates with MPLS to provide a secure VPN. In an IPSec/MPLS VPN, the data will be encrypted using IPSec first, then injected to the MPLS backbone [7]. Figure 2.10 is an example of an IPSec/MPLS VPN scenario.

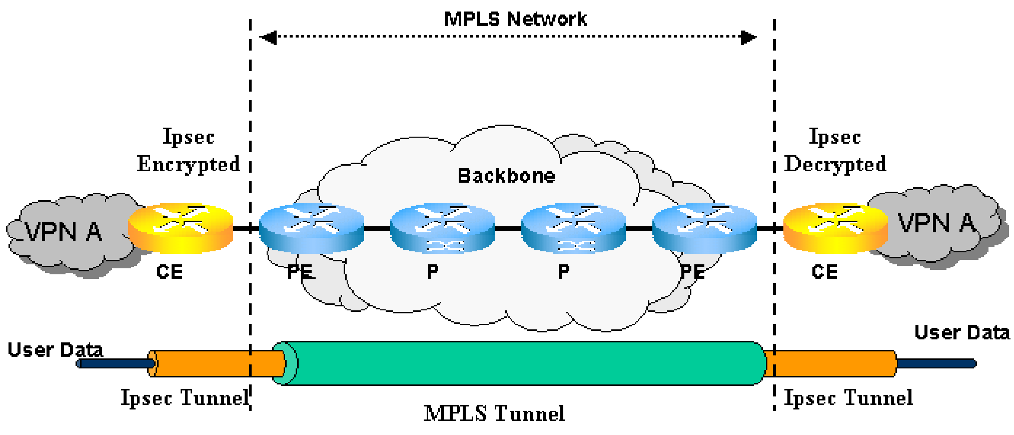


Figure 2.10 An Example of IPSec over an MPLS VPN

2.7 Inter-domain Connectivity

2.7.1 Inter-domain Issues and Shortcomings of BGP

Today, the Internet is partitioned into different Autonomous Systems (AS). An AS is a connected collection of IP routing prefixes under the control of one or more network operators that presents a common, clearly defined routing policy to the Internet [39]. The data exchange between ASes is carried out by one or more providers. Providers usually use independent policies and can vary in the geographical scope of their operations. For this reason, an inter-domain routing protocol is required to operate the packets exchange across AS borders.

As described above, BGP Version 4 is the current inter-domain routing protocol in Internet. A few papers have studied the weakness of BGP for providing inter-domain QoS guarantee and security [41][42][43][44][45][46][47][48][49]. The key challenges for BGP are described in the following paragraphs.

A. Convergence and Route Stability

BGP is an incremental protocol that sends updates only if something with regard to topology and/or reachability changes. Typically, if there is a change, it is followed by a number of updates sent between peers. Once all peers have settled on their best routes, the BGP advertisement actions will stop until the next change happens. A change causes BGP updates to be sent, and when all peers have settled on their new best route, the convergence process is finished. The convergence time is the time from an instability event to the stable best route on the observed router. Some possible reasons leading to BGP updates are listed as follows:

- Failure or repair of a physical link
- Reboot of a router (e. g. for software updates)
- Policy changes in originating or transiting ASes
- Addition/deletion of network prefixes.

When the instability event affects connections among ASes, it will cause an AS to withdraw a previously announced route or announce an alternative one. This new reachability information propagates in some way through the network of border routers and eventually, there will be no new updates to be sent. From the point of view of one router, this router may receive a number of updates from every peer, compute its own best route and announce it to its peers.

A typical distance vector protocol usually employs shortest-path routing, which means that if a choice has to be made between many routes to the same destination, the shortest one will be selected. However, BGP is not a pure distance vector protocol. BGP advertisements can include numerous attributes. It allows each AS to independently formulate its routing policies and allows these policies to override distance metrics in favour of policy concerns, which might result in an AS choosing a non-minimal path because of its local policy. Because the routing policies are implemented locally with little or no global knowledge, it is possible that

BGP routing messages are exchanged indefinitely, leading to a failure to converge on stable routes; what is called route oscillation or protocol divergence.

The commercial Internet grows both in size and topology complexity and route oscillations become a more and more serious problem [41][42][43][44][45][46]. “BGP is known to suffer from significant route instabilities, route oscillations and long convergence times. Nearly 25% of BGP prefixes continuously flap and a large fraction of these have convergence times on the order of hours. The remaining 75% of relatively stable prefixes typically take between 2 − 5 minutes to converge.” [47].

B. Security

When BGP appeared the Internet was not “dangerous” like today, and Internet was also only used for military and research but not business purposes. As a result, BGP was designed with little consideration for protection of the information it carries. There are no mechanisms inside the BGP to protect it from attacks that modify, delete, forge, or replay data, any of which may disrupt the overall network routing behaviour. At the same time, faulty, mis-configured, or deliberately malicious behaviours can also disrupt overall Internet performance by modifying, forging, or replaying BGP packets or breaking the distributed communication of information between BGP peers. The sources of bogus information can be either outsiders or true BGP peers. RFC 4272[48] describes the damage that these attacks can do.

C. Quality of Service (QoS)

QoS routing is essential for providing the end-to-end quality of service guarantees. Routing protocols on Internet are divided into two levels: intra-domain routing and inter-domain routing. Routing protocols need to be QoS-aware at both these levels to provide end to end QoS guarantees. There are already many solutions for intra-domain QoS routing protocol, as described in [49]. However, few solutions have succeeded in providing inter-domain QoS guarantees based on the BGP protocol, which mainly results from the lack of understanding QoS information between ASes [50]. BGP is currently the only popular inter-domain routing protocol. Although the BGP extension attribute field provides the space to include QoS information, it is still a challenge to make all these domains support a common QoS policy considering the complexity and diversity of the Internet. Indeed, two major challenges, scalability and heterogeneity, make the QoS extension to BGP difficult.

2.7.2 Related Research on BGP

BGP is the crucial inter-domain routing protocol in the current Internet, much research with regard of the above problems has been done recently. A summary of this related research is provided in the following paragraphs.

In [41], the authors discussed the BGP convergence problem in detail. In [42] and [43] the authors described the relationship between the topological structure of the Internet and the time it takes BGP to converge following the detachment of a subnet. In [42] the author made the conclusion that "customer sensitive to fail-over latency should multi-home to large providers, and that smaller providers should limit their number of transit and backup transit interconnections". In [43] the authors claimed that they can improve BGP convergence through the addition of synchronisation, diffusing updates, and additional state information, but all of these changes to BGP come at the expense of a more complex protocol and increased router overhead. These solutions for the convergence problem in intra-domain are proposed to be introduced into BGP, but it requires some complex and expensive changes in the existed BGP.

One previously reported effort to improve BGP convergence time is the sender side loop detection approach proposed in [44]. The simulation results show that in a 7-node fully connected topology, this approach improved the convergence time from 120 seconds to 30 seconds. However, Griffin et al [45] observed that in general, this approach reduces the convergence time only by a limited amount. They also observed that for each specific network topology they simulated, there is an optimal value for "minimal route advertise time" that minimises the convergence time. However, this optimal value varies from network to network. In [46] the authors proposed a new solution to reduce the convergence time, which uses the information provided in ASpath to define route consistency assertion and use these assertion to identify infeasible routes. It is achieved by defining and using new community attributes. The community attribute is a 32-bit value, normally associated with route advertisements and used to convey routing policy information. In [51] the authors proposed a solution by the modification of BGP (inject extra withdrawal messages) to flush the "ghost information" from the network.
Meanwhile, many papers pointed out that BGP is vulnerable and a great deal of work has focused on this, such as [52][53]. To prevent malicious amending and fake BGP information, hashing and/or signature-certificates are currently employed to secure the inter-domain routing. In [54] the authors proposed a secure border gateway protocol (s-BGP) for an authorisation and authentication system that addresses most of the security problems associated with BGP. In this solution, a Public Key Infrastructure (PKI) architecture is employed for authorisation and authentication. A BGP speaker will send signed update message to protest BGP router accepting malicious spoofing updates. S-BGP uses two PKIs (One public key binds with the certificate for the organization and a set of AS numbers; the other one binds to an AS number and to a BGP router ID), based on X.509 (v3) [55] certificates, to enable BGP speakers to validate the identities and authorisation of BGP speakers and of owners of ASes and of portions of the IP address space. The PKI is parallel to the existing IP address and AS number assignment delegation system. [54]

Some security solutions use systematic monitoring to protect BGP against malicious or abnormal routing behaviours. For instance, ASes can set up dedicated servers that contain a database of route information and routing policy configuration. These servers may exchange and serve authenticated data using a new protocol other than BGP [56]. The major advantage is that the authentication function is separated from normal routing activities. As for the cryptographic approaches, update messages need to carry additional cryptographic data for route authentication.

BGP extension attributes provide the space to put QoS information into BGP. However, because BGP routers can only infer limited QoS information from the advertisement they receive, the inter-domain routing decisions consider almost nothing about the real end-to-end QoS metrics, such as delay and bandwidth. At the same time, as the Internet is a best effort network, and given the characteristics of inter-domain routing described above, it is difficult to provide QoS guarantees by BGP itself.

Several related studies have been done on inter-domain QoS routing. Bonaventure [57] focuses on how to distribute QoS information flexibly by BGP in different network scenarios. Cristallo and Jacquenet [58] propose a new attribute for the BGP UPDATE message, QOS NLRI, to record QoS related information. Abarbanel and Venkatachalam [59] utilise BGP to propagate a Traffic Engineering Weight, which represents a summary of the traffic conditions in an AS. In [60] the authors extend MBGP (Multiprotocol Extension to BGP4) for inter-domain QoS Multicast.

2.7.3 Trusting Routing in Internet

In the current Internet routers blindly accept routing advertisements from other routers, which incurs a lot of security and service quality problems [61]. For this reason, many attempts have been made to provide trust model for distributed systems such as on-line trading (e-commerce), peer-to-peer systems and Extranets. Manchala [62] introduces trust metrics and models for the measurement of trust variables and fuzzy verification of transactions in e-commerce. Ba [63] demonstrates the application of a community responsibility system for building and enhancing trust in e-commerce. Au, *et. al*. [64] described a new paradigm for establishing trust across multiple organisations in Extranets. The trust token is used to propagate trust management information. Mui, *et. al*. [65] proposed a computational trust model based on reputation system, and this model is implemented in a real e-commerce system to compute trust and reputation scores of online users. Siyal and Barkat [66] showed how trust service can be provided through a network of trust service providers (TSP) and design a trust framework to support large scale B2C (Business to Customer) e-commerce. Li and Liu [67] created PeerTrust - a coherent adaptive trust model for quantifying and comparing the trustworthiness of peers based on a transaction-based feedback system. Kapadia, *et. al*. [68] created a model for discovering routes based on situational trust attributes of routers in ubiquitous computing, high performance, and peer-to-peer environments, they also define a "confidence" measure that captures the "quality of protection" of a route relevant to various dynamic trust relationships. According to these research results, the trust routing model can help network administrator to select trustful path when setting routing policies in BGP. The survivability of IP routing infrastructure is improved greatly when attacked, and the performance of IP routing infrastructure is improved greatly when part of the network is congested [61]. This is an important consideration when the inter-AS credit scheme is introduced in Chapter 7.

2.7.4 Routing in Mobile, Ad Hoc Network

Mobile, ad hoc network (MANET) is a kind of wireless network without centralised administration or a fixed network infrastructure, in which nodes perform routing discovery and routing maintenance in a self-organised way.

An Ad-hoc network has some similar characteristics to the Internet, one of which is the potentially selfish behaviour of each individual node. Each node may try to avoid working as a transit node for others to save power while trying to send its own traffic as much as possible. A credit-based mechanism is currently a popular solution to encourage cooperation among nodes against the node's selfish behaviour in ad-hoc networks [69][70]. Figure 2.11 is the architecture of the Sprite proposed in [69]. This architecture consists of the Credit Clearance Service (CCS) and a collection of mobile nodes. The nodes are equipped with network interfaces that allow them to send and receive messages through a *wireless overlay* network, When a node sends its own messages, the node (or the destination, see later) will lose *credits* (or virtual money) to the network because other nodes incur a cost to forward the messages. On the other hand, when a node forwards others' messages, it should gain credits and therefore be able to send its messages later. The research proves that the credit based architecture can encourage the cooperation among the mobile nodes in MANET.

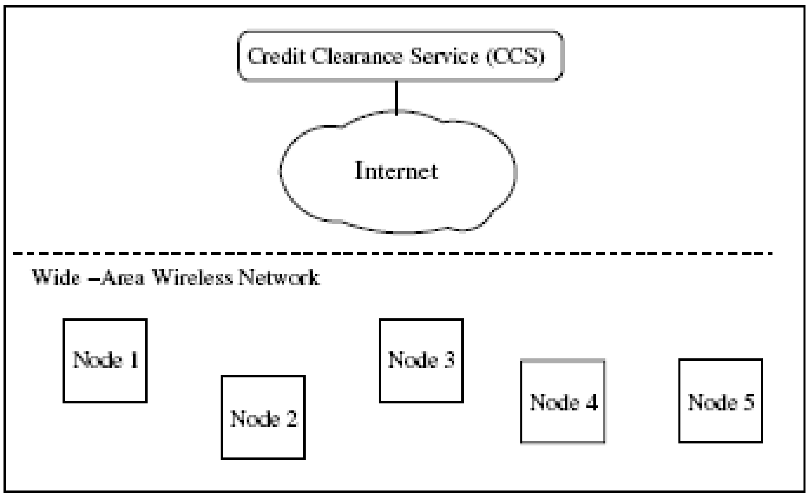


Figure 2.11 The Sprite Architecture [69]

2.8 Research on Dynamic VPNs

VPNs already carry advanced IP applications such as Enterprise Resource Planning (ERP), Customer Relationship Management (CRM), videoconferencing, and other mission-critical applications that require guaranteed performance. Typically the formation and maintenance of VPNs within an operator's domain is achieved using manual intervention to configure the various network resources. This is acceptable for long-term connectivity such as inter-connection between enterprise campuses. However, new forms of working are likely to have resource usage demands that do not fit well with this arrangement. In these cases it is desirable to establish community relationships that may last minutes, or hours through to weeks.

Indeed, seeing these limitations, a number of Dynamic VPN initiatives have recently begun to emerge. Examples include Closed User Groups (CUG) [71][72], the work of Kindred and Sterne[73], Jia[74][75]and the Defense Research & Development Canada (DRDC) demonstrator[76].

Closed User Groups (CUG) can be regarded as flexible VPNs for supporting the dynamic formation and self-management of distributed collaboration networks [72][73]. CUGs are coordinated groups of peers who reside in different organisational domains, managed by an administrator. The ability of users to initiate the groups themselves is a key feature that differentiates CUGs from traditional VPNs. Figure 2.12 shows the logic architecture of the CUG.

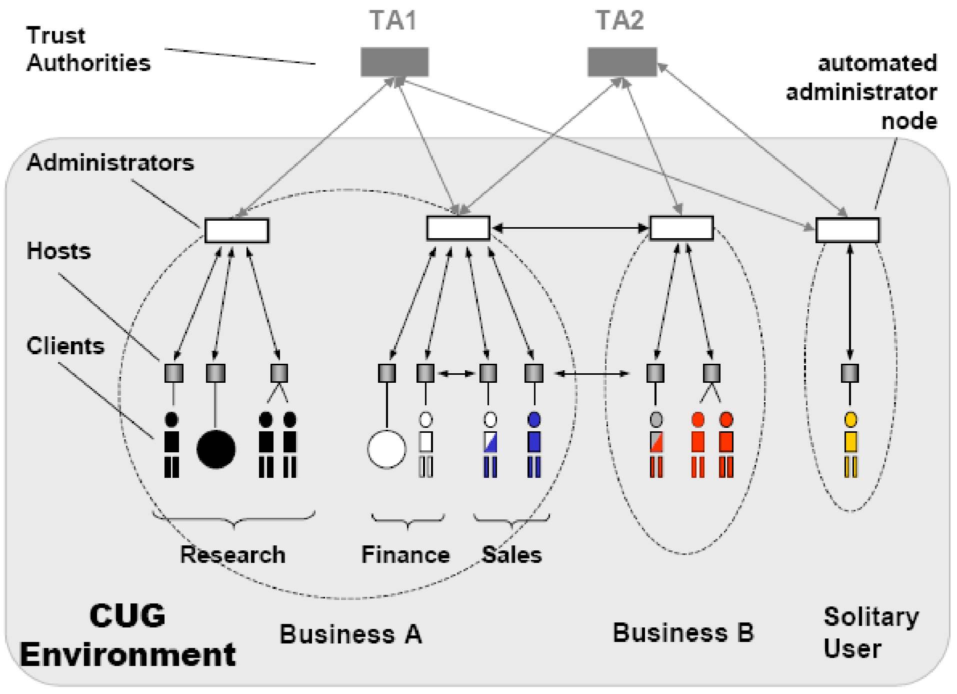


Figure 2.12 The CUG Architecture [77]

The circles representing different businesses do not have any topological implication, but only refer to the logical structure of the organisation. Members of different groups are distinguished by their colours. Departments in the organisation would naturally group their staff in different CUGs. However, clients can be members of several groups.

The firewall functionality is localised to each host, which (through a login process) controls personalised user access through the firewall at the given host based on the certificates in its possession for identified user(s). This enables different working environments to share the same terminal between clients. The firewall enforces organisational policy (set by the administrator at the client's initial setup) and regulates client's interactions within CUGs (on the basis of a group-specific certificate). All hosts then become part of a large distributed firewall, also providing all the features offered by a traditional firewall choke point.

From the viewpoint of group-work, the CUG model distinguishes two main classes of roles: *group members* and the *group manager*.

• **Group members** are peers (clients) who are informed of each other's identity and location and interact with each other on a peer-to-peer basis by using a form of group certificate embedded in the messages they interchange. Messages with certificate information that do not match the required group certificate are either deleted or ignored without further processing.

• **Group manager** (Administrator) is responsible for managing group membership and issuing or updating group certificates (in terms of group membership and policy). The group managers of one level in the organisational hierarchy may themselves be viewed as members of a CUG at a level above the level of the teams they manage.

This structure resembles line management in structured organisations. The managers maintain many simultaneous groups (distinguished and constrained by the group certificate issued by a manager) in terms of group memberships and security policies, and remain responsible from group creation until its termination.

In [78], the author described an interesting scheme to deploy overlay networks that achieve both policy update and Layer-3 topology reconfiguration in a scalable manner. The system architecture is shown in Figure 2.13.

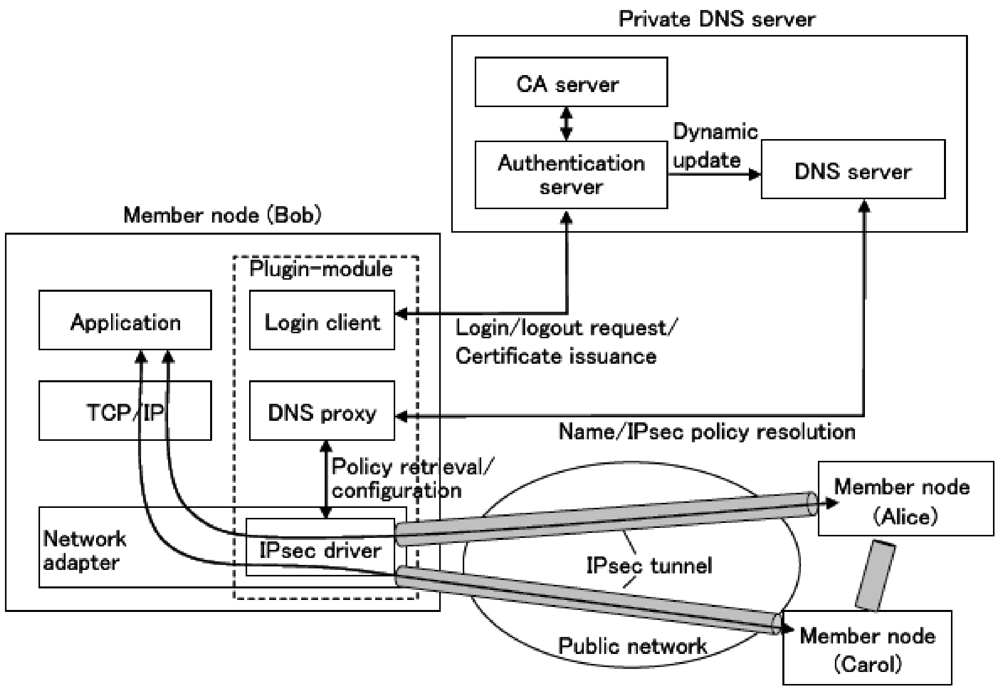


Figure 2.13 Dynamic CUG Based on Overlay Network [78]

It is composed of a private DNS server and a plug-in module and IPSec is employed to provide security communication and authentication. The plug-in module is installed on each member node terminal. An overlay network is provided by these two modules for each CUG. The private DNS server provides similar management functionalities. Initially, an administrator defines an overlay network for a CUG with its membership and miscellaneous information (e.g., IP address space and domain name space) through a Web GUI provided by the private DNS server, where multiple overlay networks can be defined at the same time. After this process, each member can request to participate in the closed network through the login client in the plug-in module. If the login request is accepted, the private DNS server assigns an inner IP address, a private domain name, and an X.509 certificate to the member node. The certificate is signed by the certificate authority (CA) server and is used for authentication both in establishing a control channel (TCP/SSL) with the private DNS server and in establishing IPSec tunnels with other member nodes. The DNS proxy in the plugin module performs IPSec policy resolution as well as normal DNS resolution for domain names associated with the CUG via the control channel established with the private DNS server. This scheme can be generally applied to a system to deploy overlay networks for a CUG with dynamic membership. [78]

In the 6net [79] project, Defence R&D Canada (DRDC) developed the dynamic virtual private network controller (DVC) prototype for the rapid deployment and self-configuration dynamic coalition virtual private networks (VPNs) [80]. DVC is the equipment carrying out the dynamic VPN operations. VPNs can be created, dismantled, and monitored concurrently. With single Management Console commands, the DVC operator can instruct the DVC to establish a VPN to all sites in a coalition or to dismantle the VPN to all coalition sites. A single VPN connection encrypts the traffic between two coalition sites. Traffic sent from one coalition site to another coalition site is never routed through a third coalition site. The DVC prototype uses secure authenticated out-of-band channels to establish, monitor, and dismantle the VPN connections. Each DVC maintains a DNS name, an IP address, and a security policy for every potential coalition partner.

Figure 2.14 illustrates an example of this DVC based dynamic VPN approach. The site security officer uses the DVC Policy Editor to create and install the security policy. The DVC operator uses the DVC Management Console to manage the VPNs. The DVC transmits security policies to remote peers, and receives and evaluates security policies from remote peers. When a decision has been made to create or dismantle a VPN, the DVC modifies the configuration of the IPSec, firewall, DNS, and routes sub-systems to enforce the local and remote security policies.

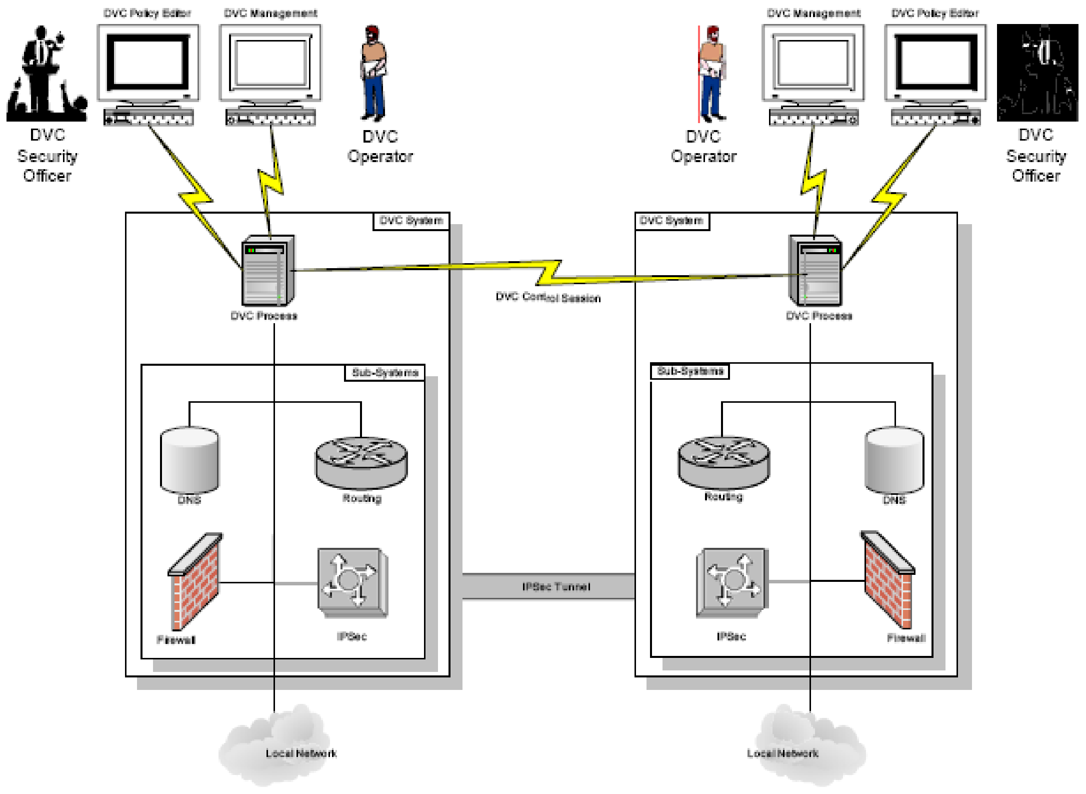


Figure 2.14 DVC Dynamic VPN Architecture [80]

The DVC project has successfully demonstrated a solution for the rapid deployment and self-configuration of VPN connections that provide a secure information exchange capability in a dynamic coalition environment. Figure 2.15 is the implementation of the Defence Research and Development Canada (DRDC) agency's Dynamic VPN Controller (DVC) system where DVCs exist at networks (NRNS, UCL, DRDC and UMU).

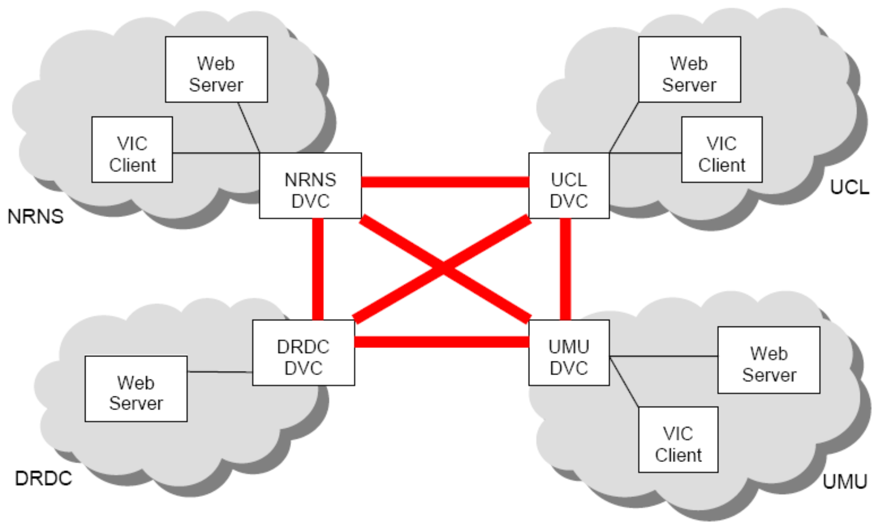


Figure 2.15 DRDC DVC Network Set-up [81]

Another Dynamic VPN solution and its implementation prototype are demonstrated in [73]. The system architecture is shown in Figure 2.16 which is redrawn in accordance with [73]. The Community Manager (CM) manages the membership of VPNs in its community. The community is the set of all the networks managed by a CM and the term "Community Manager" loosely refers to both the software components that support community management and the human operating the software. The CM is the central authority of community membership and distributes community membership information to each firewall of the sub-network. The firewall for each member sub-network automatically establishes individual IPSec-based VPN links with all other members based on the CM's membership information. By introducing the central manager-CM, the VPN might be operated automatically, which enables fast dynamic VPNs operations while avoiding human mistakes in configurations as well as reducing the maintenance costs.

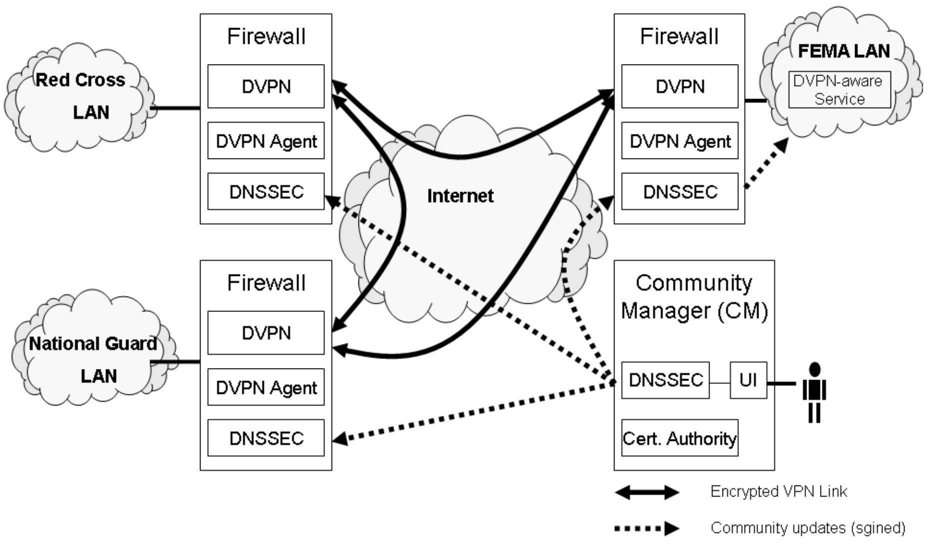


Figure 2.16 CM managed Dynamic VPN architecture

2.9 Resource scheduling

Unlike conventional networks that focus on communication among devices, some new applications such as grid-computing harness distributed resources in a network for solving problems or supporting group-working activities. Typically, a grid system coordinates resources that are not subject to centralised control and uses standard, open, general-purpose protocols and interfaces. It allows these constituent resources to be used in a coordinated fashion to deliver various qualities of service and meet complex user demands.

These resources may themselves be dynamic and may enter or leave the system at any point of time or fail. A scheduling strategy is required to generate schedules, which seek to minimise the total execution time of jobs and also adapt to the heterogeneity and the dynamism of the environment. The allocation of a resource to a job, involves three basic steps [82]:

1. Resource Discovery: Identifying the resources that are currently free (without being allocated to any jobs) in the system.
2. Resource Selection: Choosing one of the free resources based on some underlying algorithm to

schedule it to a job on the ready queue.

3. Job Execution: Allocating the chosen resource to a job and executing the job.

The goal of the resource allocation is to achieve an optimal solution such as the shortest completion time. There are few publications addressing the resource-scheduling problem for independent jobs [83][84][85][86]. Meanwhile, there might be dependences among the tasks of a customer application. For example in Figure 2.17, task 2 and tasks 3 are the upper dependent tasks of task 4. Task 4 cannot start until task 2 and task 3 complete. Here, the resource allocation can be converted into the workflow scheduling problem. Workflow management techniques aim to support business processes across organisations. Now it is used with many other complex parallel multitasking scenarios such as automatic control [87], grid computing and e-business [88][89]. There are many studies focused on this topic such as [87][90][91][92][93][94][95][96]. Normally, a Directed Acyclic Graph (DAG) [97] is employed to describe a workflow. Figure 2.17 shows a simple DAG example. The blue nodes are the tasks and the directed arrows represent the workflow directions and dependency relationships. The number above every arrow shows the distance (the amount of data transferred) from the upstream task to the downstream task.

CPA (Critical Path Analysis) is a method for finding the optimal solution [92], which searches for the longest path from the start point to the finish point among the workflow modelling net (such as a DAG). For example, in Figure 2.17, the longest path 1→3→4→5 is the critical path. By minimising the length of the critical path, the resource allocation with the shortest completion time can be found.

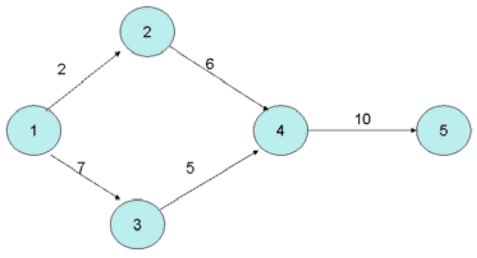


Figure 2.17 A Directed Acyclic Graph Example

As introduced above, a Directed Acyclic Graph (DAG) is a common tool for representing the dependency of tasks. A number of researches have focused on how to allocate network resources to the tasks represented as a DAG. For example, a dynamic algorithm named the Hybrid Remapper is proposed in [96]. The algorithm architecture is shown in Figure 2.18. Initially, the DAG is partitioned into several blocks numbered consecutively from 0 to *n-1*, where n is the total number of the blocks shown in Figure 2.18 (a). The subtasks within a block are independent, and there are no data dependencies among the subtasks in a block. Once the subtasks in the DAG are partitioned, (assuming there are totally *B* blocks) each subtask is assigned a rank by examining the subtasks from block (*B* -1) to block 0. The rank of each subtask is set to its expected computation time on the machine to which it was assigned by the initial static matching. The initial static matching indicates the mapping relationship between the tasks and the resources before the Hybrid Remapper algorithm commences and provides the algorithm starting point.

Figure 2.18 (b) illustrates the rank assignment process for the subtask S*i.* The rank of a subtask can be interpreted as the length of the critical path from the point the given subtask is located on the DAG to the exit node, i.e., the time until the end of the execution of all its descendents. By executing the subtasks with higher ranks as quickly as possible, the overall expected completion time for the application can be minimised.

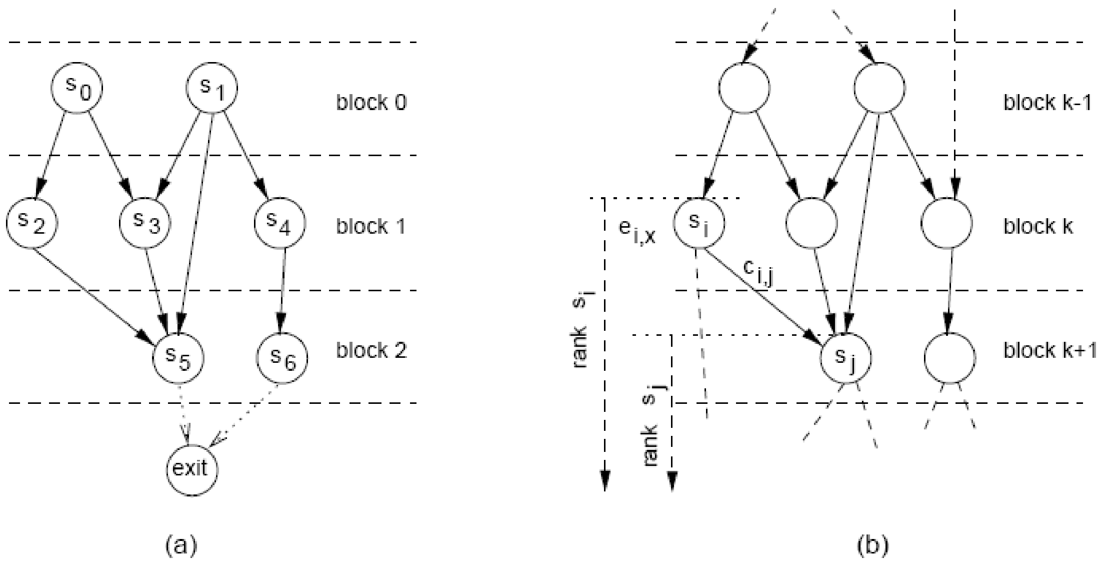


Figure 2.18 The Hybrid Remapper Algorithm Architecture [96]

There might be various resources needing to be assigned to the customer applications, such as data repositories, in addition to computing resources. For example, the applications can be computationally demanding and communication intensive as well. A unified scheduling strategy described in [97] studied how to allocate network resources to the DAG to meet various demands in a unified manner. In the paper each application job is represented with a Directed Acyclic Graph (DAG). A job consists of several subtasks and the resource requirements are specified at the subtask level. Several scheduling algorithms consider computing resources and data repositories that have advanced reservations. The simulation results show that it is advantageous to schedule the system resources in a unified manner rather than scheduling each type of resource separately. The algorithms have at least 30% improvement over the separated approach with respect to completion time.

Multithreading is defined as a concept that aims to optimise execution time while keeping the resource utilisation near optimal in [98]. In [93] the author aims to address this issue using a two-stage approach, resource estimation, which determines the optimal number of resources required to speedup the execution of a given application, and scheduling, which orders the execution of the threads of a given application by mapping them to the resources while minimising execution time.

To estimate the optimal number of the resources, efficiency is defined as representing the nearness of the maximum execution time of the DAG to the best case which is the critical path time of the DAG.

Considering the example in Figure 2.19, if there is one resource, because one resource can only carry out one task at a time, the maximum execution time is 3+1+1. While the critical path time of this DAG is 3. As a result:

Efficiency = 3/ (3 +1+1) × 100% = 60% **(2.1)**

If there are two resources, task T14 and T13 can be carried out at one resource while T4 is carried out at another one, the maximum execution is max {3, 2} and Efficiency = 3/3 = 100%. Hence, two processing elements are needed for the DAG shown in Figure 2.19.

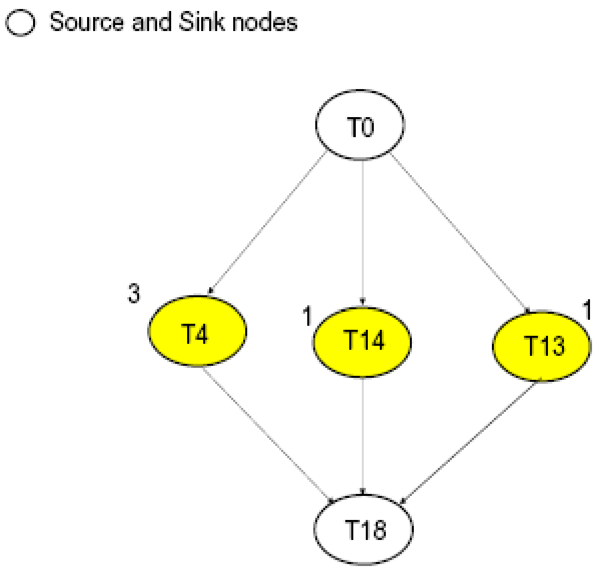


Figure 2.19 Example of Resource Estimation [93]

The customer applications might also have time constraints. For example, some tasks may need to be completed before a deadline. In [92] the author researched how to manage workflows with time constraints such as deadlines by finding the critical path, because the critical path is the longest execution path in a workflow which directly affects missing the workflow deadline if the completion of an activity within the critical path is delayed. A method named Innermost Control Structure First (ICSF) is also proposed to find out the critical path in the time constrained workflow.

2.10 Optimisation Algorithms for Resource Scheduling

Resource scheduling in applications such as grid computing is a very complex heterogeneous multi-resource and real time scheduling problem and is well-known to be NP-complete problem [17] [18]. Various meta-heuristics methods might be used to solve the optimisation in NP-complete situations.

Local Search (LS) uses the notion of neighbourhood (where a set of solutions are possible to generate). These algorithms start from a chosen initial solution and move from one solution to another by searching successive neighbourhoods so that the solution closest to the optimum is achieved. The LS strategy is used to find solutions for hard combinatorial optimisation problems [99]. One disadvantage of LS is that, there are chances of a LS algorithm getting stuck in a Local Maxima or Local Minima. Simulated Annealing (SA) [99] imitates the way in which a metal cools and freezes into a minimum energy crystalline structure (the annealing process). SA works by searching the set of all feasible solutions, and reducing the chance of getting stuck in a poor local optimum by allowing moves to inferior solutions under control of a randomized scheme. The disadvantage of SA is that determining the initial value of control parameter and the method of decreasing its value is complex and difficult. Tabu Search (TS) is a meta-heuristic strategy based on neighbourhood search whilst overcoming local optimality. Unlike SA, TS tries to model human memory processes. Memory is implemented by the implicit recording of previously solutions. This approach focuses on the creation of a Tabu list of moves that have been performed recently and are forbidden to be performed for a certain number of iterations, so that cycling can be avoided and promote the search in a diversified space. In [99] the authors compared the various heuristic algorithms (Opportunistic Load Balancing (OLB)**,** Minimum Execution Time(MET) , Minimum Completion Time (MCT), Min-min, Max-min, Duplex and Simulated Annealing (SA), GA). Their results show that GA always gives the best scheduling arrangement with the scenarios they considered.

2.10.1 Genetic Algorithm

A Genetic Algorithm (GA) [100] simulates evolutional phenomenon that the fittest of the natural world survives, and maps to a genetic space instead of search space. Then GA codes the possible solutions to a vector - chromosomes, and each element of the vector is called a gene. In each iteration, the fitness of each chromosome is calculated. The fitness value represents how the chromosomes fit the criteria. The chromosomes with bigger fitness tend to survive and the chromosomes with poor fitness are more likely to die. This process repeats for a number of iterations then the chromosome with the best fitness value is picked out as the best solution.

GA operates in three basic steps: Selection, Crossover and Mutation. The three steps are analysed in the follow paragraphs:

1> Selection

Select a new population of P members to the next generation according to the fitness value:

Let f_i be a measure of the fitness of member I, the fitness function is shown as:

$$P_i = f_i / \sum_{k=1}^{m} f_k \qquad \textbf{(2.2)}$$

where P_i is the probability of choosing i and m is the number of chromosomes.

Consider an example where there are 4 chromosomes A, B, C, D in the current population, The relationship between the fitness value and the probability of choosing the chromosome into the next generation are shown in Table 2.2:

Table 2.2 Example of The Relationship Between Fitness and Probability Chosen

Subject	Fitness	Probability of Selecting
A	1	1/7 = 0.143
B	1	1/7 = 0.143
C	2	2/7 = 0.286
D	3	3/7 = 0.428
Total	7	7/7=1

2> Crossover

Randomly select two chromosomes (chromosome parents), designate one point or more to exchange to generate two new chromosomes, shown as Figure 2.20.

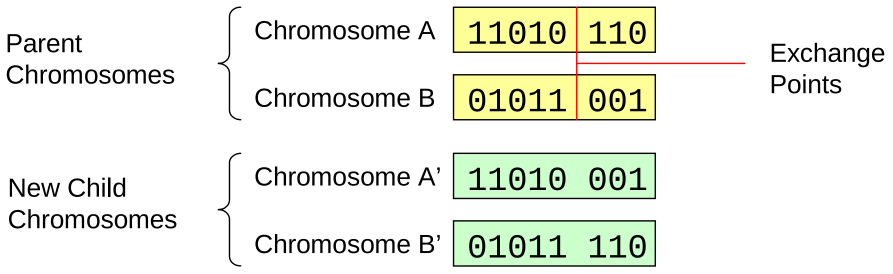


Figure 2.20 The Crossover Process

3> Mutation

Changes in the natural environment can lead to gene mutations. In binary coding of chromosome, 1 becomes 0, or 0 into 1. Chromosome mutations produce diversity and avoid premature in evolution sinking into a local maximum point.

Figure 2.21 represents the flow chart of the GA process.

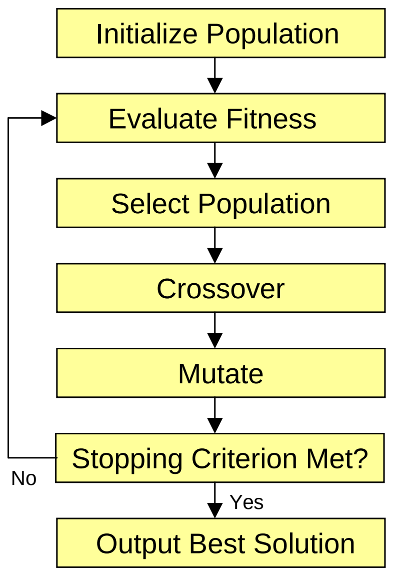


Figure 2.21 GA Flow Chart

GA will repeat its three basic processes until a termination condition has been reached. Common termination conditions are [100]:

- A solution is found that satisfies minimum criteria
- Fixed number of generations reached
- Allocated budget (computation time/money) reached
- The highest ranking solution's fitness is reaching or has reached a plateau such that successive iterations no longer produce better results
- Manual inspection
- Combinations of the above.

The chromosome with the best fitness is picked out from the population of the last generation. This chromosome maps the best solution.

Genetic Algorithms represent a complex phenomenon by natural evolutional mechanism, which can solve difficult problems quickly and reliably. GA possesses inherent parallelism and parallel computing capacity. In addition, it is scalable and easy to mix with other technologies. GA has two main characteristics as follows:

A. Group search.

Many of the traditional search methods are point-to-point searching (i.e. start from one point and go in one direction in each searching iteration), which usually sink into local peak of the extreme point when searching multi-peak distribution of space. On the contrary, genetic algorithms use a method that deals with a number of individuals in a group simultaneously, evaluating many solutions of the search space at the same time. This characteristic has a good overall searching performance.

B. Heuristic characteristics of stochastic search.

Genetic algorithms make use of probability changes to guide the search direction. It seems to be a blind search method; in fact it has a clear direction, and inherent parallel search mechanism. Genetic Algorithms search the population is parallel, rather than a single point. They do not require supplementary information, needing only an objective function of search direction and the corresponding fitness. GA uses probability transformation rules, instead of identified changing rules.

A genetic algorithm, as an optimisation method, inevitably has its limitations. The efficiency of genetic algorithms often is lower than other traditional optimisation methods. In some scenarios, if it is not configured properly, a Genetic algorithm may become to premature convergence, which can find the local optimal solution instead of global optimal solution [101].

2.10.2 Particle Swarm Optimisation

The PSO (Particle Swarm Optimisation) algorithm was proposed by Kennedy and Eberhart in 1995 [102][103][104]. It was inspired from the social behaviours of bird flocking and fish schooling; a bird flock is searching for targeted food whose position is not known by the birds, i.e. they do know how far is their present position from the food. As a result, the most effective way to find the food is to explore the area around the nearest birds. Based on such behaviour, PSO, to be an optimisation tool, provides a search procedure based on population of particles changing their states with time. In a PSO system, a large number of particles fly around synchronously. They can change their direction suddenly, benefiting from the experience of their own and that of the other members of the swarm during the search for an optimal target. Each particle adjusts its position according to the best position which is encountered by itself and its neighbours.

To simulate the bird flocking for food foraging, the particle vectors are iteratively modified during the PSO evolution. According to the fitness values (which indicate the merit of this particle vector such that the swarm evolution is navigated towards best particles unless better solutions are discovered) of these particle vectors, each particle remembers the best vector it experienced so far, referred to as pbest (personal best vector), and the best vector experienced by its neighbours referred to as gbest (global best vector). The gbest is determined across all the particles in the entire swarm. During each PSO iteration, particle i adjusts its velocity v_{ij} and position vector $particle_{ij}$ through each dimension j by referring to, with random multipliers, the personal best vector ($pbest_{ij}$) and the swarm's global best vector ($gbest_{ij}$) as follows:

$$\mathbf{v_{ij} = C0 \times v_{ij} + C1 \times rand1(gbest_{ij} - particle_{ij}) + C2 \times rand2(pbest_{ij} - particle_{ij})} \tag{2.3}$$

$$\mathbf{particle_{ij} = particle_{ij} + v_{ij}} \tag{2.4}$$

where C0, C1 and C2 are the cognitive coefficients and rand1 and rand2 are random real numbers drawn from U(0,1).

Figure 2.22 shows the movement of a particle (P). The particle's position at *(t+1)*th iteration is affected by the personal best vector, global best vector and itself vector of *t*th iteration.

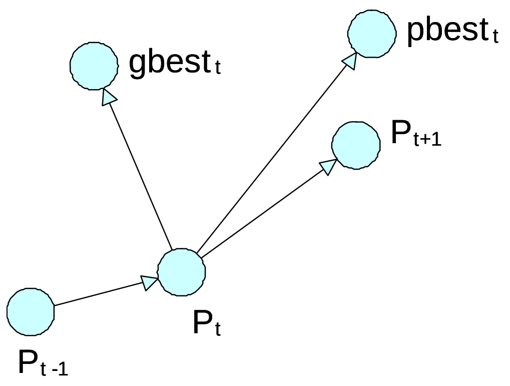


Figure 2.22 Particle Movement at Iteration t+1 with the influence of pbest and gbest

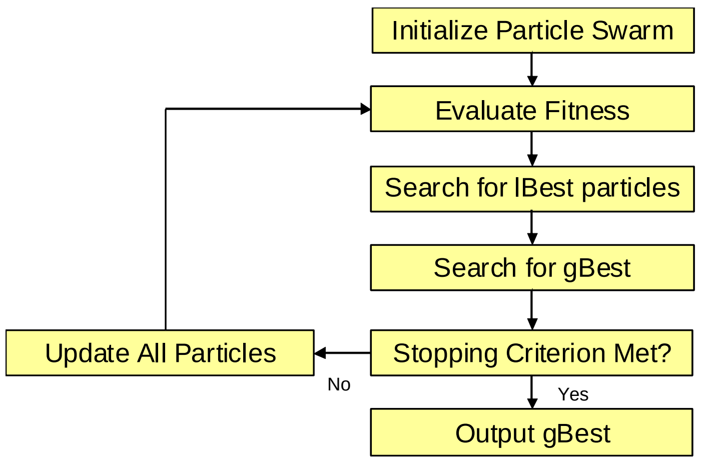


Figure 2.23 PSO Flow Chart

Figure 2.23 shows the PSO flow chart. The particles are first initialised. Then the fitness value of each particle is calculated based on the fitness function and particles are updated. For each particle, if the fitness value is better than the best fitness value achieved (pbest) in history, the pbest is set to the current vectors. Next, PSO chooses the particle with the best fitness value of all the particles as the gbest, for each particle, the velocity is calculated according to equation (2.3) and particle position is updated according to equation (2.4). PSO will repeat the above processes until the maximum iterations or some other stopping criteria is met.

Compared to traditional approaches to resource allocation [105][106][107][108], according to the published research results, PSO shows the following advantages: firstly, the traditional approach may fail to deliver exact solutions within a reasonable time for problems of large size. Secondly, compared to traditional approaches, the high-speed searching ability enables PSO to obtain a good solution in a much shorter time.

Thirdly, according to its versatility in handling all types of objective function and constraints, PSO can be easily combined with traditional methods, and then used to provide efficient solutions to specific problems.

2.10.3 Comparison between PSO and GA

PSO and GA have many things in common: both of them are population-based, evolutionary computation algorithms. They both use the fitness value to evaluate the system and execute a random search depending on it.

In genetic algorithms, the movement of the entire population is relatively homogeneous towards to optimal region, that is to say, the GA-based method does not actually reduce the searching space. On the other hand, in PSO, only gbest (or pbest) provides information to other particles, so this is a single-direction information flow. The whole searching process is directed towards the current optimal solution although a small amount of swarm exploration is configurable (i.e. based on rand1 and rand2). So in comparison with the genetic algorithm, in most cases, PSO-based algorithms may convergence to a good solution faster.

2.11 Summary

VPNs provide safe and confidential communications between sites in different locations. Users of a VPN can work as if they are within one private network. Meanwhile, a VPN is a low cost and flexible solution compared to the leased line and other private network technologies because different customer VPNs can share the same underlying backbone network resources and not be aware of each others' existence.

MPLS is a popular technology that provides reliable and QoS guaranteed communications over IP network. BGP4 is the current inter-domain routing protocol used in the Internet to advertise routes crossing domain borders. RFC 2547 defined the most mature BGP MPLS VPN solution; MPLS provides the VPN tunnel crossing backbone network and the label information is carried by BPG messages.

However BGP4 has reliability issues as well as concerns in terms of QoS and security because the Internet is such a complex and distributed system. Several studies have focused on the current BGP shortcomings and inter-domain routing problems. Many solutions have been proposed to provide security and reliable inter-domain communication. Indeed, several approaches appear to provide inter-domain, dynamic and secure VPN solution.

The resource scheduling problem is also investigated in this chapter. Scheduling jobs to available network resources can be converted into a workflow management problem. CPA is a common approach for workflow management because the overall completion time is related to the longest workflow path which is the Critical Path. Several resource scheduling algorithms are analysed. Because resource scheduling in a complex and dynamic network is a NP-hard problem, the optimisation algorithms for NP-hard problems are also analysed. GA and PSO are introduced as they are robust and feasible optimisation tools.

The chapter has reviewed the technologies and solutions relevant to this thesis. In the following chapter, the author will analyse the weakness of the existing solutions and describe the motivation for the proposed approach.

Chapter 3 Motivation and Requirements for Inter-Provider Dynamic VPNs

Today, many new applications challenge the existing network technologies. International cooperation and the global businesses require a more reliable, flexible and secure "global network". However the existing VPN solutions cannot satisfy these new requirements. For this reason, a new network solution is necessary.

3.1 New Applications

Grid Computing

Scientists are using computers to research large and complex problems. Evidence of this fact is the National Science Foundation's Office of Cyber infrastructure [109]. Although CPU performance has improved dramatically in the past years, it is still far from adequate for many complex computing intensive applications such as weather predictions and scientific calculation [109]. Meanwhile, there are numerous processors (i.e. end user computers) idle in the Internet. For this reason, grid computing [110][111] is employed to use these idle processors for the computing intensive jobs. Currently global grid computing is becoming a very attractive topic as it enables computing intensive jobs to be carried out using the global network resources. The users and resources of the grid computing infrastructure are distributed globally and communicate over the Internet.

Grid Computing is a form of networking [112]. However, unlike conventional networks which focus on communication among devices, Grid Computing harnesses unused processing cycles of all computers in a network for solving problems too intensive for any stand-alone machine. In [113] the authors defined a Grid Computing as a system that:

- Coordinates resources that are not subject to centralised control. Instead, a grid integrates and coordinates resources and users that live within different control domains; different administrative units of the same company; or different companies; and addresses the issues of security, policy, payment, membership, and so forth that arise in these settings.
- Uses standard, open, general-purpose protocols and interfaces.
- Allows its constituent resources to be used in a coordinated fashion to deliver various qualities of service, relating for example to response time, throughput, availability, and security, and/or co-allocation of multiple resource types to meet complex user demands, so that the utility of the combined system is greater than that of its constituents [112].

Grid computing is designed to support the access to the idle high-end computation resources. More generally, grid computing provides an infrastructure to utilise the dynamic and inexpensive resources in the network to achieve common goals，for example scientific calculations or physics simulations.

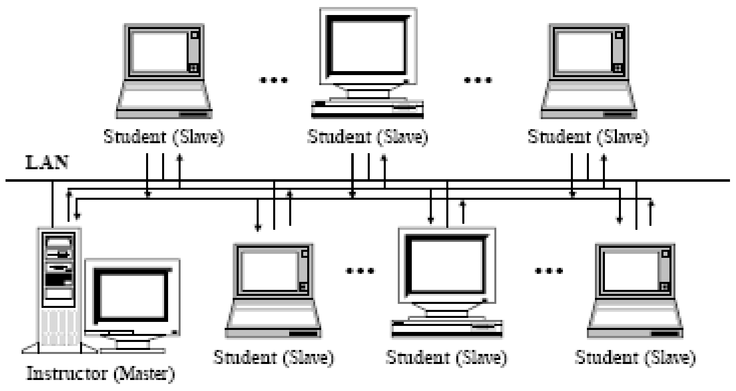


Figure 3.1 Grid Computing Example [114]

Figure 3.1 illustrates a university grid computing example. One machine works as a master PC where the master controller program of the middleware is installed. The master program controls the distribution of the tasks and data, and retrieval of the results of the distributed computation from each of the slave machines connected to this grid. A slave machine requests jobs from the master. The master machine selects a target data set that has not been processed and sends the data and the task to the slave machine. In the university environment, the grid machines tend to be uniform and often form a cluster in the same location. However, the grid computing can extend out of university campus and even countries. It could also be implemented on a mixture of different hardware platforms in a business IT environment.

In [115], the author analysed 5 types of grid computing applications as follows:

1. Distributed supercomputing applications which use grids to combine substantial computational resources in order to tackle problems that cannot be solved on a single system.
2. High-throughput computing applications, which use the grid to schedule large numbers of loosely coupled or independent tasks and put unused processor cycles (often from idle workstations) to work.
3. On-demand applications, which use grid capabilities to meet short-term requirements for resources that cannot be cost-effectively or conveniently located locally. These resources may be computation, software, data repositories, specialised sensors, and so on. In contrast to distributed supercomputing applications, these applications are often driven by cost-performance concerns rather than absolute performance.
4. Data-intensive applications, which focus on synthesising new information from data that is maintained in distributed geographies.
5. Collaborative applications which are concerned primarily with enabling and enhancing human-to-human interactions. Such applications are often structured as a virtual shared space. Many collaborative applications are concerned with enabling the shared use of computational resources such as data archives and simulations.

Grid Computing requires that the network resources can be explored and allocated to the end users dynamically. QoS, especially inter-domain QoS guarantees are also important because Grid Computing often requires reliable communication and normally spans more than one domain. Meanwhile, it is necessary to consider how to organise the available resources to process tasks in the efficient way.

E-business

New forms of e-business [116] working are becoming popular, such as the formation of extranets, whereby users from different organisations choose to work together to achieve some common goals. The partners of the e-business might be located in different areas and belong to different organisations. They communicate through the Internet. Meanwhile, e-business applications also use common resources located in the Internet. A resource might be a database storing huge amounts of data or a super CPU processor.

Service oriented architectures (SOAs) have been adopted for automatic e-business both within and across organisational boundaries [117]. A SOA based e-business normally relies on web services (i.e. Web Service Description Language (WSDL) [118]) and the Simple Object Access Protocol (SOAP) [119].

Consider an example of a loan procurement application implemented by a company named “AutoLoan” [120] in Figure 3.2. After receiving the customer application, a series of tasks start. AutoLoan will leverage several trading partners (United Loan and Star Loan) who provide the actual financing. It also needs to work with the existing information systems and legacy applications for customer information, credit ratings, etc. In addition, AutoLoan requires the system to support interactions with people, such as customer service representatives. Here several services are orchestrated to provide the customer an end-end service. Customer information needs to be stored in a database (normally a huge storage capacity is necessary for the banking system), the credit rating and risk estimation also requires CPU capacity for calculation. There are dependency relationships among tasks in the service orchestration. For example, to rate the customer’s credit, customer information must be available in the Customer Database and accessed by the rating module. Also, the loan offer can be made to the customer leveraging with trading partners, until the credit rating completes and the risk estimation is provided. The loan process is represented in Figure 3.3. During the job processing, the communication links among different departments and organisations need to be created for the data transactions and message exchanges.

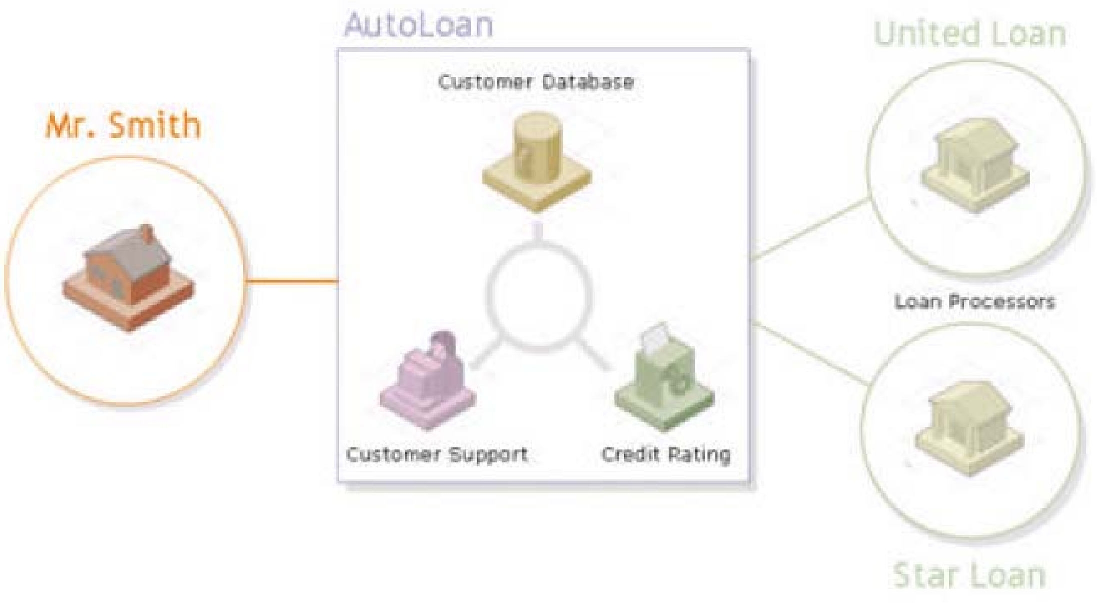


Figure 3.2 An SOA E-business Scenario [120]

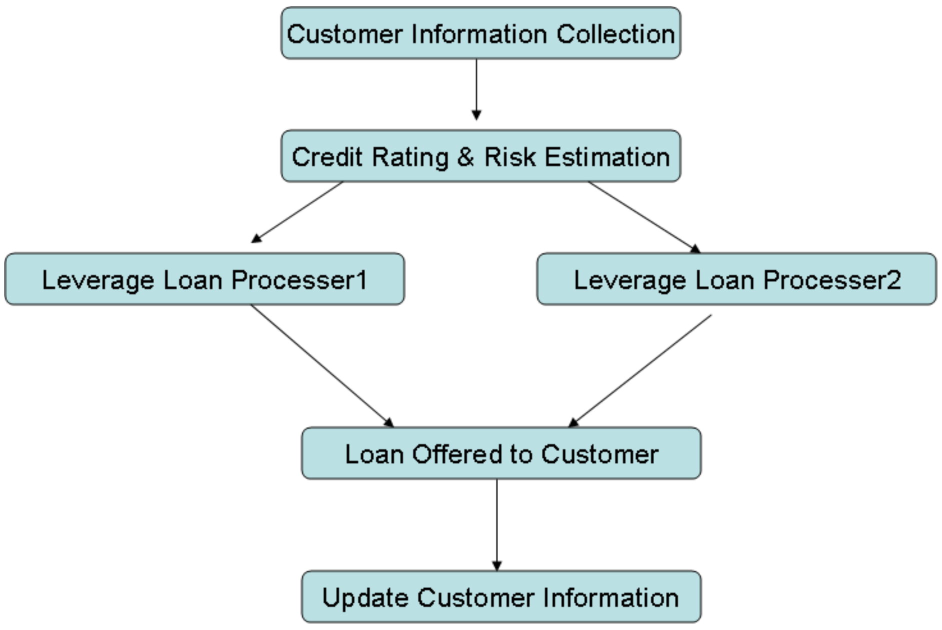

Customer Information Collection
Credit Rating & Risk Estimation
Leverage Loan Processer1
Leverage Loan Processer2
Loan Offered to Customer
Update Customer Information

Figure 3.3 Loan Process

The above scenario shows an example of the SOA based e-business. Normally the SOA e-business relies on Web Services [121]. Currently, the Business Process Execution Language for Web Services (BPEL) is adapted to provide enterprises with an industry standard for business process orchestration and execution. A web service is invoked by a BPEL interpreter using the Simple Object Access Protocol (SOAP), which defines the exchange of messages that are encoded in the eXtended Markup Language (XML) [122]. From a technical perspective, BPEL offers a standard language for defining how to send XML messages to remote services, manipulate XML data structures, receive XML messages asynchronously from remote services, manage events and exceptions, define parallel sequences of execution and undo parts of processes when exceptions occur

[117]. Because service orchestration normally spans autonomous domains, WS-Security [123] is often used with BPEL to provide the encryption of SOAP messages and proof of origin.

The SOA e-business is a very dynamic environment where the customer applications might arrive at anytime. Huge resource (CPU, storage) usage requires the resources (provided by SP) to be involved. The resource availability and network reachability also changes dynamically. As a result, a mechanism of providing reliable, secure and dynamic communication is necessary. Meanwhile, it is required that the available network resources can be explored and allocated for customer applications. A workflow management mechanism is necessary for the system to arrange the task processing and resource scheduling. Lastly, this mechanism must be automatic and on-demand to be feasible in the dynamic environments.

3.2 Existing VPN Approaches and Limitations

As IP is not designed to provide secure and reliable communication among a wide range of public networks, the Internet itself cannot satisfy the requirements of the new applications mentioned above. VPNs based on MPLS provide a solution for these new applications because MPLS-VPN transport is able to provide a controllable means of supporting particular QoS requirements among the community members by using connection-oriented delivery while the same safety as layer2 can be provided. Additional cryptograph technologies can also be implemented such as in IPSec VPN [124].

Considerable effort has been devoted to the formation and maintenance of VPNs within an operator's domain. These VPNs are typically established by manually configuring various network resources. This is acceptable for long-term connectivity such as inter-connection between enterprise campuses. However, new forms of working are likely to have resource usage demands that do not fit well with this arrangement. In these cases it is desirable to establish community relationships of a transitory nature.

In addition, inter-operator MPLS signalling piggybacked on BGP routing messages provides an opportunity for LSPs to straddle provider domains. However, little attention has been paid to the management of these inter-domain pathways. The problem is compounded by the lack of inter-operator QoS support, due to limited trust that operators possess in each other. Only specific peering relationships exist. For example, a company hopes to build a VPN between the branch in London and the branch in Beijing, the VPN has to traverse the border of the BT and Chinese Telecom. Although the agreement might be made between the service providers, it takes a long time and requires complex human operations.

As analysed in the former chapter, seeing these limitations, a number of Dynamic VPN initiatives have recently begun to emerge. Examples include Closed User Groups (CUG) [71][72], the work of Kindred and Sterne[73], Jia[74][75] Leetnet [125] and the Defense Research & Development Canada (DRDC) demonstrator[76].

As introduced in Section 2.8, Closed User Groups (CUG) can provide flexible dynamic formation and self-management of distributed collaboration networks. However, mechanisms for resource discovery and reservation have not been hitherto integrated into the CUG architecture.

In the case of Kindred and Sterne, the focus is on inter-provider security. Their work examines the issues associated with trust in dynamic communities, or enclaves. However, they do not address how the infrastructure would be set up to meet particular traffic engineering requirements. In addition the community manager is a human operator, which limits the scalability and responsiveness of the system dynamics. A similar human control architecture is presented in [78].

Yuxiao Jia *et al* have taken a different focus [74][75], concentrating on the traffic engineering aspects of VPNs. In their case, the emphasis has been on predicting traffic patterns and using this information to ensure the MPLS-DiffServ enabled VPN is able to deliver the required QoS. The VPNs are not dynamic, nor are they inter-provider solutions; rather it is the allocation of communications resources that is adapted to meet changing user needs. A more flexible VPN framework is considered with Leetnet [125]. This is a dynamic network of small networks connected securely via the Internet using IPSec. Managed by a central web interface and database, all networks can establish secure IPSec VPN connections via the automated central web server. The limitation is really that it is not MPLS based, and does not couple the VPNs with the underlying resources.

An approach that is perhaps more similar to the author's proposal is based on the Defence Research Establishment Network (DREnet) [79][80][81] which interconnects the headquarters of the Defence Research and Development Canada (DRDC) Agency with its five research centres (section 2.8, Figure2.14). Nevertheless, this scheme neither includes provision for negotiating and assigning resources, nor does it provide the benefits of managed MPLS inter-connectivity.

A final state-of-the-art example is the architecture put forward by Fujita *et al* [78]. They describe an interesting scheme to deploy overlay networks that achieves both policy updates and Layer 3 topology reconfiguration in a scalable manner. Each node retrieves the IPSec parameters required for sending packets to a destination node on-demand using an extended Domain Name System (DNS) behaviour, eliminating the need for advertisement-based updates when the membership changes. However, once again the approach is neither coupled with MPLS and its associated traffic engineering benefits, nor with the inclusion of processing resources as specialised group members with particular responsibilities.

3.3 Dynamic Resource Scheduling

In the Dynamic VPN (DVPN) system, a customer service requirement will trigger a series of tasks which is defined as a "job". A dynamic VPN instance will be created for each individual job. A job will typically consist of a number of tasks. There might be the data dependences among these tasks. At the same time the jobs (VPN instances) are independent of each other. Each job can be represented by a DAG [90] which consists of several dependent sub-tasks. Resources are necessary for carrying out jobs and the resource requirements can be specified at either the job level or the sub-task level. Figure 3.4 shows an example of the DVPN Job Scenario. There are two jobs (Job1 and Job2). There is no relationship between Job1 and Job2. Job1 consists of 4 tasks and Job2 consists of 3 tasks. The directed lines represent the relationships among tasks. For example, T1.2 and T1.3 cannot start until T1.1 completes; T1.4 (the end point of Job1) cannot start until T1.2 and T1.3 complete.

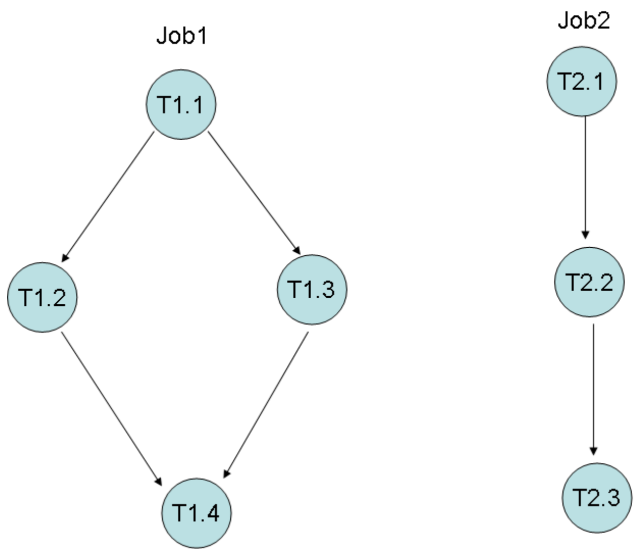


Figure 3.4 DVPN Jobs Scenario Example

The goal of the workflow management is to provide service for customer requirements in the most efficient way (i.e. the minimum overall completion time), in other words, how to schedule the resources to tasks optimally. New applications (such as grid computing and automatic e-business) utilise the network resources and orchestrate the equipment involved to complete certain jobs efficiently. This requires that the workflow management is automatic and dynamic.

Considering an example in Figure 3.5, User 1 requires service (Job1) from SP, Resource 1 is the current available resource in the SP network. During the time SP provides service for Job1, User 2 submits another service application (Job2), It also happens that resources (i.e. Resource2) change their status (i.e. from unavailable to available). After jobs complete, the resources occupied need to be released. Once the job scenario and the resource availability changes, the current in-use resource scheduling scheme might not be suitable and needs to be adjusted to react to real time situations. Meanwhile, it is also possible that during the time that jobs are processed, the network reachability changes, for example data links failures. The resource scheduling algorithm also needs to be feasible under dynamic network environment.

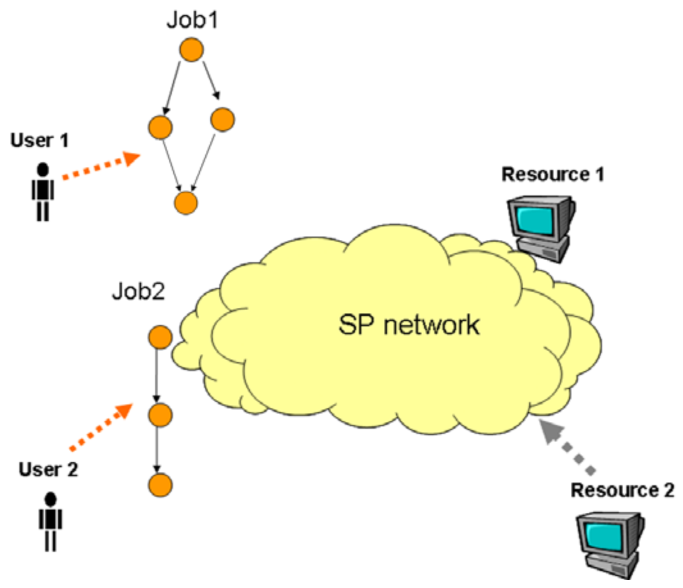


Figure 3.5 Dynamic VPN Environment Example

Resource scheduling in such a heterogeneous system has been a well-known NP-complete problem [17][18][95]. A number of researchers have addressed the resource-scheduling problem for static cases of dependent / independent mixed jobs [18][84][85] which are similar to the DVPN scenario. However they are static in the sense that the characteristics of the jobs (i.e. arrival time, processing complexity) are known in advance.

In the DVPN system, the job scenarios are dynamic and consist of dependent and independent tasks together where none of the existing static solutions are suitable. In [97] a job can have multiple sub-tasks, and these tasks can have data dependencies, which is similar with the scenarios in the DVPN, but each application makes its own scheduling decisions and there is no the central coordination. This makes the architecture not suitable for the DVPN system where the dynamic VPN manager (DVM, section 4.2) provides centralised resource scheduling for the entire network exploiting global knowledge.

3.4 Inter-domain Cooperation

Newly appearing applications may require services that are provided across multiple AS domains. For example, in global grid computing, processors (i.e. CPU processor, storages) located in various domains can cooperate to perform certain jobs. VPN links would normally be necessary for the data transaction among resources. It is also required that the resources belonging to different domains cooperate to provide the service together. For this reason, the cooperation is considered a fundamental requirement within the context of this thesis.

In this thesis, the term domain refers to an Autonomous System (AS) domain. The Service Provider (SP) may manage one or several AS domains. Each AS domain has its own policy, which is administrated by the SP.

Normally, the individual AS only considers itself. Given the scenario shown in Figure 3.6, a failure happening in the communication between AS4 an AS3 due to some physical fault, to continue to provide the communication service between AS4 and AS3, an alternative link needs to be created to replace the failed one. In this scenario, a possible solution is to build a link along the path AS4-AS1-AS2-AS3. However, it is no benefit for AS1 and AS2 to provide this link, they do not receive service while their communication resources are occupied. As a result, AS1 and AS2 are likely to refuse to provide such a transit service. The attempt to restore the broken communication appears to be impossible.

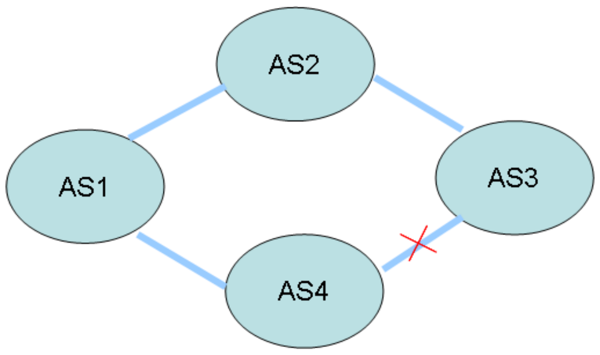


Figure 3.6 Inter-Domain Connectivity

The proposed DVPN system is designed to support orchestrated computing. The entities involved might located in different ASes which use different policies. On the other hand, reliable and QoS guaranteed inter-domain communications are required. Some relevant research has been analysed in Section 2.7 and Section 2.8; however, due to the lack of automatic trust and negotiation mechanism, none of the exiting solutions can avoid the problem arising from the AS selfish behaviours.

3.5 Aims of the Proposed Inter-Provider Dynamic VPN Architecture

In order to extend the flexibility of VPNs in terms of connectivity with QoS support and in terms of their rapid, on-demand, creation, adjustment and removal, a new dynamic VPN solution is necessary. Meanwhile, as new applications require on-demand network resource services, service provider can also provide the resources (i.e. CPU, Database) as well as bandwidth. For this reason, this architecture will also provide functions that exploit available resources and allocate them to the customer applications dynamically. This service will bring new revenue for the Service Provider. As new applications require reliable and on-demand services that can be provided across the domain borders, the proposed architecture needs to provide a mechanism to encourage the cooperation among ASes and make the overall network more reliable and efficient.

This new framework is designed to capitalise on existing VPN mechanisms. However, activities that are hitherto performed manually are augmented and even replaced by an automated infrastructure capable of inter-operator negotiation as well as supervising local dynamic configuration management of the VPN. The proposed dynamic VPN solution is envisioned to achieve these following aims:

1. The extension the use of newly deployed Multi-Protocol Label Switched (MPLS) infrastructure for the formation and operation of highly dynamic, automatically administered, VPNs.
2. The straightforward adjuncts to the control plane that can be used to support extranet-based on-demand services compatible with “grid computing” and other distributed, collaborative activities.
3. The methods for automatic support for generation of the on-demand requests for communication and processing resources will be derived from specifications of the business processes.
4. The mechanism to encourage inter-domain cooperation is needed so that the overall network reliability and efficiency can be improved.
5. The consolidation of technologies that have hitherto been treated separately. In deliberately following this strategy, the aim is to provide a clear migration path.

In the following chapters, a new inter-provider dynamic VPN is proposed based on the above analysis and a new dynamic resource-scheduling mechanism is studied. The author also proposes a mechanism to encourage the cooperation among ASes.

Chapter 4 Inter-Provider Dynamic VPN Framework

This chapter describes a novel dynamic VPN (DVPN) architecture supporting Inter-Domain Distributed Computing for applications such as Grid computing and e-business. In the architecture, the Virtual Private Network (VPN) might be created and operated dynamically while the security and QoS are satisfied among multiple domains. In addition, value-added network resources (i.e. CPU processor or storage farms) can be provided according to the requirements of the customer applications.

4.1 Motivating Example: Global Grid Computing

As introduced in Chapter 3, Grid computing is one of the most important applications proposed to be undertaken by the DVPN system. Consider the scenario represented in Figure 4.1, where a job is launched by user A. This job consists of several sub tasks, T1, T2, T3, T4, T5 and the result will be transmitted to user D. According to the task dependence represented in Figure 4.1, T5 can start when T4 and T3 have both completed; T4 can start when T2 has completed, and T2 and T3 can start when T1 completed. Also, T3 is independent of T2 and T4. In this example, user A and user D are located in different domains (User A is in AS1 and User D in AS2). There are value-added processing resources available at B in AS1 and C in AS2.

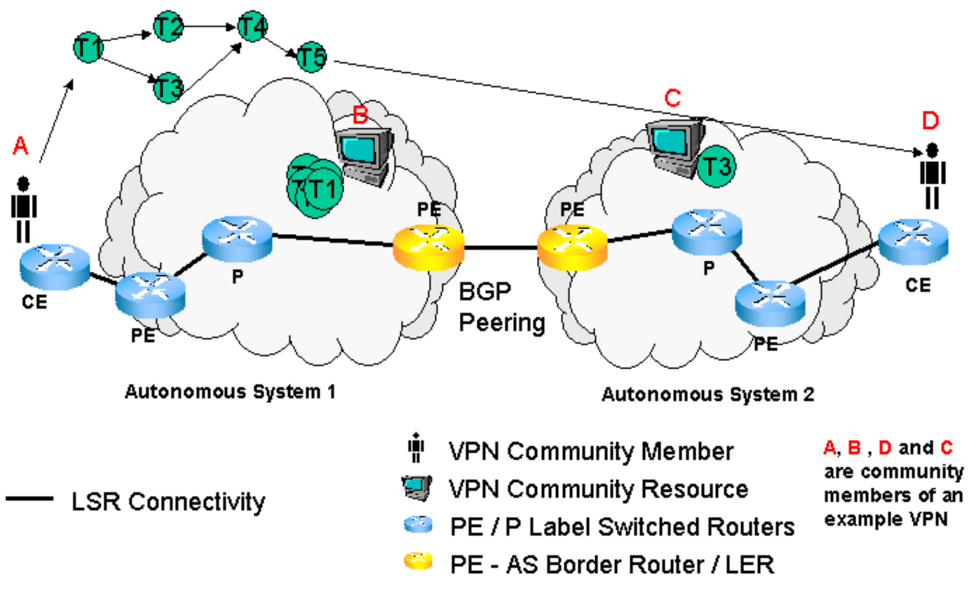


Figure 4.1 Global Grid Computing Example

Assume that this job is calculation intensive and extra processing resources are required to carry out the calculation tasks. Since some tasks are parallel such as T2 and T3, placing independent tasks on different processing resources may obtain better performance. In this example, a possible scheme is to process T2, T4 in resource B and T3 in resource C.

According to the above description, to process this job, a grid involving user A, user D, resource B and resource C is required. As mentioned before, a VPN can provide security and low cost links among the grid nodes. In order to use the network resources (i.e. bandwidth) efficiently, the VPN links are created only when the communications are necessary. After the data transitions are completed, the links are removed. At the same time, to better utilise the value-added resources such as processors, these resources can also be provided to the grid on-demand and dynamically. Meanwhile, the QoS requirements should be satisfied even when a VPN link crosses multiple domains (i.e. AS1 and AS2 in this example).

However, traditional VPNs are static and based on human operations, which limits the scalability and the system dynamics. In the above scenario, to perform the grid computing job, given a traditional VPN, user A would need to send a request to the network manager to create a VPN and the VPN links would be configured by human operators. Once the VPN is created, all the VPN links and the network resources are allocated for the whole VPN lifetime. It is difficult to introduce new resources to speed up the grid computing calculation although there could still be available redundant resources. Automatic releasing of the unused network resources is also impossible, because the network manager is a human operator, there is no such mechanism to manage VPN and network resource automatically.

Meanwhile, in this example, the destination (user D) is in a different AS from user A. The VPN includes a link that must cross the border between AS1 and AS2, which requires complex processing for the traditional VPN technologies. It is also difficult for user A to utilise the resources at C because entities are invisible outside its own AS.

Based on the above example, traditional technologies cannot efficiently provide dynamic and inter-domain services for the global computing. This scenario highlights a number of issues regarding collaboration of entities located in different ASes given that each AS will have its own access control policies and conditions of use. The challenges are to provide:

1. Dynamic resource exploitation and allocation for the customer applications among multiple domains.
2. Customer-active QoS guaranteed VPN management crossing domain borders.
3. A negotiation mechanism for providers regarding the dynamic inter-domain services.
4. Ensure that the providers' security requirements were fulfilled.

A suitable architecture needs to be able to provide the inter-domain dynamic connectivity and resource management; the rest of this chapter describes a novel inter-domain MPLS Dynamic VPN architecture.

4.2 The Inter-Provider Dynamic VPN Architecture

Based on the requirements outlined in Section 4.1, the author proposes a novel MPLS based inter-domain dynamic VPN (DVPN) solution. The proposed DVPN framework, shown in Figure 4.2, is designed to provide dynamic and multiple-AS VPN reachability while trying to utilise the existing network technologies and equipment. The DVM (Dynamic VPN Manager), which is the key contribution of this architecture, forms the central manager of the dynamic VPN, supporting user community management. This architecture makes use of existing forwarding and signalling methods or schemes as much as possible under consideration in IETF RFC documents [28][126].

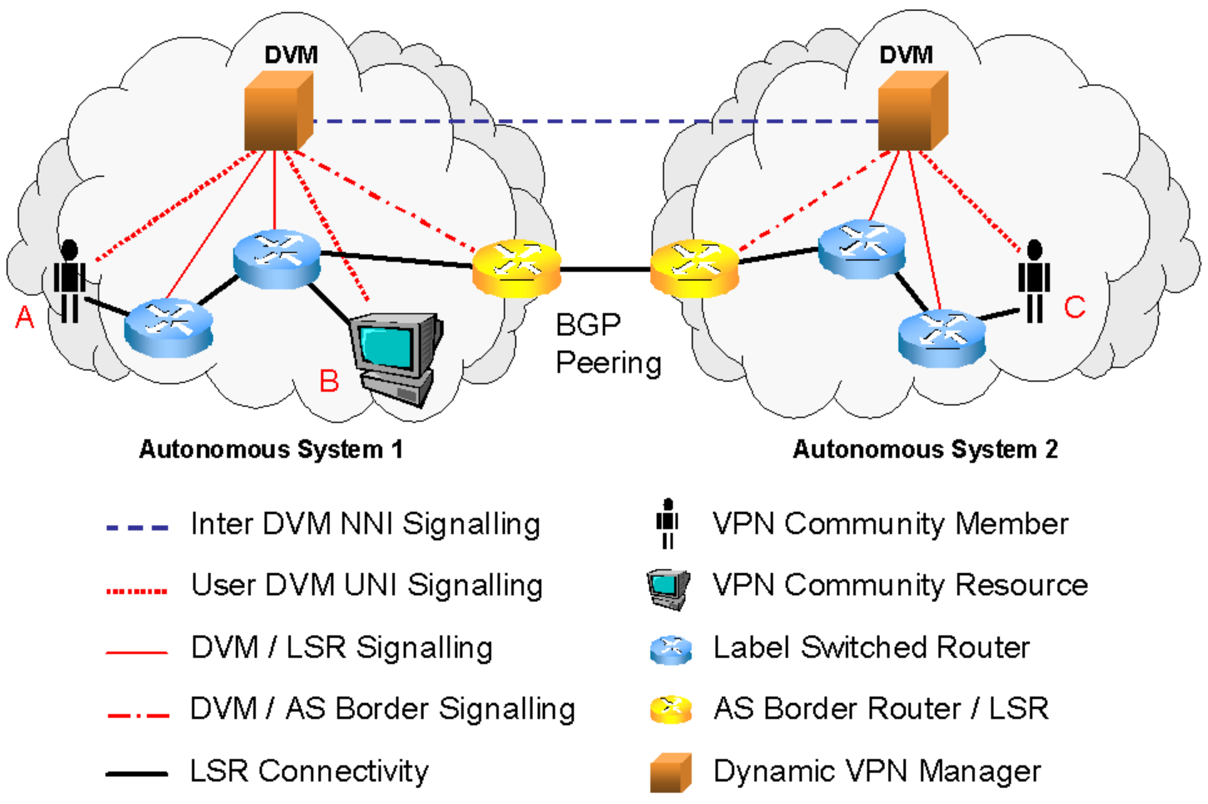


Figure 4.2 Dynamic VPN Architecture with DVM

The DVM is the central manager of an AS. Each DVM may monitor and manage multiple VPNs concurrently, some of which will be confined to a single AS, whilst others may span across multiple AS boundaries. The users request services from the DVM to build/remove a VPN, join/leave a VPN, select QoS constraints and specific service resources. A User Network Interface (UNI) exists between the user and DVM; users send information and service requests to the DVM, while the DVM manages the user population and provides VPN management services to them. The DVM collects network information such as topology details (for its local AS) and the availability of network resources such as CPU, bandwidth and database resources, so that VPN links can be suitably configured. Based on the network conditions and the user requirements, the DVM will adjust the VPN and allocate resources accordingly. The DVM also provides value-added resources to users, such as databases for archiving and CPU engines to be allocated to specific VPN communities in support of their operational needs.

The underlying network of this VPN can consist of different ASes, as illustrated in Figure 4.3. The technology for each AS might be different. For example, AS1 is an MPLS/IP network, AS2 is a "traditional" IP network and AS3 exists on top of an Asynchronous Transfer Mode (ATM) infrastructure. There is one DVM for each AS and each DVM only manages equipment belong to its own AS. DVMs communicate with others through the logical links. For example, to build a path from AS1 to AS3 via AS2, DVM1 will send the service requirement to DVM2, and in turn, DVM2 will send the service requirement to DVM3. If DVM2 agrees to provide transit services from DVM1 to DVM3 while DVM3 also agrees to provide service for this path, one path from AS2 to AS3 and one from AS1 to AS2 are constructed. Each DVM only manages the connection tunnel establishment within its own AS. This is done by signalling with the local switching infrastructure based on messages native to the cross-connect equipment or via network management, such as Simple Network Management Protocol (SNMP) [127], as necessary. In this example it would not be possible to have a layer2 link between AS domains due to the different layer2 technologies in use in each of them. The ASBRs would need to exchange VPN traffic at layer3. The VPN tunnels between domain borders are created according to the negotiation among DVMs of different ASes. If a VPN path spans across different ASes, the DVM in one AS will build a link from the appropriate ASBR (AS Border Router) in its own AS to the advertising BGP router in the next AS along this path. How the data forwarding plane is configured for this VPN in the next AS is determined by the next DVM. Although QoS constraints may be requested, how a given DVM fulfils its commitments is an opaque local decision.

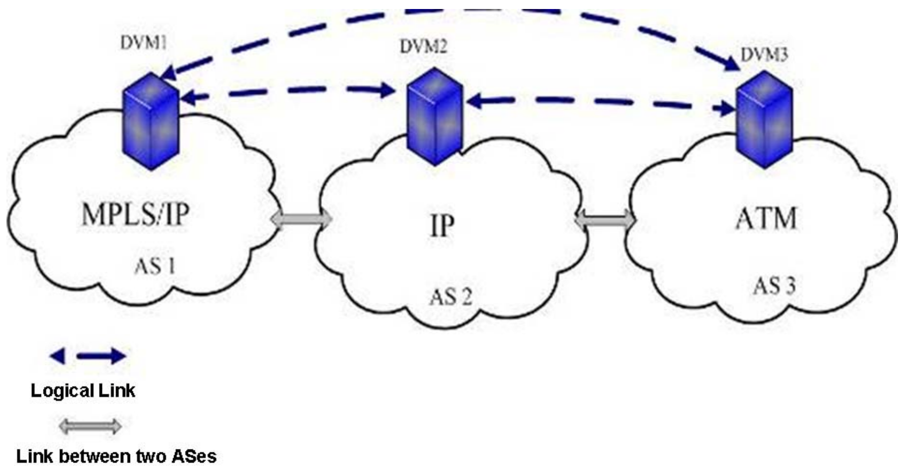


Figure 4.3 Underlying Network

4.3 Dynamic VPN Manager

4.3.1 DVM Functions

The DVM acts as an agent, liaising with separate resource and connection management software to deliver the infrastructure necessary for the business process action(s) that are to be undertaken. The DVM manages its VPN based on the network conditions and user service requirements. As the network status changes due to traffic fluctuations or changes in the user population, the DVM is able to make some adjustments according to the changes. For example, DVM would be able to order the connection management to perform the LSP path re-routing to alter the shape of the VPN by moving LSPs away from congested links.

Figure 4.4 represents the function blocks of the DVM. The DVM UNI (user-network interface) signalling provides communication between the DVM and users. Users can submit their applications to the DVM while the DVM can send management messages to users. Connection management liaison is for communicating with the connection management system, the DVM sends commands to the underlying network devices (where the connection management software is) to order the PE to perform VPN actions such as building/removing VPN links. The Resource booking function is for the DVM to book network resources for user applications. The DVM explores the available network resources and allocates the resources to users according to the user applications and the network conditions. Through the NNI (network-network interface), DVMs can find other DVMs for VPN discovery and negotiation. For example one DVM might require services from another DVM with network resources outside its own AS, or a VPN link bypassing other ASes. To build a VPN link across AS borders, a DVM needs to liaise with the ASBR (or the CM managing the ASBR) to build a LSP tunnel from the local ASBR to the ASBR in the next hop AS (assuming both Assess support MPLS). The AS Border Operations function provides this role.

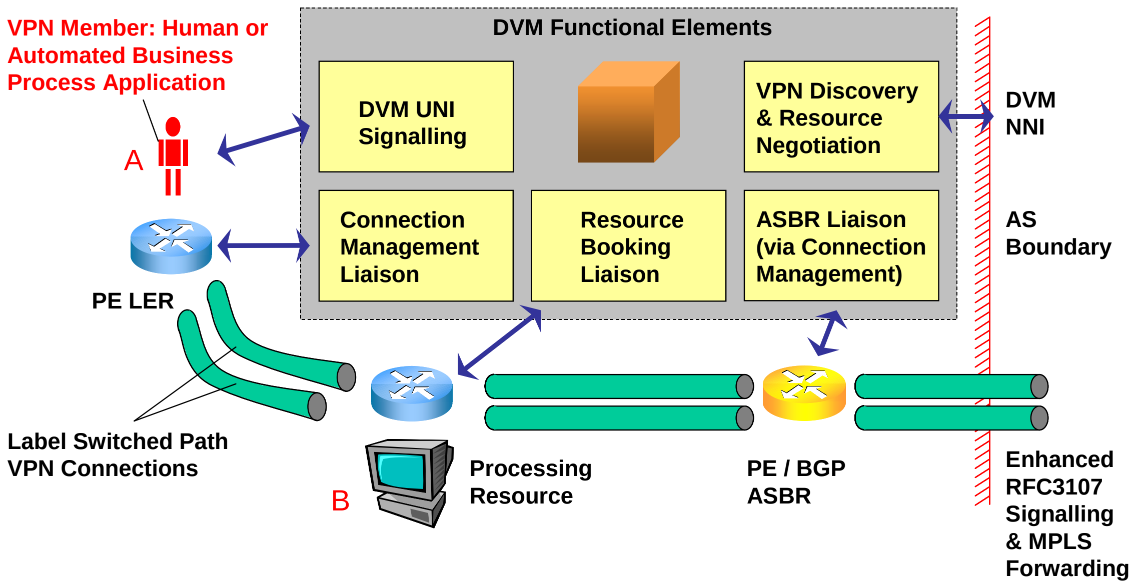

VPN Member: Human or Automated Business Process Application
A
DVM Functional Elements
DVM UNI Signalling
VPN Discovery & Resource Negotiation
DVM NNI
Connection Management Liaison
Resource Booking Liaison
ASBR Liaison (via Connection Management)
AS Boundary
PE LER
Label Switched Path VPN Connections
B
Processing Resource
PE / BGP ASBR
Enhanced RFC3107 Signalling & MPLS Forwarding

Figure 4.4 The Functions of The DVM

4.3.2 The Internal Architecture of the DVM

Figure 4.5 illustrates the architecture of the DVM and the interactions and data flows among all components inside it.

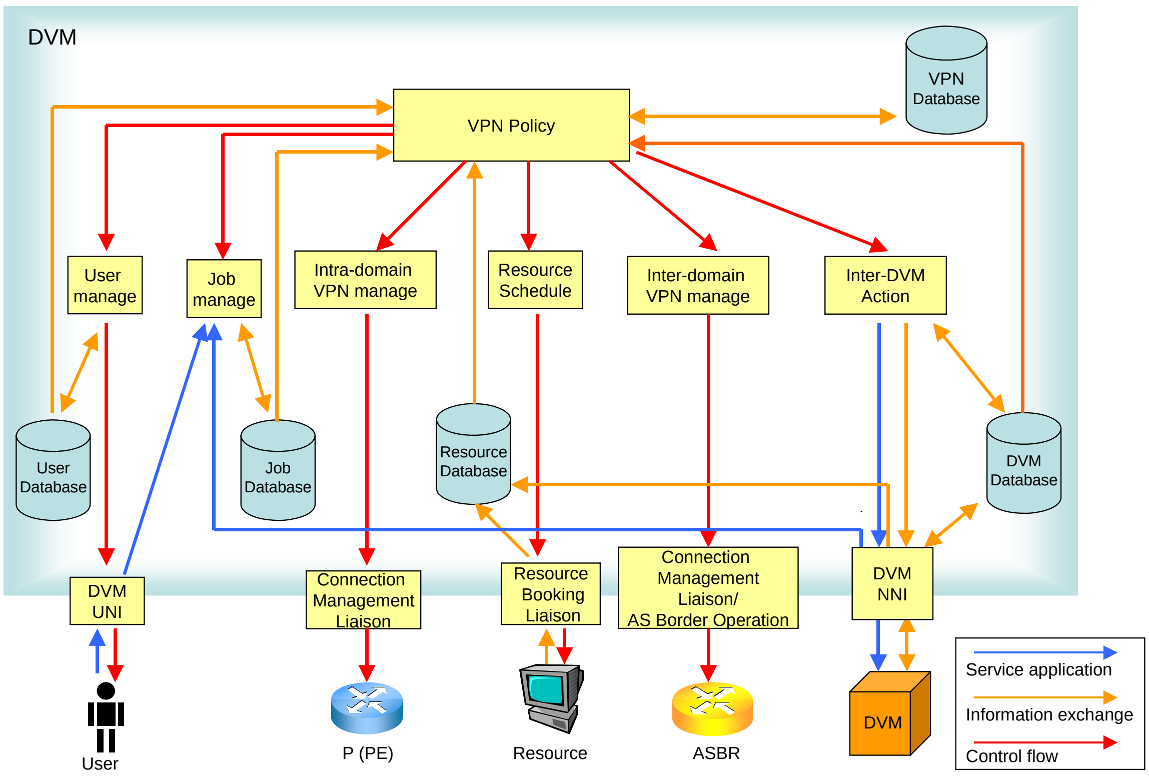


Figure 4.5 The DVM Architecture

The user database stores the user information shown in Figure 4.6. The user account stores the user ID and certification information; only the users with appropriate account information can access DVM services. The user attribute stores information such as user's location (the address) and the user's PRI (represent the user level and rights), there may be additional attributes. VPN ID lists the VPNs of which a certain user is a member. According to the VPN ID, detailed information of this VPN can be investigated in the VPN database. The user information will be updated by the "User manage module". The user manage module also manages the users according to the information in the user database, for example it decides what kinds of services a user can access (i.e. a user paying a higher service fee might have higher priority).

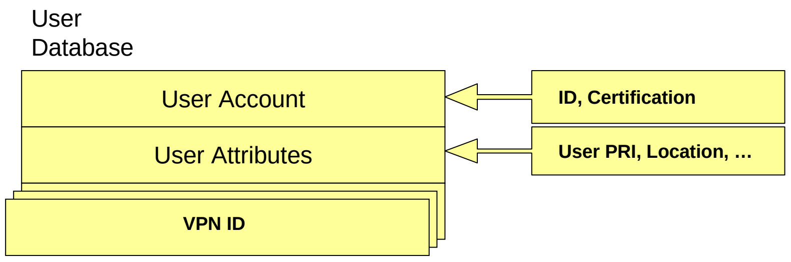


Figure 4.6 The User Database

The job database stores the jobs being served and waiting to be served. A user submits a service application (job) to DVM through the DVM UNI. Applications may also come from the DVMs of other ASes through the

DVM NNI. The job information is stored in the job database. The information is employed by the "VPN policy" to govern the VPN policy. The "VPN policy" will tell the "job manage" its decision (which jobs will be served and how they will be served). Then the "Job management" will update the job database according to this information. Figure 4.7 shows an example of job database where job1 is submitted by user1, it needs CPU resource; the necessary capacity is 13 (representing the CPU capacity), this job starts at 00:30 and its deadline is 00:41, the current status of this job is "running".

Job	Applicant	Resource Requirement	Capacity	Start time	Deadline	Status
Job1	User1	CPU	13	00:30	00:41	running
Job2	User2	CPU	14	00:32	00:38	wait
Job3	User2	CPU	22	00:40	00:42	wait
Job	……	......	......	......	……	……

Figure 4.7 The Job Database

The resource database stores the resource information. Figure 4.8 shows an example of the resource database. The resource ID, location (address), type, quality (i.e. robust), capacity and price are listed in the database. Remote resources advertised by DVMs of other ASes also are listed. The VPN policy will make decisions on which resource and when this resource will be allocated to a certain VPN. The available period represents the time period when a resource can be used. As resources will not be occupied permanently by a certain applications, the time is divided into a series of periods and resource is booked for specified periods. For example, R1 is available between 00:30-00:35, 00:40-00:45 and 00:45-00:50, but during the period 00:35-00:40, this resource will not be available for the reason that some application will occupy it according to the "VPN policy".

Resource	Address	Type	Quality	Capacity	Available Period	Price
R1	10.0.0.3	CPU	1	40	00:30-00:35 00:40-00:45 00:45-00:50 …..	20
R2	10.0.4.18	Memory	3	4000	……	40
R3	10.0.16.67	CPU	2	33	……	30
R…	……	…	…	…		…

Figure 4.8 The Resource Database

The DVM database stores the information of DVMs in other ASes, shown in Figure 4.9. The information includes the DVM account information that is necessary for identification and authorisation. There are also

DVM attributes (address, PRI) which will be used by the VPN policy to make the inter-VPN policy. Extension attributes may also be available.

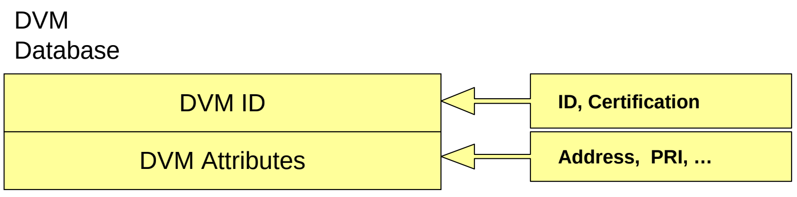


Figure 4.9 The DVM Database

The VPN database stores the information of all the VPNs. Figure 4.10 shows the architecture of VPN database. The VPN ID is globally unique (the AS number added to the local VPN number). For example, for a VPN was built by the DVM in AS 100 with the local ID 35 (assigned by the DVM in AS 100), its global VPN ID is represented as 100.35. The Host DVM is the DVM which built and managed this VPN, any VPN actions related to this VPN such as VPN creation/removal must be performed by this DVM. The Administrator of the VPN might be a human operator or a software entity. With an appropriate account, an administrator can manage the VPN (his application for administrating this VPN will be accepted by the DVM). Access policy is a policies matrix. Each user has different access rights to a certain VPN. For example, the user with administrative right is the most powerful user, it can revoke out a user, even terminate the VPN while users with normal user rights can only join or leave the VPN and use the resources in the VPN, and users with basic user right can only join and leave this VPN but cannot access the resources. Members of these VPNs are also stored in the VPN database. The user account stores the user identity information. The user address represents the location of this user, the access right defines the user's right to the VPN and PRI defines how this user will be treated when it competes with others.

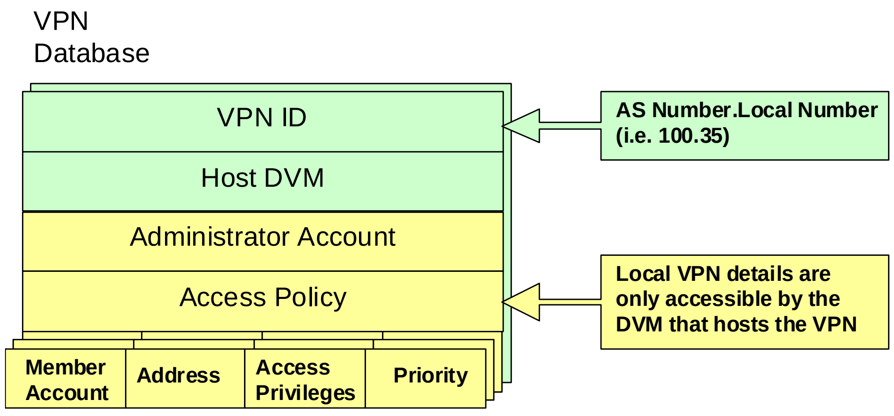


Figure 4.10 The VPN Database

In this DVM architecture, the "VPN policy" makes the whole VPN policy based on the information of the databases described above. Then the "VPN policy" tells "user manage" how to manage users. It also triggers "job manage" to update the job database, for example, which job will be executed at what time, and the resource it will occupy. The "VPN policy" will also tell the "resource schedule" to book resources for VPN services. The resource database will be updated in the case of resource changes (for example some resource is booked or released).

The “VPN policy” will tell the “intra-domain VPN manage” how to manage VPN links within the local domain, then the “intra-domain VPN manager” sends commands to PE routers (its connection manager) asking them to execute VPN actions (such as creation/removal VPN links). If a VPN is an inter-domain VPN, “VPN policy” also needs to tell “inter-domain VPN manager” how to configure ASBR for this VPN link across the AS border. VPN databases will be updated after successful VPN actions.

DVM also communications with other DVMs in other AS to detect, negotiate, exchange information, these actions are executed by the “Inter-DVM action”. The messages between DVMs are sent and received through the DVM NNI. The “VPN policy” may also require the “Inter-DVM action” to carry out certain actions. For example, if the existing local resources are not enough to meet the users’ requirement, the “VPN policy” will ask the “Inter-DVM action” to find and book resources in other ASes by exchanging information and negotiating with DVMs in other ASes.

4.3.3 DVM Connectivity Management

The DVM liaises with the Connection Management (CM) to perform the connectivity management. The Connection Management Entity (CME) supports the DVM UNI/NNI Entity and the DVM Resource Management Entity in order to provide network connectivity with Service Level Agreement (SLA) and QoS features. The CME employs MPLS to establish Label Switched Paths between endpoints. However, a number of novel extensions to the MPLS control plane are specified and implemented allowing the DVM to have a very high degree of flexibility in connection establishment. Extensions include third party initiated connection establishment, pre-booking and pre-emption of pre-booked resources to provide the flexibility required to efficiently support grid computing and other distributed applications. Resource pre-booking constitutes a major extension to the control plane because future reservations must be taken into account. To some extent, pre-booking can be compared to off-line traffic-engineering, opening new possibilities for efficient planning and optimisation.

Conventional MPLS has CM performed in the ingress LSR of the MPLS LSP, however, scenarios also exist where a connection between two endpoints could be initiated by a third party, e.g. a grid manager that controls transport of data between different clusters. Furthermore, the ability to perform pre-booking of resources is considered important for grid applications. In this case the network must reserve bandwidth for future use, e.g. when a computation finishes. Implementation of pre-booking is a major research challenge with huge implications for existing MPLS connection management procedures.
This entity provides technology and/or operator specific means of interacting with the MPLS connection management software present on the PE devices, i.e. the Label Edge Routers. A typical activity would be to request the PE to set up LSPs between itself and a specified series of other PEs, associated with a particular

VPN label that is added to the FEC to Label binding table at the ingress. This function is also responsible for signalling the removal of LSPs when they are no longer needed. The connection management is informed so as to initiate the necessary PathTear messages. It is needed to be clear that how the communication tunnels are created is the job of the CM (i.e. located in PE devices). The DVM only requests connection service from the CM, it is not the responsibility of the DVM to perform the LSP operations.

4.3.4 DVM Resource Management

A key function of the DVPN is to enable a user to solve problems on a distributed platform in a reliable and confidential (secure) manner within a specified time. User applications could embrace both commercial and academic communities. Typical academic applications may involve climate change calculations. These are processor intensive but not time critical. Conversely, commercial applications are time sensitive and must have guaranteed confidentiality. An example might be the evaluation of airflows over a new commercial airframe. Clearly, the data supplied and the results obtained would be strictly confidential. The operator and the resource infrastructure must have mechanisms in place to guarantee the isolation of this data from anyone outside the VPN, possibly including the operators themselves.

Distributed and parallel computations, mostly scientific ones, need to locate and access large amount of data and to identify appropriate computing platforms on which to execute. Most often, applications need to be carried out dynamically across distributed computing platforms such as clusters and grids. Therefore, it is also necessary to consider the resource management function itself. Techniques such as over-booking of finite resources and the pre-emptive scheduling of tasks of differing priorities will need to be addressed. In the latter case, it could be that a pre-booked low priority task could be interrupted by a transient high priority one.

The DVM can contain a Resource Manager or it is an agent that acts on behalf of the VPN users, liaising with separate resource management islands that may, or may not, be owned by the operator. The Resource Manager (RM), illustrated in the Figure 4.11, focuses on performing the management of the resources under its responsibility. However, a DVM may have access to a number of Resource Managers both within and beyond its own Autonomous System. Associated with a given Resource Manager there will be a number of RM modules that will perform distributed resource management tasks. The resources involved are classified into the following three categories:

1. Computational Resources (CR) including both processors and cache and main memories.
2. Data Storage (DS) resources that include any types of storage devices (hard disk, DVD, and so on). The main measure is the capacity but the characteristics of the device can also to been taken into account when making the resource allocation decisions.
3. Bandwidth (BW) referring to the communications capacity required locally between the RM modules. This is separate from the VPN communication resources, which are handled by the operator's connection management / traffic engineering system.

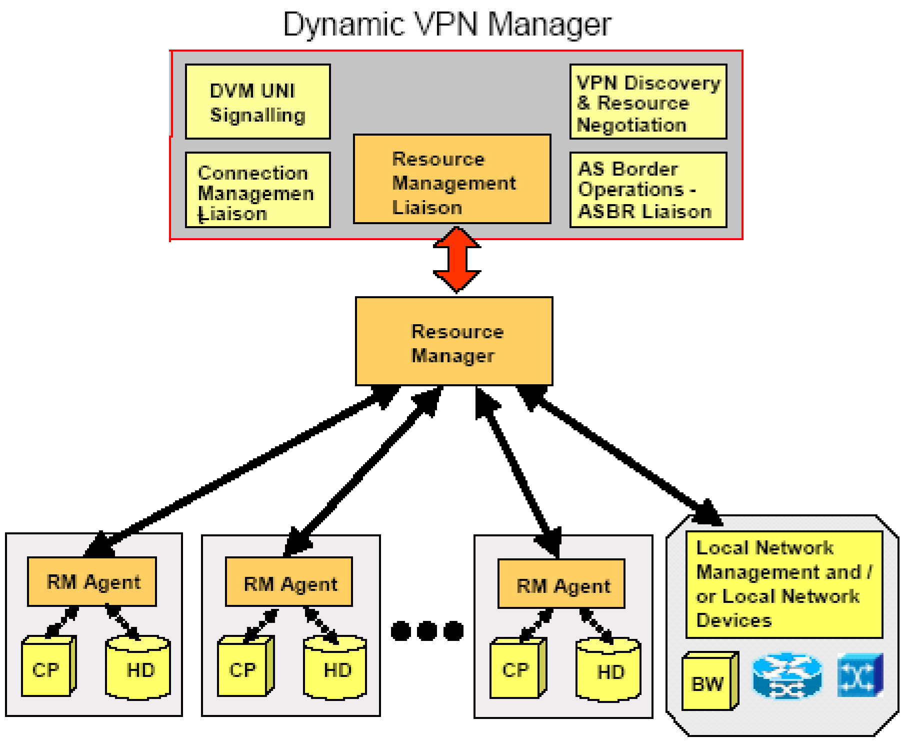

Dynamic VPN Manager
DVM UNI Signalling
VPN Discovery & Resource Negotiation
Resource Management Liaison
Connection Managemen Liaison
AS Border Operations - ASBR Liaison
Resource Manager
RM Agent
CP
HD
RM Agent
CP
HD
RM Agent
CP
HD
Local Network Management and / or Local Network Devices
BW

Figure 4.11 Resource Management Architecture

Usually an application will ask the RM for multiple types of resources – via the DVM. A typical situation to consider is when the user does not know in advance (or at least not exactly) the required resources. In this case the RM and RM modules will provide mechanisms for monitoring the used resources (with the possibility of predicting future needs) and mechanisms to limit the amount of allocated resources.

The RM is in charge of the resource management; the DVM's role is simply to request those resources on behalf of the VPN users. The DVM usually asks for specific functions of the RM (ask for booking/releasing resources, etc). However in several cases the RM may also take the initiative and notify the DVM of any unexpected change/problem in the resources monitored (e.g. a hard disk failure). The RM module will also deal with prioritised scheduling of tasks, security and auditing.

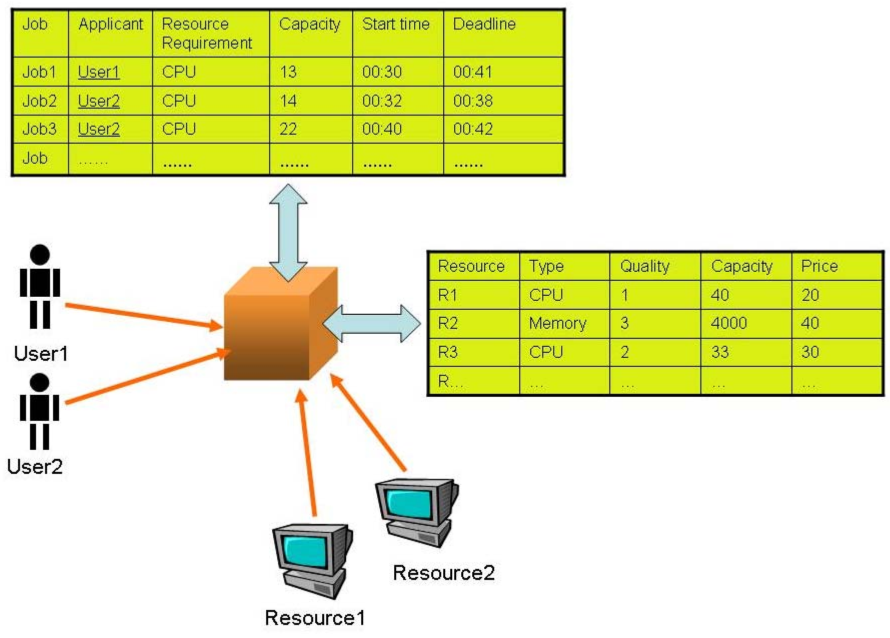
The RM module also has to interact with specific local resources i.e. with the operating systems running in the computers where the resources are. This could be performed directly using any available service or functionalities offered by Operating Systems and other management software, but the possibility of placing an RM agent (proxy module) on computers where the resources reside is also considered in order to give better functionality, control and performance. The RM architecture is therefore decentralised to ensure robustness, scalability and efficiency. Each computational platform under the RM could be equipped with a RM agent and those agents organise themselves in various virtual network topologies sensitive to the underlying physical environment and trigger application reconfiguration when necessary. However, it is beyond the scope of this thesis to consider this matter of that.

In the DVPN, a VPN is formed according to DVM's decisions. The DVM decides which user and resource nodes should be in the same VPN based on the user requirements and network conditions. During the VPN lifetime, it is possible that the resource availability and the customers' requirements change. According to the real time status, DVM will re-arrange its VPNs and resources allocations. For example if a customer application needs more resources than original requested, the DVM might arrange additional resources to this VPN.

As analysed above, the resource booking liaison is responsible for the resource exploration and booking. In the DVPN system, network resources are managed dynamically. The DVM determines the best way to assign these resources to the user jobs. The DVM will also automatically adjust the resource allocation scheme according to the real time status.

The DVM keeps a database record of the network resources. The data in the database is updated by the resource advertisement periodically or when resources are running out. The DVM also keeps a database recording all of the jobs submitted by users. The DVM decides VPN policy (i.e. when to build or remove VPN links, how many resources should be included in one VPN, when to add or remove certain resources into or from a VPN). The resource and job databases are updated constantly, and the DVM manages VPNs according to the real time information.

In Figure 4.12 an example is shown. User1 and user2 submit their job applications to DVM, these jobs are stored in the job database. For example, job1 is submitted by user1, this job needs extra CPU. "Capacity" represents the required resource capacity (i.e. CPU). "Start time" represents the job start time and "Deadline" represents the deadline of this job. Meanwhile Resources send advertisements to DVM to update themselves. Figure 4.12 shows part of the resource database. The "Quality" implies the resource's QoS (i.e. time delay, reliability) and "Capacity" is the resources capacity (i.e. the speed of the CPU). "Price" represents the cost of using a certain resource. The DVM will try to guarantee that every job can meet its deadline in an efficient and cheap way.



Job	Applicant	Resource Requirement	Capacity	Start time	Deadline
Job1	User1	CPU	13	00:30	00:41
Job2	User2	CPU	14	00:32	00:38
Job3	User2	CPU	22	00:40	00:42
Job	……	……	……	……	……

Resource	Type	Quality	Capacity	Price
R1	CPU	1	40	20
R2	Memory	3	4000	40
R3	CPU	2	33	30
R…	…	…	…	…

Figure 4.12 How the DVM Makes VPN Policy

4.3.5 Inter-Domain VPN Support

A key aspect of this work is to enable dynamic MPLS VPNs to be supported between provider domains without divulging sensitive operator information. To demonstrate this principle an example will be considered, as shown in Figure 4.2. It is assumed that the user “A” has created a VPN including the special resource(s) located at “B”. This was achieved by the user contacting the DVM. The DVM then liaises with the connection management software at the ingress LSR point(s) to establish LSPs from A to B and from B to A. The path taken by the LSPs is not known to the DVM, however, it has the ability to specify connection QoS requirements. This allows Layer-2 resilience path switching to be carried out transparently to the DVM.

At some later time the user “C” wishes to join the VPN group. It starts this process by contacting its local DVM. Due to inter-domain DVM advertisements, C’s local DVM is aware that the target VPN exists and that it is managed by the DVM in AS1. It then uses inter-domain Network-Network Interface (NNI) signalling to negotiate the joining operation on behalf of C. If this request is permitted, the DVM in AS2 uses local connection management to establish LSP(s) from C to the relevant AS boundary router. The DVM in AS1 also sets up LSP(s) from its local VPN group member users and resources to its own AS boundary router. In both cases the TSpec (Traffic Specification) can be used to ensure QoS constraints are observed. So far, two incomplete MPLS tunnels have been created, each terminating at the local AS border router. The final stage is for the border routers to exchange labels to enable the splicing together of the tunnels. This can be done by piggy-backing MPLS labels on BGP routing messages, although other means are also possible. The crucial point is that the operator of AS2 has no knowledge of the LSP tunnel path taken in AS1. The complete LSP is not created in a true end-to-end fashion; rather it is formed from the concatenation of separate tunnels. The

CR-LDP or RSVP-TE signalling messages need not contain the complete source-routed path, only that portion of the path associated with a given segment.

As described above, As 2 is a “black box” to DVM1, to get service from AS2, DVM1 only submits its service application to DVM2, it will never see the detail information (such as routing information) inside AS2, which meets the security requirement of service provider. The current inter-domain routing protocol BGP could be the candidate of inter-AS label distribute protocol. Labels for the inter-domain LSPs might also be distributed directly by DVMs.

The complete process of an inter-domain LSP creation example is illustrated in Figure 4.13. User A and user B are two users in different ASes that wish to communicate via a dynamic LSP that is to link these two VPN members.

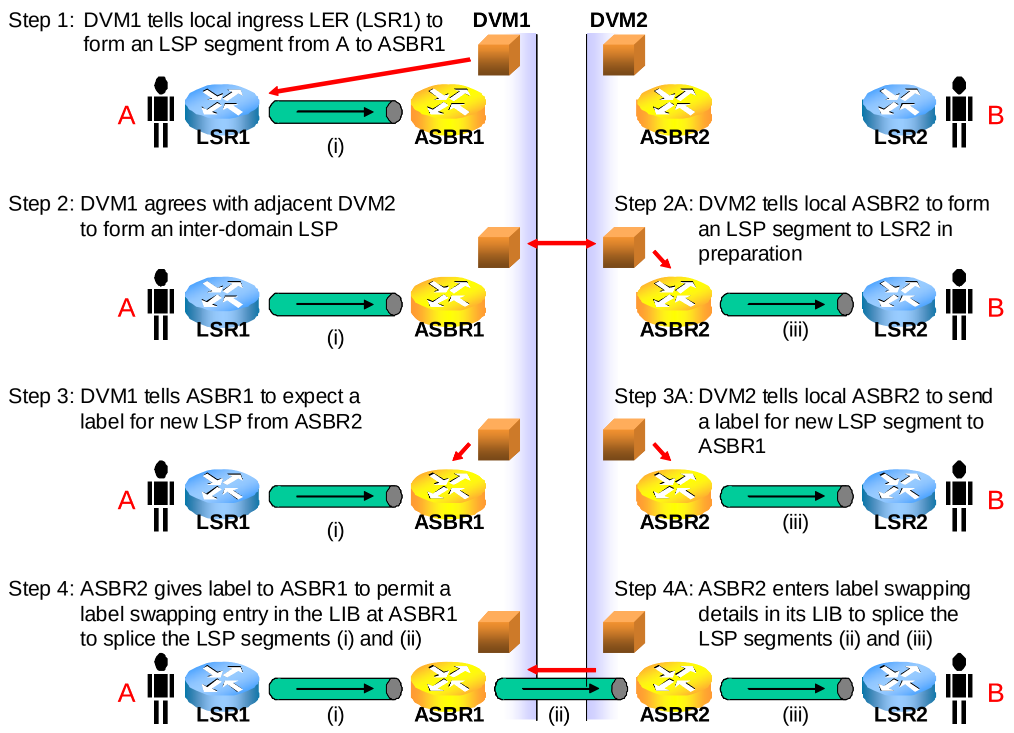

Step 1: DVM1 tells local ingress LER (LSR1) to form an LSP segment from A to ASBR1
DVM1
DVM2
A
LSR1
(i)
ASBR1
ASBR2
LSR2
B
Step 2: DVM1 agrees with adjacent DVM2 to form an inter-domain LSP
Step 2A: DVM2 tells local ASBR2 to form an LSP segment to LSR2 in preparation
(iii)
Step 3: DVM1 tells ASBR1 to expect a label for new LSP from ASBR2
Step 3A: DVM2 tells local ASBR2 to send a label for new LSP segment to ASBR1
Step 4: ASBR2 gives label to ASBR1 to permit a label swapping entry in the LIB at ASBR1 to splice the LSP segments (i) and (ii)
Step 4A: ASBR2 enters label swapping details in its LIB to splice the LSP segments (ii) and (iii)
(ii)

Figure 4.13 Inter-domain LSP Creation

4.3.6 Inter-domain Path Selection

All data packet transmissions across the AS border are via the ASBR routers (BGP router). When a packet arrives at the BGP router, if an inter-domain LSP has been established, a normal label swapping operation takes place. However, if the adjacent domain is unable or unwilling to accept continuation of the LSP, then the BGP router processes each packet at the conventional network layer, choosing the next hop as appropriate.

Assuming MPLS is employed across domains, the BGP router's Next Hop Label Forwarding Entry (NHLFE) table information must be set up with values learnt from the BGP router(s) in the neighbouring ASes. To some extent this is a selectable DVM operation, as there may be alternative means of "connecting" with remote VPN members. The DVM will try to select an optimal "inter-domain route" by considering Service Level Agreements (SLAs), cost and reliability factors.

In the scenario of Figure 4.14, a site in AS1 wants to build a VPN link to AS4. There are two choices, one is bypassing the AS2 which is an IP over MPLS network, and the other is bypassing the AS3 which is a pure IP network. When DVM1 submits its VPN link requirement to DVM2, DVM2 agrees to provide the service and QoS guarantee while DVM3 agrees to provide the link but does not support QoS guarantee. Considering the QoS requirement, DVM1 decides to select AS1-AS2-AS4 as the inter-domain route.

One the other hand, this DVPN architecture also provides the ability for automatic inter-domain route restoration. Assuming an AS path has been established from AS 1 to AS 4 bypass AS 4 in the scenario of Figure 4.14 and this path failed during the VPN lifetime. Under this condition, DVM1 will try to create a VPN link to AS4 bypass AS2 automatically. If AS2 agrees to provide this service after the negotiation between DVMs, a new VPN link will replace the failed link to provide service while the customers need not be aware of what happened.

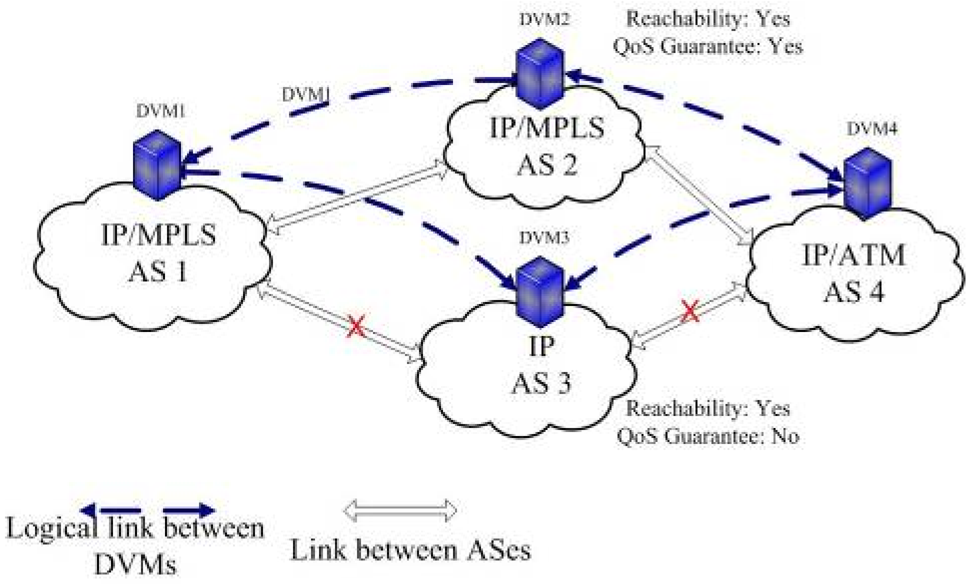


Figure 4.14 Inter-AS Routing Management

For scalability and security requirements of the SP, a DVM only has the detailed knowledge of its own AS. Other ASes are just "black boxes" as mentioned above, and the DVM can only see and talk with the DVMs in other ASes by NNI. It just submits its service requirements and other DVMs will decide whether or not to provide the service and return the information about the QoS and the price and reachability. A DVM does not care about and will never know the details of other domains. How to build a LSP and perform data transport in other ASes is not a concern of the DVM.

4.4 Dynamic VPN Processes

This chapter has described how the proposed dynamic VPN operates. Figure 4.15 illustrates a general scenario where the DVPN works across two domains.

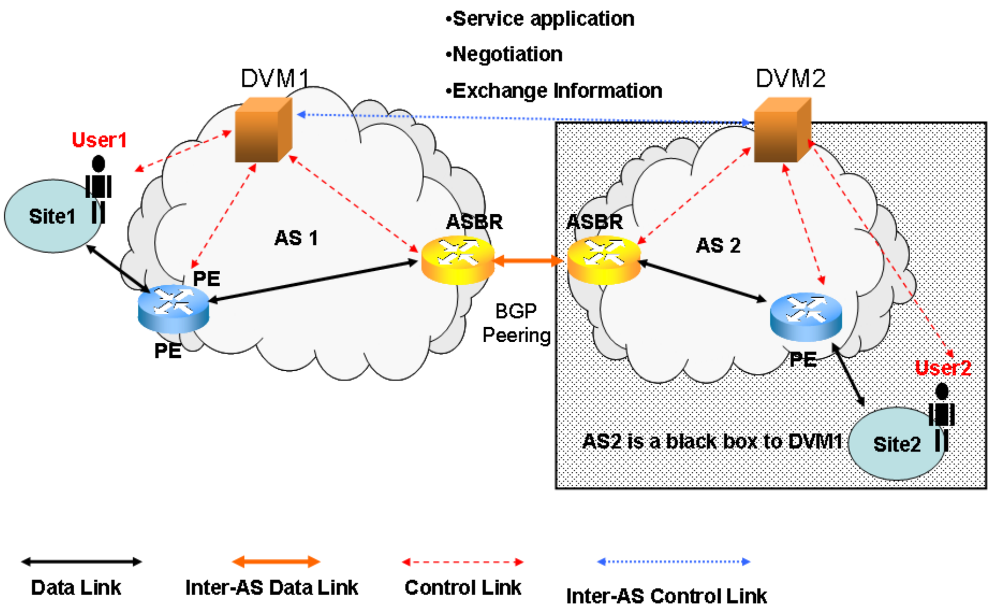


Figure 4.15 Example Multi-Domain DVPN Scenario

The users in this figure maybe human operators or software entities. If User1 in the site1 attached to PE1 in AS1 wants any service (such as building/removing a VPN link to user2 in the site2 attached to PE2 in AS2 or joining/leaving a VPN), it needs to submit a service application to DVM1 through the UNI (in the test-bed described in Chapter 5, the user submits its applications through a web page). A service application message will be transmitted to DVM1. The message format is represented in Figure 4.16:

user identity			Service request	
Name	IP	Authentication	service type	service attribute

Figure 4.16 The Service Application Message Format

By checking the user's identity information such as user name, IP address, and authentication against the information in the user information database, the DVM knows whether this user is eligible to submit this service application. The service request represents the service request details submitted by user. Service type represents what kind of service the user requires. The service attribute is a set of detailed requirements for this service. Different service types map to different service attributes. For example, if the service type is "to build VPN", the service attribute set is {SRC, DES, QoS}. SRC is the source address set of the VPN link and DES is the destination address set of this VPN link, QoS is the quality of service requirement set of this VPN link. In each message, the user can require to build more than one VPN links among several sources and destinations. The basic service types and the relative service attributes are listed in Table 4.1

Table 4.1 Service Types and Service Attributes

Service Type	Service Attribute
Build VPN	{SRC, DES, QoS
Join VPN	VPN ID
Leave VPN	VPN ID
Remove VPN	VPN ID

Receiving an appropriate application, DVM1 will undertake different actions in accordance with the service type and service attribute. In this example, user1 requires a VPN link to user2, DVM1 finds that user2 is in AS2 (by checking the address prefix of DES in the "VPN build" message from user1). It then sends a service application to DVM2 who is the controller of AS2. Receiving such an application, DVM2 also checks the applicant's identity firstly, then decides whether to provide this service or not. If DVM2 agrees to provide the VPN service to user2, it will send a message back to DVM1 saying "I agree to provide this VPN link to user2". DVM1 is a customer of DVM2 in this scenario and AS2 is a black box to DVM1. DVM1 does not know any detailed information of AS2, it only knows that DVM2 is the controller of AS2 and whether DVM2 can provide a service required or not. Then DVM1 builds a VPN link from user1 to BGP router A and DVM2 builds a VPN link from BGP router B to user 2. The LSP between ASBRs (A and B) is built according to the negotiation between DVM1 and DVM2 or some other inter-domain LSP build mechanism. Finally, the three segments are spliced together to form an inter-domain VPN from user1 to user2.

In the following paragraphs, the author will consider the communications among the DVMs, users, and underlying network devices (PEs and ASBRs) in detail. The four basic DVPN actions (build/remove, join/leave) and a network resource booking process will be employed to demonstrate the author's analysis. In this thesis, it is assumed that Transmission Control Protocol (TCP) is employed to guarantee the messages are redelivered successfully; issues of packets loss and re-delivery are considered outside the scope of this thesis.

4.5 VPN Basic Actions

4.5.1 VPN Building Action

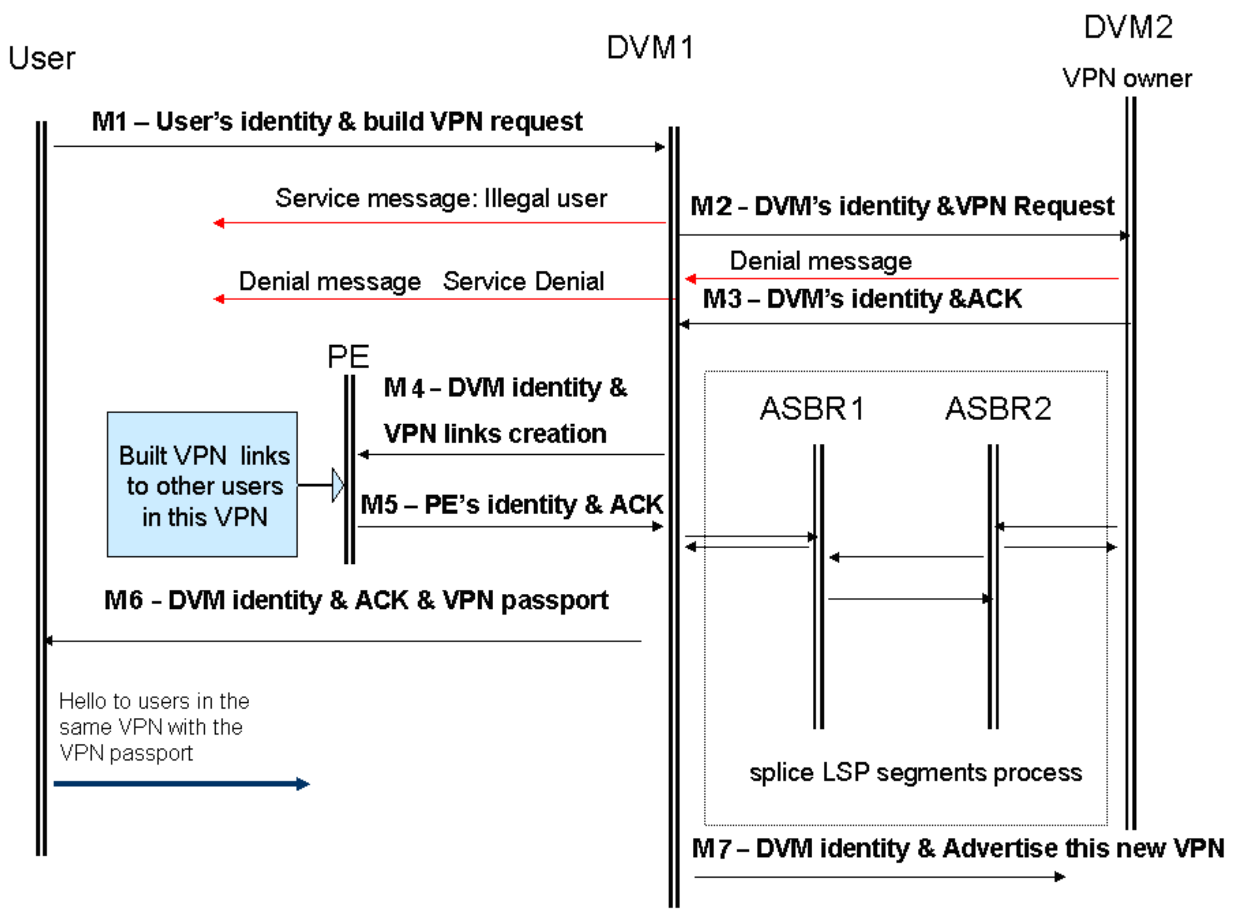

User
DVM1
DVM2
VPN owner
M1 – User's identity & build VPN request
Service message: Illegal user
M2 - DVM's identity &VPN Request
Denial message
Denial message Service Denial
M3 – DVM's identity &ACK
PE
M4 - DVM identity &
VPN links creation
ASBR1
ASBR2
Built VPN links to other users in this VPN
M5 – PE's identity & ACK
M6 - DVM identity & ACK & VPN passport
Hello to users in the same VPN with the VPN passport
splice LSP segments process
M7 – DVM identity & Advertise this new VPN

Figure 4.17 VPN Building Process

To setup a VPN service from the DVM, users need to send a service application message to the DVM in their own AS. The user's service application message format was described in section 4.4. The user identity and service request information is in this message. To set up a new VPN, the user sends a service application message (M1) with its identity and service type "set up a new VPN" to DVM1. After receiving this application, DVM1 checks the user's identity firstly. If it is not eligible, an "illegal user" message will be sent back to the applicant (user). If this user is eligible for this service, DVM1 will investigate whether it can provide such a service meeting the requirement represented by "service attribute". For example, the user may require certain network resources and QoS. If DVM1 concludes that such service requirements cannot be met, it will return a denial message to the user.

It is also possible that DVM1 needs to require the service from other DVM outside its own AS. For instance, a user requires a VPN link to another user located outside of its own AS. In this case, DVM1 needs to send a service request (M2) to the other DVM (DVM2 in Figure 4.17). The format of the service request message among DVMs is shown in Figure 4.18. They include the DVM's identity, the applicant's (user) identity and service request information in this message. If DVM2 agrees to provide this service to DVM1, it will send an ACK message (M3) to DVM1. Finally DVM1 determines that it can provide the user this inter-domain VPN service. If DVM2 refuses to provide such a service, it will return a denial message to DVM1. DVM1 will also send a denial message to the user after receiving the denial message from DVM2.

DVM identity			Applicant identity			Service request
ID	Address	Authentication	Name	IP	Authentication	service attribute

Figure 4.18 The DVM's Service Request Message Format

On the understanding that the user' service requirement can be met, The DVM1 will assign a VPN ID for this newly created VPN. The VPN ID is a global unique number represents the name of this VPN. As mentioned earlier, the VPN ID consists of the AS number (where the VPN is created originally) and the local VPN ID (only unique in local AS). The format of the VPN ID is shown in the Figure 4.19. For example, a VPN is built by DVM1 whose AS number is 0001 and this VPN is local numbered as 0035, so that the global VPNID of this VPN is 0001.0035.

AS number	Local VPN number

Figure 4.19 The VPN ID Format

DVM1 then sends a command (M4), ordering the PEs (to which the VPN source nodes i.e. users are attached) to build VPN links to the destinations. The format of this message is shown in Figure 4.20. Only a command message with a legal DVM identity can be accepted by the PE. "VPN command" describes this command in detail. "Command type" represents the kind of job the DVM ordered the PE to do. (i.e. 1 is to build VPN link and 2 is to remove VPN link). The command attribute is different according to different command types. For building a new VPN, there are the source and destination addresses, QoS requirements and VPN ID. The command types and command attributes are list in Table 4.2

DVM identity			VPN command	
ID	Address	Authentication	Command type	Command attribute

Figure 4.20 DVM Command Message Format

Table 4.2 The Command Types and Command Attributes

Command Type	Command Attribute
Build VPN	{SRC, DES, QoS}
Remove VPN	VPN ID

Receiving the “build VPN command”, (command type is 1), PE determines how to build the VPN link(s) and initiates the VPN link building process. To build a VPN link, the PE needs to do the following (which is described in RFC 2547 [5]):

1. Build a LSP from ingress to egress PE,
2. Create the VRF (Virtual Routing and Forwarding table),
3. Exchange routing information with CE and other PEs related to this VPN.

After completing these jobs, the PE sends an ACK (M5) back to the DVM, informs the DVM that the “job is done”. If the VPN link is an inter-domain one, the DVM also needs to communicate with the ASBR to splice the LSP segments as described in section 4.3.

After all of the above jobs are completed, the VPN is created. The DVM will also assign a user account for each user of this VPN. The information will be updated in the user database and VPN database of DVM. The passport is maintained by the DVM. The DVM is also responsible for advertising valid passports and withdrawing passports. However, the DVM does not manage the VPNs owned by other DVMs. For these remote VPNs, the DVM only keeps its DVPN ID in the VPN database and the DVPN’s owner DVM identity.

Finally, the DVM1 will send an ACK message (M6) to the applicant (user), saying its applicant is accepted and the VPN has been created successfully. The DVM also advertises this new VPN (M7) to other DVMs outside its own AS. Receiving the notification, the user can start the “hello” session to other users in the same VPN.

4.5.2 Join VPN Action

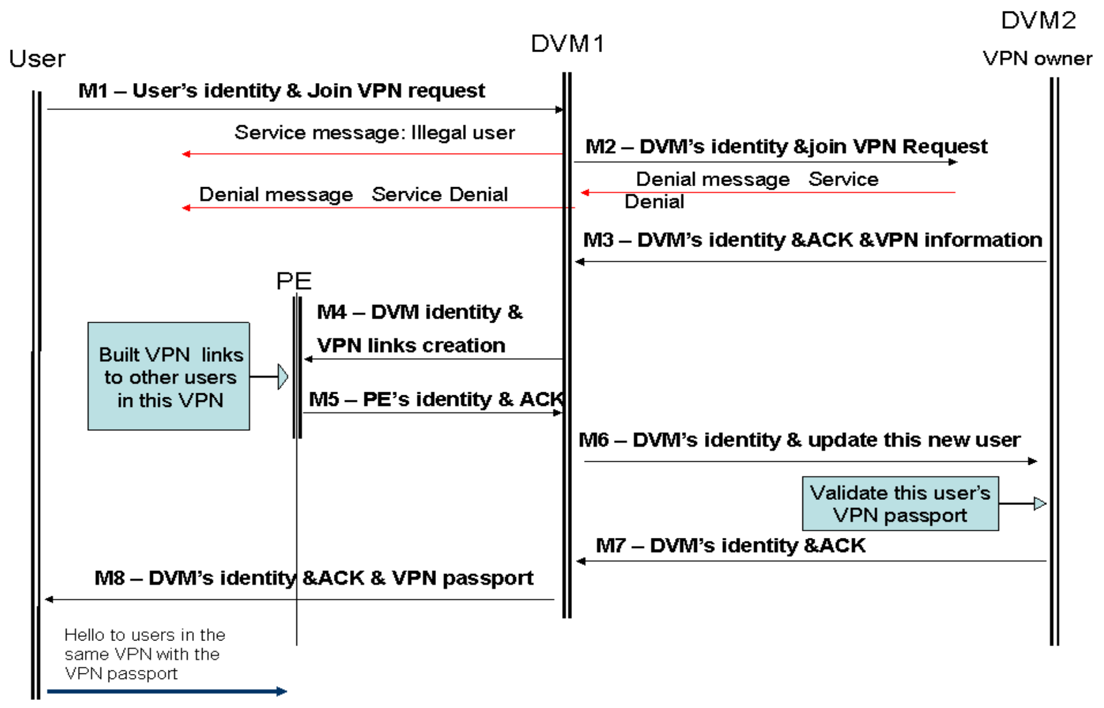

User
DVM1
DVM2
VPN owner
M1 – User's identity & Join VPN request
Service message: Illegal user
Denial message Service Denial
M2 – DVM's identity &join VPN Request
Denial message Service Denial
M3 – DVM's identity &ACK &VPN information
PE
M4 – DVM identity & VPN links creation
Built VPN links to other users in this VPN
M5 – PE's identity & ACK
M6 – DVM's identity & update this new user
Validate this user's VPN passport
M7 – DVM's identity &ACK
M8 – DVM's identity &ACK & VPN passport
Hello to users in the same VPN with the VPN passport

Figure 4.21 Join VPN Action

A user can also join an existing VPN. The join VPN action is described in the Figure 4.21. Before submitting the application, the user needs to know the VPN's ID. The user can get the VPN ID from the VPN administrator or through others means. To join a certain VPN, the user needs to send a "join VPN" request message (M1) to DVM. This message's format is the same as the "create VPN" request message and the "service type" is set to "Join VPN" and the service attribute is also set to the VPN ID of the VPN which the user wants to join into. After receiving this message, DVM1 will check the user identity first, if this is a legal user, DVM1 then investigates the VPNID to check if this is a remote VPN. If it is, DVM1 will send the join VPN request (M2) to the DVM who is the owner of this VPN. The DVM owning this VPN will return the ACK (M3) if it agreed for this user to join the VPN. If the VPN is a local one, the DVM1 will decide whether to permit the user join into this VPN.

If this application is accepted (locally or remotely), DVM1 will send a VPN build links command (M4) to the PE (the user attached). The necessary VPN links involving this new user are represented in this command. For example, there are already 2 users: user1 and user2 in this VPN, if user3 wants to join (a full mesh), two bi-direction VPN links user1-user3 and user2-user3 are necessary.

After the PE completes its job, it will send an ACK (M5) to DVM1, in the local case, DVM1 will update this new user in the user database and VPN database, in the remote case, DVM1 will send an update message (M6) to the owner DVM (DVM2), then DVM2 can update its user and VPN database, and return an ACK (M7) back to DVM1.

At last the DVM1 sends an ACK (M8) to the user who sends the “join VPN” request. Finally, this user can start to say hello to the other members in this VPN.

4.5.3 Leave VPN Action

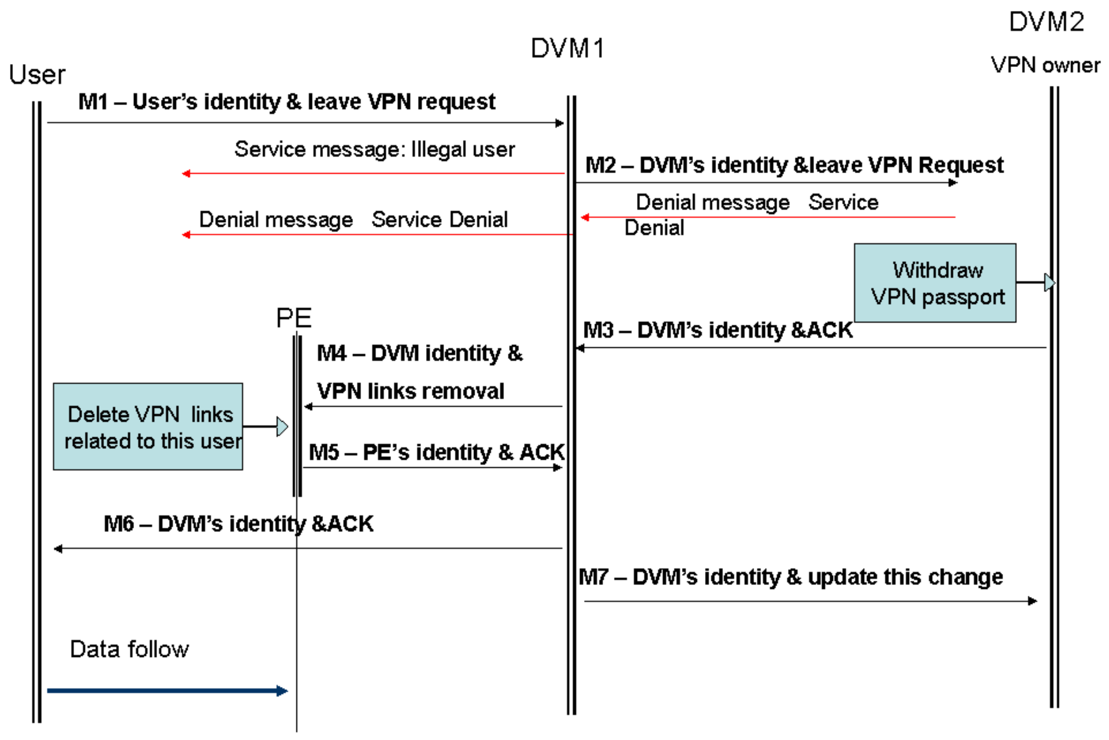

User
DVM1
DVM2
VPN owner
M1 – User's identity & leave VPN request
Service message: Illegal user
Denial message Service Denial
M2 – DVM's identity &leave VPN Request
Denial message Service Denial
Withdraw VPN passport
M3 – DVM's identity &ACK
PE
M4 – DVM identity & VPN links removal
Delete VPN links related to this user
M5 – PE's identity & ACK
M6 – DVM's identity &ACK
M7 – DVM's identity & update this change
Data follow

Figure 4.22 Leave VPN Action

A user can leave a VPN by submitting a “leave VPN” request. This process is described in the Figure 4.22. The user sends leave VPN request (M1) to its DVM (DVM1) first. The service type is set to “leave VPN” and the VPNID which the user wants to leave is represented in the service attribute. If this application is a legal one, DVM1 will look into the service type and service attribute. If this VPN is a remote one, DVM1 will send a leave VPN request message (M2) to the owner DVM (DVM2). There are user identity information and the VPNID in this message. For a legal request message, DVM2 will delete this user from the VPN database, and withdraw the VPN passport of this user. Then it sends an ACK (M3) to DVM1.

Receiving this message from DVM2, DVM1 will send a command (M4) to the PE, telling it to remove all the VPN’s links related to this user. PE will return an ACK (M5) back to DVM1 after completing these tasks. Finally, DVM1 sends an ACK (M6) to the user saying that “you have left this VPN successfully”.

4.5.4 Remove VPN Action

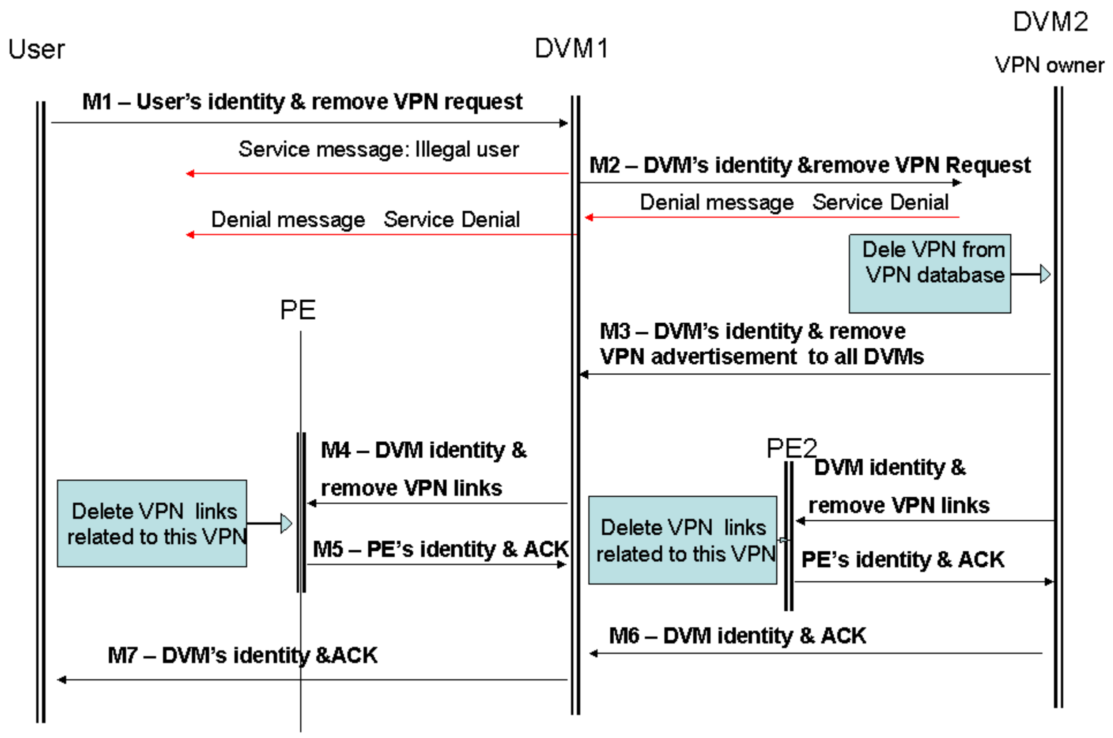

User
DVM1
DVM2
VPN owner
M1 – User's identity & remove VPN request
Service message: Illegal user
Denial message Service Denial
M2 – DVM's identity &remove VPN Request
Denial message Service Denial
Dele VPN from VPN database
PE
M3 – DVM's identity & remove VPN advertisement to all DVMs
M4 – DVM identity & remove VPN links
Delete VPN links related to this VPN
M5 – PE's identity & ACK
PE2
DVM identity & remove VPN links
Delete VPN links related to this VPN
PE's identity & ACK
M6 – DVM identity & ACK
M7 – DVM's identity &ACK

Figure 4.23 Remove VPN Action

A VPN can also be removed according to a user's request. The user needs to send a "remove VPN request" message (M1) to the DVM (DVM1). In this message, the service type is set to "remove VPN", and the DVPNID is included in the service attribute. DVM1 will check the user identity. If this is a legal request, the service type and attribute will be investigated by DVM1. According to VPNID, DVM1 can get the information whether this VPN is local or remote. For the remote one, DVM1 will send a request message (M2) to the owner of this VPN (DVM2). Receiving this message, DVM2 will check the DVM1's identity first. DVM2 also needs to check whether this user has the right to remove this VPN in the VPN database. If this user has the right, DVM2 will accept this request. It will delete this VPN from the VPN database, and advertise (M3) this removal of the VPN to other DVMs (including DVM1). Receiving such a message, the DVMs will investigate its user database, pick out all the users related to this VPN, and order (the VPNID is in this message) the PEs relating to these users to delete these VPN links of the VPN (M4) and receive ACKs (M5) from the PEs (assuming PEs operate properly). (DVM2 also does the same within its AS). Finally, DVM2 will send an ACK (M6) back to the DVM1 saying that this user's remove VPN request has been accepted and this VPN is removed. Consequently, DVM1 will send an ACK (M7) to the user when the VPN removal action finished.

4.5.5 Resource Booking Action

As described in the previous chapter, reserving network resources for user's applications is one of the most important functions of the DVM, which is also one of the key novelties of the DVPN architecture. Figure 4.24 describes an example of the resource booking process.

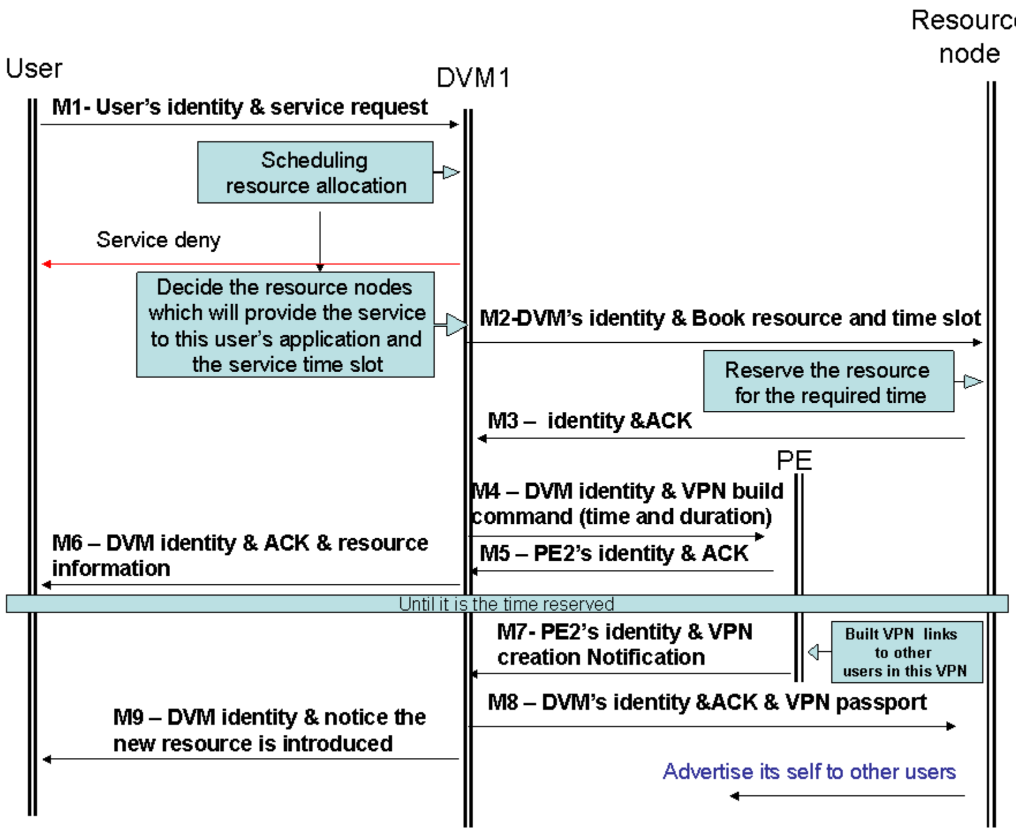

User
DVM1
Resource node
M1- User's identity & service request
Scheduling resource allocation
Service deny
Decide the resource nodes which will provide the service to this user's application and the service time slot
M2-DVM's identity & Book resource and time slot
Reserve the resource for the required time
M3 – identity &ACK
PE
M4 – DVM identity & VPN build command (time and duration)
M5 – PE2's identity & ACK
M6 – DVM identity & ACK & resource information
Until it is the time reserved
M7- PE2's identity & VPN creation Notification
Built VPN links to other users in this VPN
M8 – DVM's identity &ACK & VPN passport
M9 – DVM identity & notice the new resource is introduced
Advertise its self to other users

Figure 4.24 DVM Booking Network Resource for Users

In this scenario the resource manager is located at the resource node itself. The user sends a service request message (M1) to its local DVM. The user identity information and the service requirement details are in this message. Receiving this message, the DVM will check the user's identity first. If this is a legal message, the DVM will investigate the requirements in detail. The DVM calculates how to schedule the workflow and resource allocation according to the user's application and network conditions. If the DVM cannot find enough network resource to meet this user's requirement, it will return a "service deny" to the user. An illegal service application will also be denied; for example an application coming from a user with incorrect identity information, or a user has no right to receive this service (For example a user with insufficient privileges, or a user that has not paid the bill for previous DVPN service). If the DVM can find enough network resource to meet the user requirement, it will decide the resource allocation -which resource to be included into this user's VPN and the time the resource begins to serve in this VPN. After that, the DVM will send a "resource booking" message (M2) to the resource node with the DVM identity information. In the DVPN system, the resources outside the local AS can also be utilised. To book a resource from another AS, the DVM needs to send request to the DVM of the AS where the resources is located. This resource node checks the DVM identity first. If it is legal, the resource will be booked for that time and that VPN. Then an ACK message (M3) is returned to the DVM to confirm this booking. After DVM receives this ACK message, it knows that the resource is booked successfully. It then updates the resource database. At the same time, the DVM sends a "VPN build" message (M4) to the PE attached to that resource. In this message, the DVM includes the information about the VPN and the time for the resource to join the VPN and the time for it to leave. The PE will return an ACK message (M5) to DVM if it receives a legal command from the DVM. Receiving this ACK message, DVM will return the user an ACK (M6) to confirm its request is accepted and tell the user the resource booking information (such as available time, location, capacity).

When it comes to the reserved time, the PE attaches the resource to the VPN users by setting up links to all the other users in that VPN and sends a notice message to DVM (M7). Then DVM updates the related databases and sends a message (M8) including a valid VPN passport to the resource node. The DVM will also send a notice message (M9) to the user informing that this resource has been introduced into the VPN. Then the resource can start to serve in this VPN.

4.6 Orchestrated Computing

In this section, how the orchestrated computing works in the DVPN system is described along with all the DVM messages required to perform the various VPN activities.

4.6.1 Ticket Identifiers

A unique ticket is assigned (by the DVM) to each task to distinguish it from others. The format of a task's ticket is "DVM1id.userjobid.DVMjobid.taskid". The DVMid indicates which DVM assigned this ticket. The userJobid.DVMJobid consists of two parts and indicates which job the task belongs to. The user assigns the userJobid and the DVMJobid is assigned by the DVM to which the user submitted the job. For example a 10.12 implies that this job is the 10th job of this user and 12th job of this DVM. Taskid indicates which task of the job this task is. For example a ticket 1.10.12.23 represents task number 23 of job 10.12 submitted to DVM1.

4.6.2 DVPN Message Types

The messages that carry the DVPN control information are defined as follows:

1. **Service request message** (Users to the DVM)

   A user sends a "service request" message to the DVM to request service. The user's identity and job information are included in this message.

2. **Service accepted message**: (DVM to the user)

If the DVM can provide the service requested by the user, the DVM will return a message defined as "service accepted" message back to the user. Included in this message a unique task identifier, known as a ticket, is provided for each of the tasks that comprise the job submitted by the user.

3. **VPN link set up message**: (DVM to CM)

To set up a VPN link, the DVM sends the "VPN link set up" message to the CM. The source and destination address of the VPN link are included in this message. The VPN QoS requirements are also included in this message.

4. **VPN link set up confirm message**: (CM to DVM)

The CM always returns a confirm message after the VPN link request by the DVM is set up, defined as "VPN link set up confirm" message. DVM will assign a VPN ID for this new VPN link after receiving this message.

5. **VPN link remove message**: (DVM to CM)

To remove an exiting VPN link, the DVM sends this message to tell the CM to remove a VPN link. The VPN ID is included in this message to identify the VPN link.

6. **VPN link remove confirm message**: (CM to DVM)

After the VPN link is removed, the CM sends the DVM a message to confirm that this VPN link has been removed. The VPN ID is included in this message to identify the VPN link.

7. **Inter-domain VPN link setup message**: (DVM to DVM)

To use an inter-domain message, the DVM sends an "inter-domain VPN link setup" message to the DVM of the remote AS. The destination of this inter-domain VPN link and the QoS requirements are included in this message.

8. **Inter-domain VPN link confirm message**: (DVM to DVM)

If a DVM can provide the inter-domain VPN link service to another AS, this DVM will return an inter-domain VPN link confirm message. The label for the inter-domain LSP is included in this message.

9. **Inter-domain VPN service request message**: (DVM to DVM)

To use an inter-domain service, the DVM sends an "inter-domain VPN service request" message to the DVM of another AS. The destination of this inter-domain VPN link and the QoS requirements are included in this message.

10. **Inter-domain label request message**: (DVM to CM)

To provide an inter-domain VPN link service to another AS, the DVM needs to request a label for the inter-domain LSP. To request this label, the DVM sends an “inter-domain label request” message to the CM.

11. **Inter-domain label reply message**: (CM to DVM)
The CM returns an “inter-domain label reply” message including the label for the inter-domain LSP to the DVM; this message is sent in response to an “inter-domain label request” message.

12. **Inter-domain LSP setup message**: (DVM to CM)
To create the inter-domain LSP, the DVM sends an “inter-domain LSP setup” message to the CM to create the inter-domain LSP. The message contains the label received from the DVM of the destination AS, which is the label for this inter-domain LSP (provided by the DVM of the destination AS).

13. **Inter-domain LSP setup confirm message**: (CM to DVM)
The CM returns a confirm message to the DVM after the inter-domain LSP is created. The VPN ID is included in this message to identify the VPN link.

14. **Inter-domain VPN link remove message**: (DVM to DVM)
The DVM sends an “inter-domain VPN link remove” message to the DVM providing the inter-domain VPN link service. The VPN ID is included in this message to identify the VPN link.

15. **Inter-domain VPN link remove confirm message**: (the DVM to the DVM)
The DVM returns this message after the inter-domain VPN link is removed. The VPN ID is included in this message to identify the VPN link.

16. **Inter-domain LSP remove message**: (DVM to CM)
DVM sends a message to CM to remove the inter-domain LSP for the inter-domain VPN link. The VPN ID is included in this message to identify the VPN link.

17. **Inter-domain LSP remove confirm message**: (CM to DVM)
The CM returns an “inter-domain LSP remove confirm” message after the inter-domain LSP is removed. The VPN ID is included in this message to identify the VPN link.

18. **Inter-domain label release message**: (DVM to CM)
After the DVM revive the “inter-domain VPN link remove” message, it sends a message to the CM to release the label assigned for the inter-domain LSP, the VPN ID is included in this message to identify the

VPN link.. The CM will order the ASBR to release this label and the resources after receives this message. This message is defined as "Inter-domain Label release message".

19. **Resource booking message**: (DVM to Resource)
To book a resource to perform a task, the DVM sends a message defined as the "resource book" message to the resource (or RM) located within its own domain. The task's ticket is included in this message.

20. **Resource booking confirm message**: (Resource to DVM)
The Resource returns a "resource book confirm" message to the DVM to confirm with the DVM that the resource has been booked for this task. The confirmation includes the ticket number.

21. **Inter-domain resource booking message**: (DVM to DVM)
In the DVM need to book a resource located in another AS, it will a send an "inter-domain resource booking" message to the DVM of the AS where the resource is located.

22. **Inter-domain resource booking confirm message**: (DVM to DVM)
The DVM will return an "inter-domain resource booking" message if it successfully books the resource acquired by the DVM of the remote AS.

23. **Data transfer message**: (DVM to Data Supplier i.e. User/Resource)
After the VPN link is set up from the data source (user or resource) to the destination (user or resource), the DVM sends a message to the data resource to notify the resource to send the necessary data to the destination. The destination address is provided in this message. The task ticket, which this data is for, is also included in this message. This message is defined as "data transfer" message.

24. **Data transfer complete confirm message**: (Data Supplier to DVM)
After the data transfer completes, the data resource (sender) returns a "data transfer complete confirm" message back to DVM. The task's ticket is also included in this message.

25. **Inter-domain data transfer message**: (DVM to DVM)
In the case when a resource[3] or user in a remote domain is ready to send some data to a user or resource in a local domain, the remote DVM informs the local DVM of this situation. At this time the local DVM then issues this "inter-domain data transfer" message to the remote DVM to tell it where the data should be delivered to.

26. **Inter-domain data transfer complete message**: (DVM to DVM)

[3] i.e. when a task has finished being processed

After the DVM[4] that hosts the destination of the data transfer is informed that the transfer has finished, this local DVM must inform the remote DVM, where the data originated, that the transfer has completed satisfactorily.

27. **Task completed message**: (Resource to DVM)
After a task completes, the resource sends a "task completed" message to the DVM to report that the task is finished. The task's ticket is also included in this message.

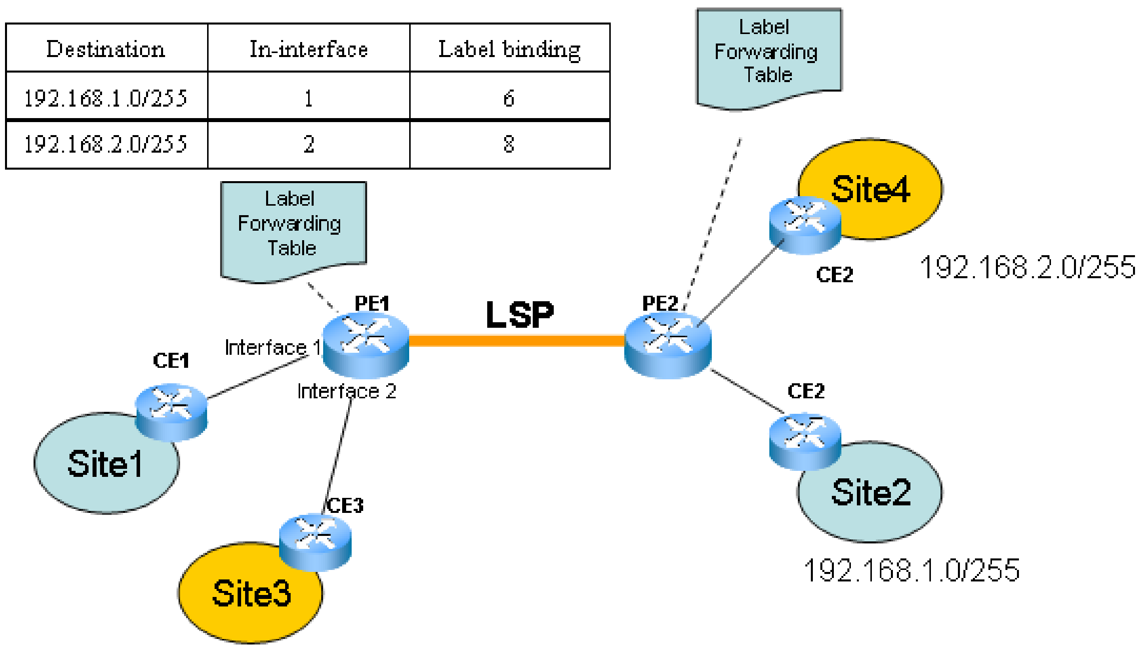
28. **Inter-domain Task completed message**: (DVM to DVM)
If the "task completed" message received is for an inter-domain task, the DVM will send an "inter-domain task complete message" to the DVM of the AS where this task originated.

4.6.3 VPN Link Setup

Orchestrated computing requires that the VPN links can setup dynamically. In this thesis, the VPN link setup process consists of the LSP creation and updating of the VPN Label Forwarding Table. Figure 4.25 represents how a VPN link from site1 to site2 is setup. In this example, two actions are carried out. The first is the setting up of the LSP from PE1 to PE2; the other one is the setting up the Label Forwarding Table entry at PE1 (The egress PE (i.e. PE2) always pops the labels). Assume that the address of site2 is 192.168.1.0/255. PE1's interface to CE1 is 1. Label 6 is assigned for this VPN link. The Label Forwarding Table will look like as Figure 1. It implies that all the IP packets (with destination of 192.168.1.0/255) coming into the interface 1 (in PE2) will be en-capsulated into MPLS packets with label 6 and sent to PE2 along the LSP. The LSP might be an intra-domain or inter-domain LSP.

It is also possible that VPNs share a common a LSP. However, each VPN will be assigned an individual label at the ingress PE and traffic of different VPNs will be encapsulated with different labels. Label stacking (to a label depth of two) is employed; the label assigned to VPN is the bottom label while the LSP's label is on the top of the label stack. In this example, the VPN from site1 to site2 is assigned a label 6 and the VPN from site3 to site4 is assigned label 8.

[4] Referred to as the local DVM



Destination	In-interface	Label binding
192.168.1.0/255	1	6
192.168.2.0/255	2	8

Figure 4.25 The VPN link Setup

Moreover, it is also possible that a user belongs to multiple VPNs at the same time. The traffic are separated according to the different destinations and membership and assigned with different labels. For example, as shown in Table 4.3, from the same user the VPN to the 192.168.1.0/255 is assigned label 6 and the VPN to 192.168.2.0/255 is assigned label 12.

Table 4.3 VPN label forwarding table example

Destination	Ingress Interface	VPN Label
192.168.1.0/255	1	6
192.168.2.0/255	2	12

4.6.4 User Task Allocation to Resources

In the following paragraphs, the mechanism by which resources are accessed by the users is illustrated, showing the sequence of actions that are undertaken. Two situations are considered. In the first instance, the resources and the user exist within the same domain. In the second example, the message interactions that arise when the resources and user are in separate domains are described. This includes the messages that are required to setup an inter-domain LSP and release it when the task processing has completed.

A) Intra-domain Scenario

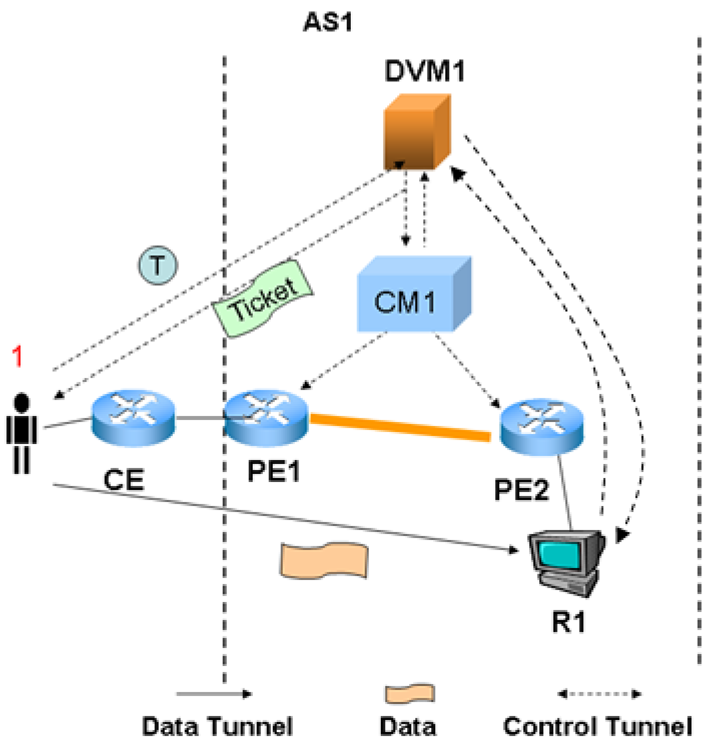


Figure 4.26 Intra-domain

1. User1 sends a service request message to DVM1, the detailed job information is included
2. DVM1 accepts this application and returns user1 a service accepted message including a set of tickets assigned to the tasks to distinguish the tasks from others.
3. DVM1 sends resource book message to R1 to book the resource for task1. Also, the ticket of task1 is attached in this message.
4. R1 returns the resource book confirm messages to DVM1
5. DVM1 sends VPN setup message to CM1 to order CM1 to create a VPN link from User1's R1.
6. CM1 returns DVM1 a VPN link setup confirm message after the VPN link is created
7. DVM1 sends a data transfer message to user1 to tell user1 to transfer data for task1 to R1
8. User1 returns a data transfer complete message back to DVM1 after the data transaction completes

B) Inter-domain scenario

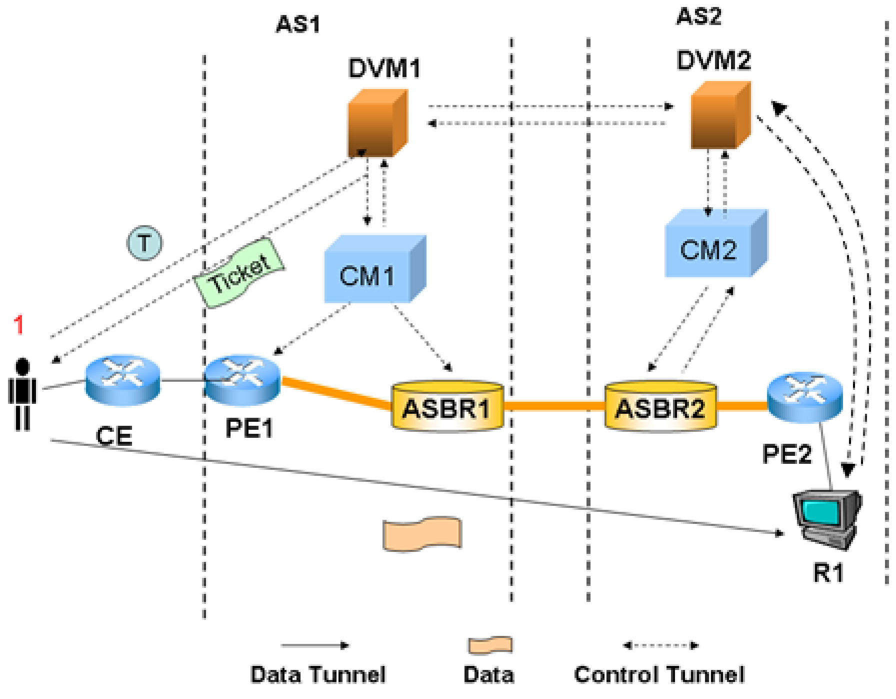
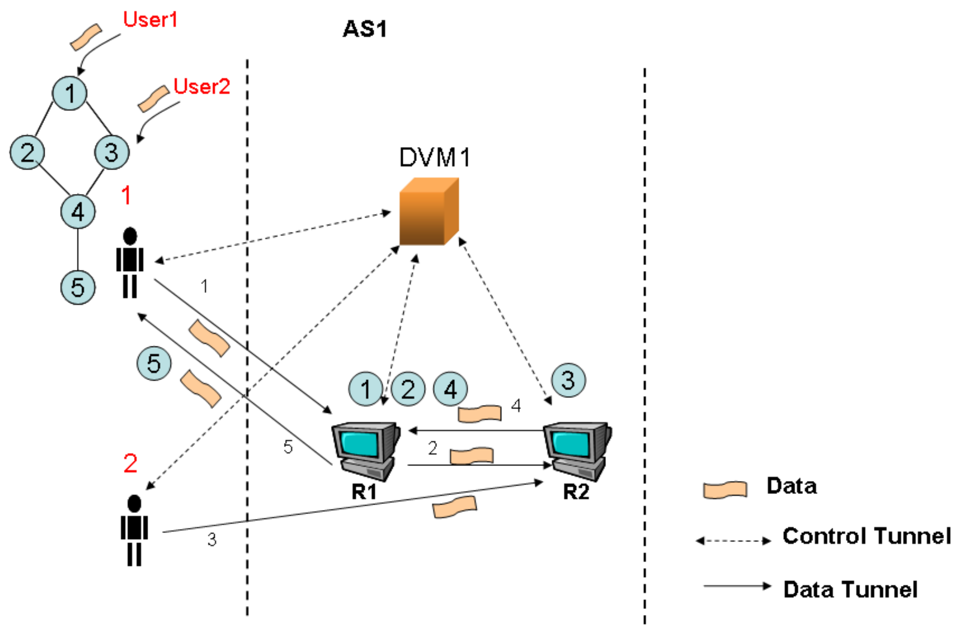


Figure 4.27 Inter-domain

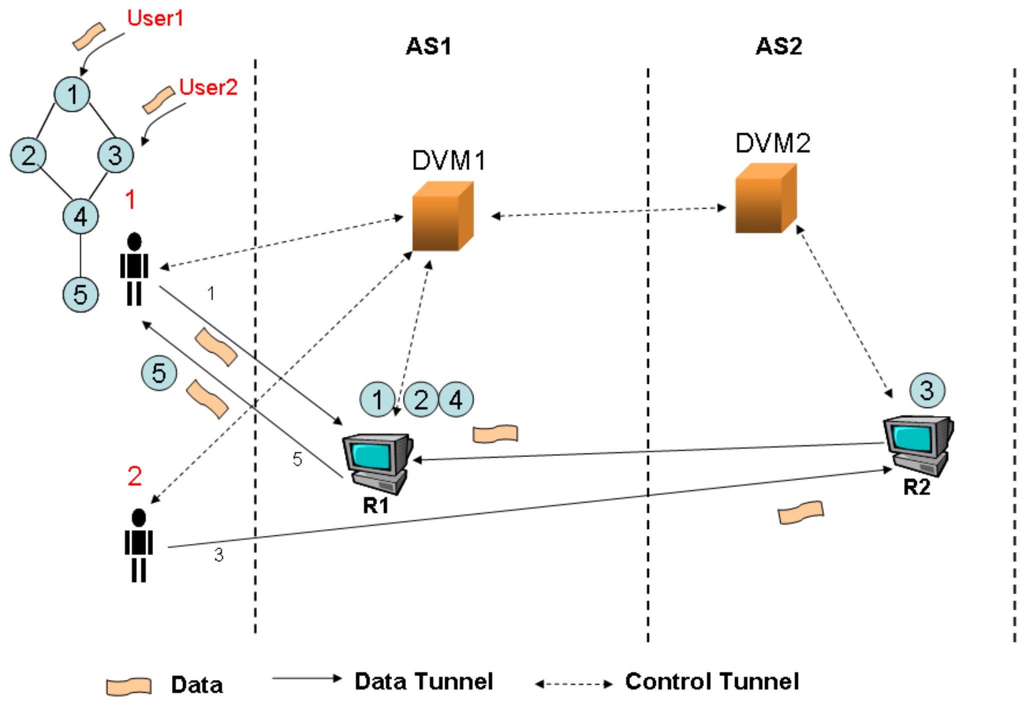
1) User1 submits service request to DVM1
2) DVM1 decides to allocate the task at R1 in AS2
3) DVM1 sends an inter-domain resource book message to DVM2 to book resource from R2 for task3
4) DVM2 sends a resource booking message to R2 to book the resource for task3
5) R2 returns a resource booking confirm message to DVM2
6) DVM2 returns an inter-domain resource booking confirm message to DVM1
7) DVM1 sends an inter-domain VPN link request (to R2) message to DVM2
8) DVM2 sends a VPN link setup message to CM2 to create a VPN link from ASBR2 (attached to ASBR1 of AS1) to R2
9) DVM2 sends an inter-domain label request message to CM2 to request a label for the inter-domain LSP
10) CM2 returns an inter-domain label reply message (including the label for the inter-domain LSP) to DVM2
11) DVM2 returns an inter-domain VPN link confirm message including this label to DVM1
12) DVM1 sends an inter-domain LSP setup message to CM1 to create LSP to ASBR2 with this label
13) CM1 returns an inter-domain LSP setup confirm message
14) DVM1 sends a VPN link setup message to CM1 to create VPN link from user2 to ASBR1
15) CM1 returns a VPN link setup confirm message to DVM1
16) DVM1 sends a data transfer message to user2 to send the necessary data to R2
17) User2 returns a data transfer complete message back to DVM1 after the transaction completes

4.6.5 Scenario Illustration

To analysis the control messages of a typical orchestrated computing in the DVPN system, the example of Figure 4.28 is considered. The intra-domain scenario is firstly analysed by employing Figure 4.28 (a), then the inter-domain scenario is analysed by employing Figure 4.28 (b). For simplicity, all the SP's underlying equipment (CM, PE, CE entities) are not shown in Figure 4.28.



(a) Intra-Domain



(b) Inter-Domain

Figure 4.28 DAG Processing in the DVPN System

In the analysis, it is assumed that all the service requests are legal and will not be denied. The DVM control messages for processing the job of user1 are listed step by step as follows:

1. User1 sends a service request message to DVM1, the detailed job information is included

2. DVM1 accepts this application and return user1 a service accepted message including a set of tickets assigned to the tasks to distinguish them from others.
3. DVM1 sends resource-booking message to R1 to book the resource for task1. Also, the ticket of task1 is attached in this message.
4. R1 returns the resource booking confirm messages to DVM1
5. DVM1 sends VPN setup message to the CM to order the CM to create a VPN link from User1 to R1.
6. The CM returns DVM1 a VPN link setup confirm message after the VPN link is created
7. DVM1 sends a data transfer message to user1 to tell user1 to transfer data for task1 to R1
8. User1 returns a data transfer complete message back to DVM1 after the data transaction completes
9. DVM1 sends a VPN link remove message to the CM to remove the VPN link between user1 and R1
10. The CM returns the VPN link remove confirm message after the VPN link is removed.
11. R1 sends the DVM1 a task complete message to report that task1 has completed
12. DVM1 sends a resource booking message to R1 to book the resource for task2
13. R1 returns a resource booking confirm message back to DVM1
14. DVM1 sends a resource-booking message to R2 to book the resource for task3.
15. R2 sends back a resource booking confirm message back to DVM1
16. DVM1 sends a VPN link setup message to the CM to create VPN link from User2 to R2
17. A VPN link setup confirm message is returned by the CM after the VPN is setup
18. DVM1 sends a data transfer message to user2 to instruct user2 to send the necessary data to R2
19. User2 returns a data transfer complete message back to DVM1 after the transaction completes
20. DVM1 sends a VPN link remove message to the CM to remove the VPN link from the User2 to R2.
21. The CM returns VPN link remove confirm message back to DVM1 after the VPN link is removed
22. R1 sends a task complete message to DVM1 to inform it that task2 has finished
23. R2 sends a task complete message to DVM1 to inform it that tasks3 has finished
24. DVM1 sends a resource booking message to R1 to book the resource for task4
25. R1 returns back a resource booking confirm message to DVM1
26. The DVM1 sends the message to the CM to build VPN link from R2 to R1 to transfer the result data of task3 from R2 to R1
27. The CM sends a VPN link setup confirm message back to DVM1 after the VPN link is created
28. DVM1 sends a data transfer message to R2, tell it to send the result of task4 to R1
29. R2 returns a data transfer complete message to DVM1 after the data transaction completed
30. DVM1 sends a VPN link remove message to the CM to remove the VPN link from R2 to R1 for the data transaction
31. The CM returns a VPN link remove confirm message after the VPN link is removed
32. R1 returns a task complete message to DVM1 after task4 is completed

33. DVM1 sends a VPN link setup message to the CM to create a VPN link from R1 to user 1 for final data transaction (task5)
34. The CM returns a VPN link setup confirm message after this VPN link is created
35. DVM1 sends a data transfer message to R1 to transfer the result data to user1
36. R1 sends a data transfer complete message to DVM1 after the data transaction is completed
37. DVM1 sends a VPN link remove message to the CM to remove the VPN link from R1 to User1
38. The CM returns a VPN link confirm message back to DVM1 after the VPN link is removed

The inter-domain is also examined considering the scenario shown in Figure 4.28 (b), where R2 is within AS2. DVM1 knows the existence of R2 since DVMs advertise the availability of resources to other DVMs. The CM in AS1 is named CM1 and CM in AS1 is named CM2. The ASBR in AS1 is ASBR1 and ASBR2 is used to represent the ASBR in AS2. It is assumed that ASBR1 is attached to ASBR2. All the PE routers, ASBRs and CMs are not shown in Figure 4.28 (b). The DVM control messages for processing the job of user1 in the inter-domain scenario are list step by step as follows:

1. User1 submits a service request to DVM1; the detailed job information is included
2. DVM1 accepts this application and returns user1 a service accepted message including a set of tickets assigned to the tasks of this job
3. DVM1 sends a resource booking message to R1 to book the resource for task1
4. R1 returns the resource booking confirm messages to DVM1
5. DVM1 sends a VPN link setup message to the CM1 to tell CM1 to create VPN link from user1 to R1
6. CM1 returns DVM1 a VPN link setup confirm message after the VPN link is created
7. DVM1 sends a data transfer message to user1 to transfer the necessary data to R1
8. User1 returns a data transfer complete message back to DVM1 after the data transaction is finished
9. DVM1 sends a VPN link remove message to CM1 to remove the VPN link between user1 and R1
10. CM1 returns the confirm message after the LSP is removed.
11. R1 sends a task complete message to report to DVM1 that task1 has completed
12. DVM1 sends a resource booking message to R1 to book the resource for task2
13. R1 returns a resource booking confirm message back to DVM1
14. DVM1 sends an inter-domain resource booking message to DVM2 to book resource from R2 for task3
15. DVM2 sends a resource booking message to R2 to book the resource for task3
16. R2 returns a resource booking confirm message to DVM2
17. DVM2 returns an inter-domain resource booking confirm message to DVM1
18. DVM1 sends an inter-domain VPN link request (to R2) message to DVM2
19. DVM2 sends a VPN link setup message to CM2 to create a VPN link from ASBR2 (attached to ASBR1 of AS1) to R2
20. DVM2 sends an inter-domain label request message to CM2 to request a label for the inter-domain LSP

21. CM2 returns an inter-domain label reply message (including the label for the inter-domain LSP) to DVM2
22. DVM2 returns an inter-domain VPN link confirm message including this label to DVM1
23. DVM1 sends an inter-domain LSP setup message to CM1 to create LSP to ASBR2 with this label
24. CM1 returns an inter-domain LSP setup confirm message
25. DVM1 sends a VPN link setup message to CM1 to create VPN link from user2 to ASBR1
26. CM1 returns a VPN link setup confirm message to DVM1
27. DVM1 sends a data transfer message to user2 to send the necessary data to R2
28. User2 returns a data transfer complete message back to DVM1 after the transaction completes
29. DVM1 sends a VPN link remove message to CM1 to remove the VPN link from the user2 to ASBR1
30. CM1 returns a VPN link remove confirm message after the VPN link is removed
31. DVM1 sends an inter-domain LSP remove message to CM1 to remove the inter-domain LSP
32. CM1 returns an inter-domain LSP remove confirm message
33. DVM1 sends an inter-domain VPN link remove message to DVM2 to remove the inter-domain VPN link to R2
34. DVM2 sends an VPN link remove message to CM2 to remove the VPN link from ASBR2 to R2
35. CM2 returns a VPN link remove confirm message to DVM2
36. DVM2 sends an inter-domain label release message to CM2 to withdraw the label for the inter-domain LSP
37. CM2 returns an inter-domain label release confirm message
38. DVM2 returns an inter-domain VPN link remove confirm message to DVM1
39. R1 sends a task complete message to DVM1 to inform it that task2 has finished
40. R2 sends a task complete message to DVM2 to inform it that tasks3 has finished
41. DVM2 sends an inter-domain task complete message to DVM1 to inform it that task3 has finished
42. DVM1 sends a resource booking message to R1 to book the resource for task4; attached is the ticket for task4
43. R1 returns a resource booking confirm message back to DVM1
44. DVM1 sends an inter-domain VPN link setup messages to DVM2 to create inter-domain VPN link from R2 to R1
45. DVM2 sends the VPN link setup message to CM2 to create the VPN link from the R2 to ASBR2
46. CM2 returns a confirm message after the VPN link is created
47. DVM1 sends an inter-domain label request message to CM1 to request a label for the inter-domain LSP
48. CM1 return an inter-domain label reply message including the label
49. DVM1 sends an inter-domain VPN setup confirm message with this label to DVM2
50. DVM2 send a inter-domain LSP setup message to CM2 with this label
51. CM2 returns a confirm message
52. DVM1 sends message to CM1 to create VPN link from ASBR1 to the PE

53. CM1 returns a VPN link setup confirm message
54. DVM1 sends an inter-domain data transfer message to DVM2 to ask R2 to transfer data from task3 to R1
55. DVM2 sends a data transfer message to R2 to transfer data of task3 to R1
56. R2 sends a data transfer complete message to DVM2 after the transaction has finished
57. DVM2 sends an inter-domain data transfer complete message to DVM1 to inform it that the transaction has finished
58. DVM1 sends an inter-domain VPN link remove message to DVM2 to remove the inter-domain VPN link from R2 to R1
59. DVM2 sends a VPN link remove message to CM2 to remove the VPN link from R2 to ASBR2
60. CM2 returns a VPN link remove confirm message
61. DVM2 sends an inter-domain LSP message to ASBR2 to remove the inter-domain LSP
62. ASBR2 returns an inter-domain LSP remove confirm message
63. DVM2 sends an inter-domain VPN link remove message to DVM1 to confirm that the VPN link has been removed
64. DVM1 sends a VPN link remove message to CM1 to remove VPN link from ASBR1 to user1
65. CM1 returns a VPN link remove confirm message
66. DVM1 sends an inter-domain label release message to CM1 to withdraw the label for the inter-domain LSP
67. R1 returns a task complete message to DVM1 after task4 has completed
68. DVM1 sends a VPN link setup message to CM1 to create a VPN link from R1 to user1 for the final data transaction (task5)
69. CM1 returns a VPN link setup confirm message after this VPN link is created
70. DVM1 sends a data transfer message to R1
71. R1 sends a data transfer complete message to DVM1 after the data transaction has completed
72. DVM1 sends a VPN link remove message to CM1 to remove the VPN link from R1 to User1
73. CM1 returns a VPN link remove confirm message back to DVM1 after the VPN link is removed

4.7 Separation of VPN Control Planes

The proceeding VPN message passing figures also imply the separation of the VPN control plane from the data plane. Shown in Figure 4.29, in this DVPN architecture, the communication between the user and the DVM, the communication between the DVM and its local underlying devices and the communication between DVMs are separated. The user requesting a VPN service only needs to contact its local DVM. From the user's point, it submits an application, receives a denial or an ACK, but it doesn't know about the communications between two DVMs and between the DVM and the PEs. Users never communicate (signalling) with a PE directly. There is no message exchange between the PE and the users. At the same time, there is also no message exchange between a DVM with any network devices of other ASes. DVMs only communicate with other DVMs. Indeed, a DVM can talk with a user outside its own AS directly, but it will never see any PEs or other backbone devices outside of its own AS. This approach satisfies the SP's security requirement: keep its network confidential so the user cannot attack the SP's backbone through DVPN signalling.

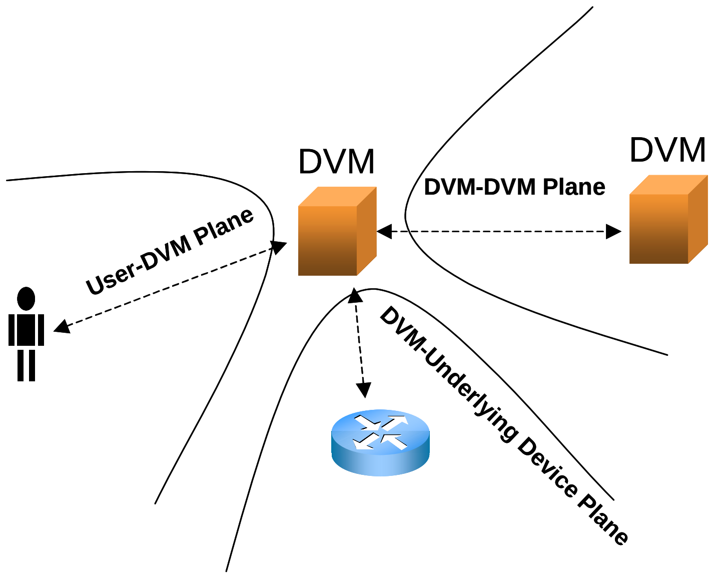


Figure 4.29 The DVPN Control Planes

4.8 Motivating Example Revisited: DVPN-Enabled Global Grid Computing

Referring back to the example shown in Figure 4.1 from section 4.1, we can now consider it from the perspective of the DVPN architecture.

At the beginning, user A sends a service application to the DVM1. The job information is included in the application message. Considering the destination D is in the AS2, DVM1 sends a request to DVM2 for a inter-domain VPN service from A to D. After receiving this request from DVM1, DVM2 sends DVM1 a message to confirm that this service is accepted. Then the MPLS VPN links between user A and user B are created via the process described in the previous section 4.3.5.
In the current scenario, resource B is located in AS1 and resource C is located in AS2. Because each DVM advertises the available resource information of its AS to others from time to time, both B and C are aware of DVM1. Based on the user A's application and the current resource availability, DVM1 decides to involve B and also the extranet resource C into this VPN to provide more powerful processing capacity for the User A's job. Then DVM1 sends the request to DVM2 for this resource. If DVM2 agrees to provide this service, resource C will also be involved in this grid computing VPN. The process of involving a resource into a VPN is similar as involving a user. In this example, to fulfil the job as fast as possible, T3 is sent to resource C while T2 and T4 are processed in resource B. These processes are illustrated in Figure 4.30.

After the job finishes, all these network resources (resources B, C) and VPN links will be released. During the VPN lifetime, resources might be involved or released according to real time status changes. It is also possible that VPN members join or leave the VPN dynamically. In the DVPN system, VPN members (resources and users) can be located in different ASes; the inter-domain VPN link might be created/released according to the negotiations among DVMs. The inter-domain QoS can also be guaranteed through the DVMs' negotiations. At the same time, each VPN only manages the network entities of it own AS. Any service regarding the extranet entities is achieved through requesting service from the DVMs in other ASes. In the above example, DVM1 creates the inter-provider VPN link through the negotiating with DVM2, and it never signals with any other entities in AS2.

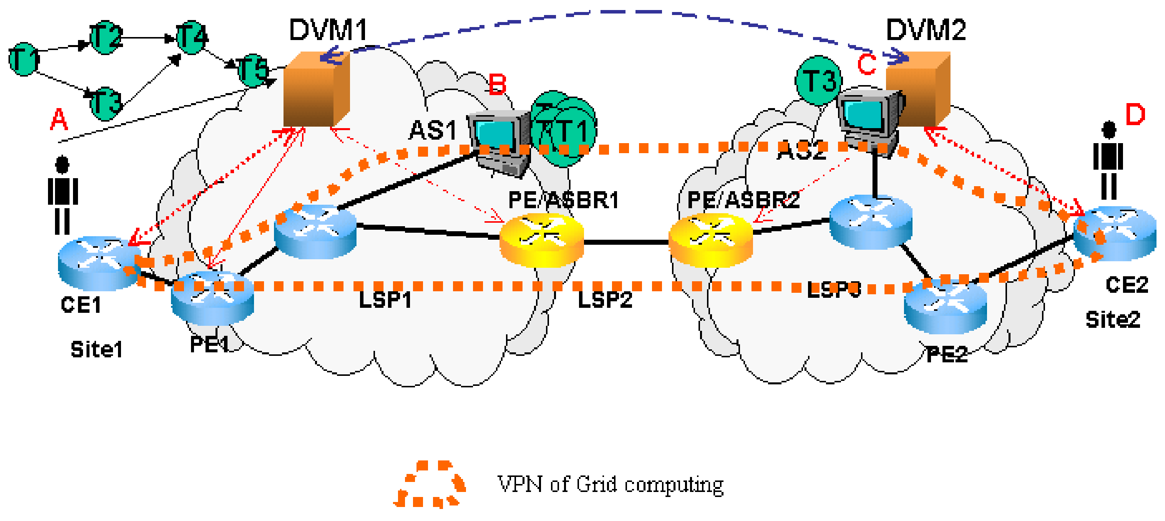


Figure 4.30 A Dynamic VPN Example

After studying the above example, the conclusion can be drawn that the proposed DVPN architecture can provide a dynamic VPN service with automatic resource exploitation and allocation. Meanwhile, the inter-domain QoS and resources exploration are achieved through DVM negotiation. The example demonstrated that this architecture can meet the requirements of the new Internet applications.

4.9 Novelty and Contribution

Although there is already considerable interest in leveraging MPLS as a technology for supporting VPNs, to-date the focus has been on providing a relatively static meshed infrastructure provided by a single carrier or service provider, supplying points of inter-connectivity between community stub networks. The operator provides the transport infrastructure together with means of ensuring VPN isolation across the shared core. The proposed architecture delivers a number of novel components that will enable new forms of highly dynamic groups working to exist. These new features are achieved through extensions of the existing protocols or by consolidating technologies that have hitherto been treated separately. In deliberately following this strategy, rather than proposing fundamentally new approaches, the aim is to provide a clear migration path leading to the realisation of truly dynamic VPNs and the support of flexible group working. The key novel contributions can be summarised as follows:

- The formation and maintenance of VPNs is administered by a new functional entity known as a Dynamic VPN Manager (DVM). This entity acts as an agent liaising between the VPN users and the network connection management. It also takes a key role in inter-provider discovery and VPN formation.
- The DVM concept not only acts as an agent for communications infrastructure booking. It is able to coordinate the resource management of services available within the provider's network, including computational resources and facilities for archival management.
- The DVM provides a means whereby automated VPN community members, such as software processes can request distributed VPN infrastructure and resources to be established and released in response to workflow activities.
- By coupling the DVM with BGP/MPLS signalling, inter-provider Label Switched Path tunnels can be spliced together to produce end-to-end pathways where TSpec requirements can ensure that traffic-engineering constraints are met.

Overall, with the inclusion of the DVM entity, this DVPN architecture provides a novel VPN solution which is the dynamic, customer-active, inter-provider, QoS guarantee MPLS VPN solution.

Chapter 5 Dynamic VPN Prototype

5.1 The Object of the Test-bed

To examine the proposed DVPN and make this architecture practical and close to an industry implementation, the author built a test-bed to represent the DVPN architecture. Only the VPN link setup and remove processes are demonstrated in the test-bed experiment since they are the most important VPN actions for the orchestrated computing. A typical test-bed scenario is represented in Figure 5.1.

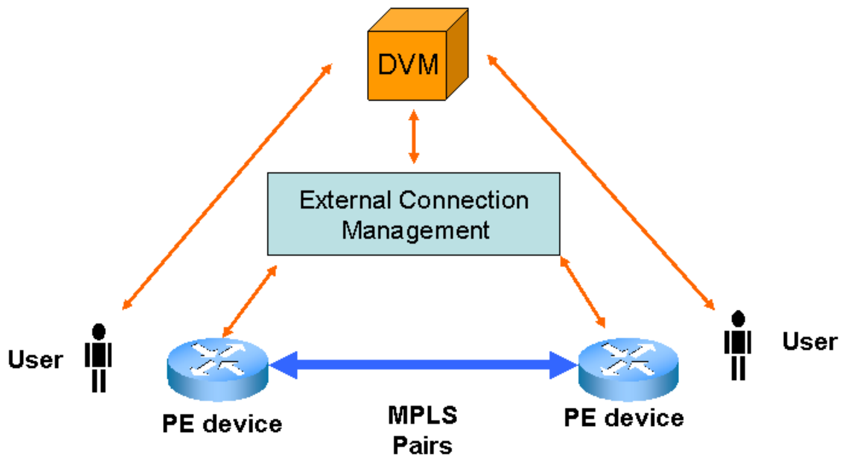


Figure 5.1The Architecture of The Test-bed

The following functions are exhibited in the test-bed experiment:

- The communications between User and DVM: User might send service application to DVM and receive reply through UNI.
- DVM's CM liaison with the External connection management: The DVM might send control message to external connection management in order to fulfil VPN operations (i.e. create/release VPN link).
- The on-demand automatic MPLS VPN Operations: The DVM can deal with user's application and create the VPN link according to user's requirement. The VPN link can also be automatically released when it is not required.
- The different QoS options: There are several QoS options for users; DVM will provide different QoS levels according to the user's requirements.

5.2 MPLS Forwarding and Automatic Control

In this experiment, an MPLS VPN tunnel is created across an IP network. To achieve the MPLS forwarding function, the PE/P routers are developed on the Linux based PCs by using an open source MPLS forwarding software MPLS-2.3 of X-net [128]. MPLS-2.3 [128] is an implementation of MPLS (Multi-Protocol Label Switching) for Linux 2.3.99-pre7 and 2.4.0, supporting IPv4 switch paths over ATM and Ethernet. MPLS-2.3 is developed in C programming language, necessary to be compiled into the Linux kernel. The MPLS works as a Linux module that can be configured by command line or modified directly from other programs by creating a special “netlink” socket. A Linux system with MPLS-2.3 can work as a simple MPLS switch providing the basic MPLS forwarding functions such as ingress mapping, egress mapping, switch mapping, and MPLS packet forwarding functions. MPLS-2.3 can work in PC, and support MPLS switch paths over Ethernet. MPLS-2.3 is employed to provide the underlying MPLS mechanism in the test-bed experiment.

The MPLS tunnel cannot be operated automatically, because MPLS-2.3 only provides the MPLS forwarding mechanism. The label mapping and interface forwarding need to be configured by human operators. However the DVPN requires the dynamic and automatic MPLS operations. For this reason, the author developed an MPLS signalling tool to perform the automatic MPLS operations. This MPLS Controlling Tool (MCT) works as the connection management entity in the test-bed. Figure 5.2 illustrates the architecture of the MCT.

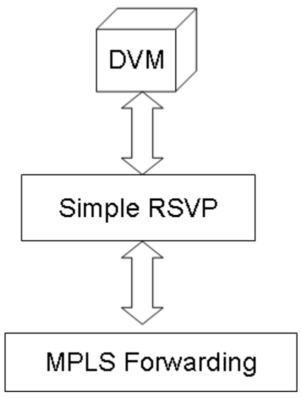


Figure 5.2 The Architecture of MPLS Controlling Tool

The MCT communicates with the DVM through an interface to the DVM. Here a simple RSVP signalling scheme is developed to process the Label advertisements. The MCT operates the MPLS link by sending configuration messages to the MPLS Forwarding Entity through the interface provided in MPLS2.3. Figure 5.3 illustrates how an MPLS path is created. Receiving a “VPN build” (i.e. from network 192.168.1.0/255 to 192.168.2.0/255) message from DVM, the path from ingress PE to the egress PE is selected according to the QoS requirement. In this test-bed, each PE router keeps a routing table. The routes to all the destinations and their QoS are stored in the PE’s routing tables by employing the “IP table” tool provided by Linux. In this example, the selected path is Ingress PE-P-Egress PE. Firstly, Ingress PE sends a label request message (including the path information) to its next hop, P. Then, P sends a label request to its next hop, egress PE. Egress PE assigns a label (i.e. 4) for this MPLS path, adds this label into its forwarding table as this MPLS path’s in label, then sends this label back to P. P used this label as the “out” label, and assigns an “in” label (i.e. 2) for this MPLS path, configures the forwarding table, then sent label 2 back to the label requester, Ingress PE. Ingress PE then uses this label to configure its forwarding table, and the MPLS tunnel is created. The data packets entering ingress PE with destination address of 192.168.2.0/255 from 192.168.1.0/255 will be encapsulated into MPLS packets with label 2. The label is swapped with label 4 at P and the packets are sent to Egress PE. Finally, MPLS packets are de-capsulated into IP packet at Egress PE and forwarded to the destination IP address.

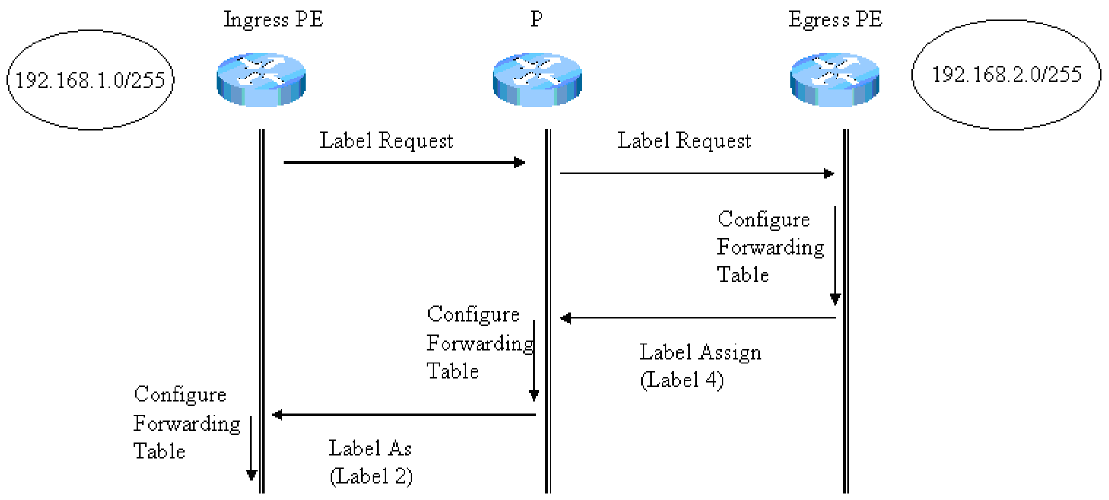


Figure 5.3 Label Advertisement Example

5.3 Test-bed Demonstration

To evaluate the DVPN architecture, a small test scenario was implemented on eight PCs, representing VPN user entities, CE, PE and P routers as well as a DVM, as shown in Figure 5.4.

The DVM is implemented as a web service. Its interface is advertised using Web Service Definition Language (WSDL) [129] and clients written in any language can invoke the operations advertised by the WSDL, as the requests come as Simple Object Access Protocol (SOAP) [130] messages compliant with the advertised interface. Requests to the DVM from web service clients are served by the Axis server [131] and this calls Java methods (corresponding to the WSDL descriptions) to perform actions, e.g. setting up a VPN between points of attachment. The Java methods call the JNI (Java Native Interface) and invoke the appropriate C functions of the DVM itself. Via a web browser, a user can communicate over HTTP with its Web Service Client located on a Customer Edge (CE) router. The web service client is a JSP running in the Tomcat web server [132]. The WS client could communicate directly with the DVM using SOAP over HTTP, but in the scenario chosen several such web service clients communicate with a BPEL processing engine (which is also a web service) also located in the CE that choreographs the interaction with the DVM. The location of the BPEL engine does not need to be in the CE. This is simply for convenience. The BPEL process forwards appropriate VPN requests to the DVM over the IP infrastructure again as SOAP messages over HTTP. Effectively its communication with the DVM is via a Web Service Choreography Interface with the DVM's WS server.

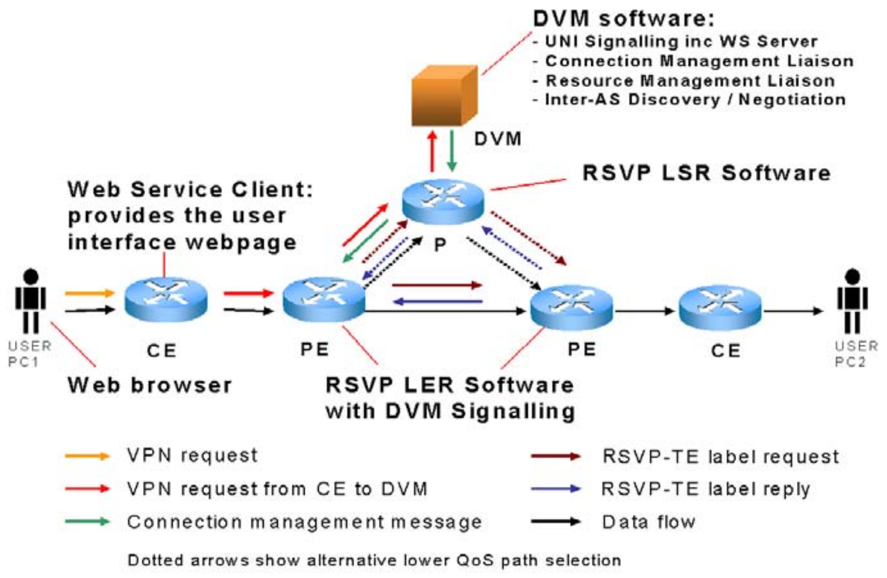

DVM software:
- UNI Signalling inc WS Server
- Connection Management Liaison
- Resource Management Liaison
- Inter-AS Discovery / Negotiation
DVM
RSVP LSR Software
Web Service Client: provides the user interface webpage
P
USER PC1
CE
PE
PE
CE
USER PC2
Web browser
RSVP LER Software with DVM Signalling
VPN request
VPN request from CE to DVM
Connection management message
RSVP-TE label request
RSVP-TE label reply
Data flow
Dotted arrows show alternative lower QoS path selection

Figure 5.4 Dynamic VPN Test Scenario and the actual Test-bed

The RSVP-TE protocol is implemented by the author. The layer 2 technology is Ethernet, which is easy to employ in the PC. The LSP processes (creation, adjustment, removal) are triggered by the commands from the DVM. The result will be returned to the DVM, either as execution success or failure. User can view the result in a web-browser such as Netscape.

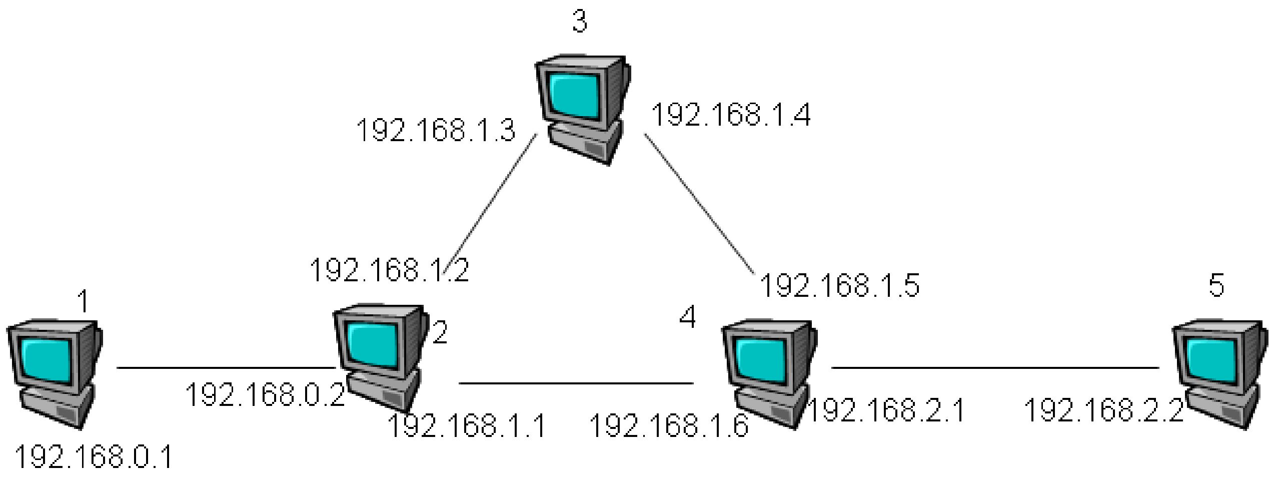


Figure 5.5 Test-bed Topology

Figure 5.5 represents the topology of the test-bed. The DVM entity is realised in computer 3. Computers 2 and 4 are PE routers and 3 is a P router. Computers 1 and 5 are CE routers. During the experiments Ethereal [133] is employed as the packet sniffer.

In the first experiment, the user chooses to create a VPN link from computer 1 to 5 with service level 1 (a higher PRI) (on the web-browser at computer 1). The VPN link from 1 to 5 is created successfully and the LSP is created along the path 2-4. After “remove this VPN link” is selected by the user, this VPN link is removed. In the second experiment, the user chooses service level 2 (a lower PRI) for the VPN link from computers 1 to 5. The VPN link from 1 to 5 is created successfully and the LSP is setup along 2-3-4. After "remove this VPN link" is selected by the user, this VPN link is removed.

Figure 5.6 is the photograph of the test-bed.

Figure 5.6 The Prototyping Test-bed

As mentioned before, the DVM is responsible for controlling VPN membership. It is also responsible for liaising with separate operator owned connections and resource management to selectively interconnect community members via MPLS LSPs between their Provider Edge (PE) router points of access to the core network. It also determines access to operator or third-party resource islands that it can make accessible to specific communities. In the test scenario, a user entity joins a community and is then able to communicate over a dynamically configured LSP with a remote member of the same community. This is achieved by the DVM liaising with connection management software located in the ingress PE router, which sets up an LSP between the identified PE endpoints. The path taken by the LSP is chosen by the connection management software based on the QoS constraints identified by the DVM. In one instance it is direct communication between the PE routers (when a latency sensitive pathway is requested) or, alternatively, via an intermediate P router, when a longer path can be tolerated. However, it is important to note that the DVM does not select the path itself; this is a connection management task.

This test-bed is the first practical dynamic IP/MPLS VPN example, it not only demonstrates the feasibility of the DVPN concept but also provided a platform for the further research. With this test-bed, the author can carry out experiments which are difficult and complex to accomplish in simulations. For instance it is easy to examine the DVPN system performance against DoS attacks. There are many DoS tools available in the Internet Hping, Trinoo, the author could run the DoS attack tools at the users and obverse the behaviour of the test-bed with the system monitor tools available in Linux. (Ethereal-catch packet; system monitor-system performance monitor). Because the test-bed is developed based on open source code, it is also not difficult to make modifications and further development. The above is an example of further development to the existing open source code. Moreover, the author could access this test-bed via the Internet and invite other researchers and hackers to examine the DVPN architecture against attacks.

5.4 Summary

In this chapter, the author describes the DVPN test-bed. A prototype is developed with Linux PCs. Open source MPLS forwarding software is employed to provide the MPLS forwarding and an MPLS control tool with a simple RSVP signalling is developed by the author. The user can submit service requests through the web interface to the server located at DVM. The MPLS VPN link can be created/released automatically according to the user request satisfying QoS. This test-bed demonstrates the basic DVPN processes. It also provides an open platform for further development and research.

Chapter 6 Dynamic Job Scheduling

6.1 Dynamic Resource Scheduling Algorithm Architecture

The DVPN architecture is designed to support services for applications that require access to on-demand and dynamic resources (i.e. orchestrated computing). Figure 6.1 shows an orchestrated computing example in the DVPN system. User1 and user2 are human users. User 1 needs to perform a science calculation job. This job can be represented as a DAG. CPU processors are required to carry the tasks of this job. In this example, tasks 2 and 3 cannot start until task 1 completes. Task 3 also requires user 2 to provide necessary data. After tasks 2 and task 3 complete, tasks 4 can start. The final task, tasks 5, is to sending the results back to user1.

There are two resources R1 and R2 available in this example scenario. Assuming task1 is carried in R1, a VPN link is created from user1 to R1 for transferring data. After the data transaction completes, this VPN link is removed. After task 1 completes, tasks 2 and tasks 3 can start. Since there is no dependence between these two tasks, tasks 2 and 3 can be processed in different resources at the same time. As shown in Figure 6.1, task 2 is carried out in R1 and task 3 is carried out in R2. Meanwhile, a VPN link is created from user2 to R2 for the data transaction from user2 to R2. This VPN link is removed after the data transaction finishes. After tasks 2 and 3 complete, task 4 can start. Assuming task 4 is allocated to R1, a VPN link is created from R2 to R1 for transferring result data of task 3 from R2 to R1. As in the former case, this VPN link is removed after the transaction finishes. After task4 completes, a VPN link from R1 to user1 is created out and the final result data is sent back to user1.

In the above example, the calculation tasks (Task 2 and Task 3) are allocated to network resources R1 and R2. In the DVM based DVPN system, there might be several jobs co-existing, which makes the resource scheduling a dependent/independent workflow management problem. The job scenario, network condition and customer requirements are dynamic. As analysed in Chapter 3, none of the existing solutions are suitable for this job scenario. For this reason, a new resource-scheduling algorithm is proposed.

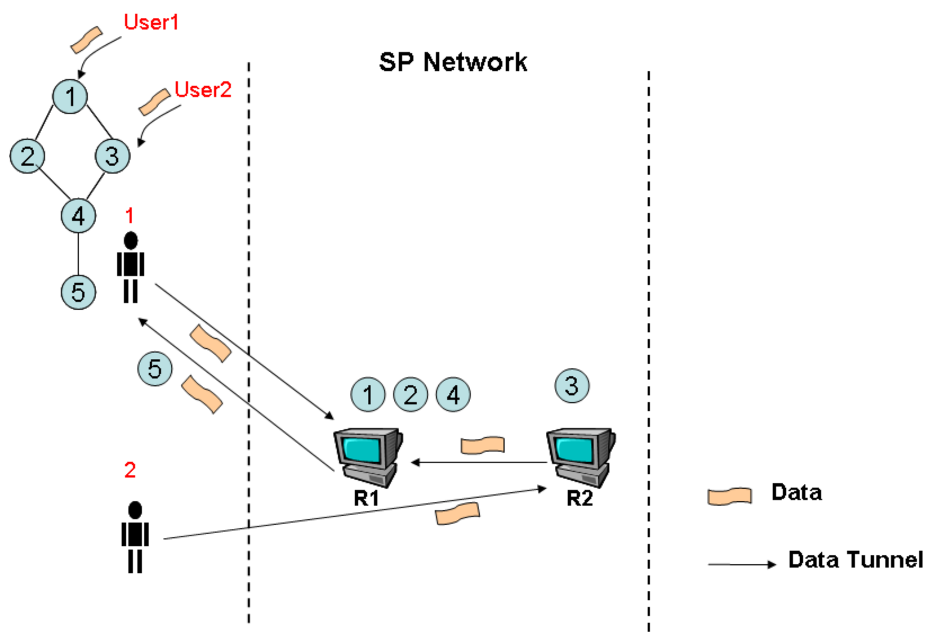
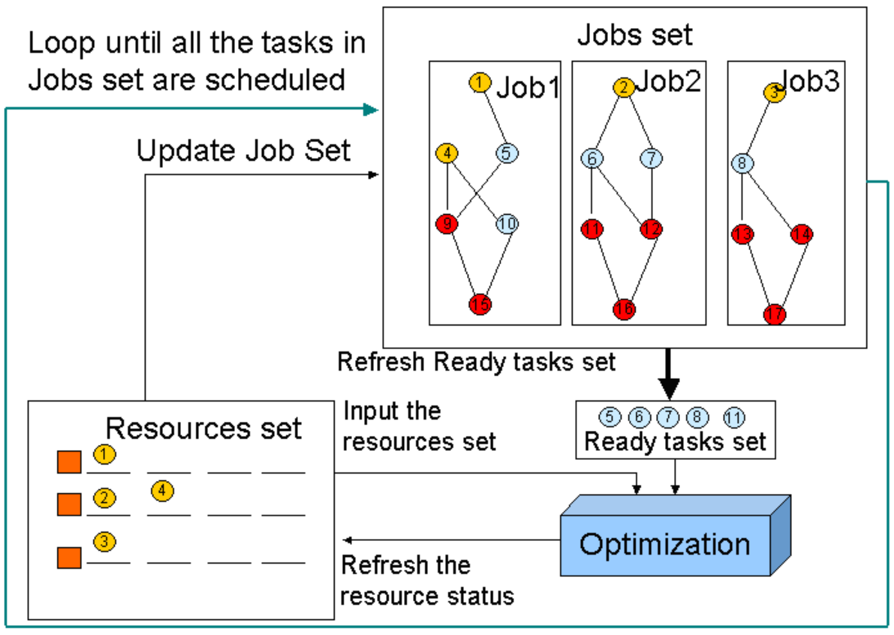


Figure 6.1 The Orchestrated Computing Example

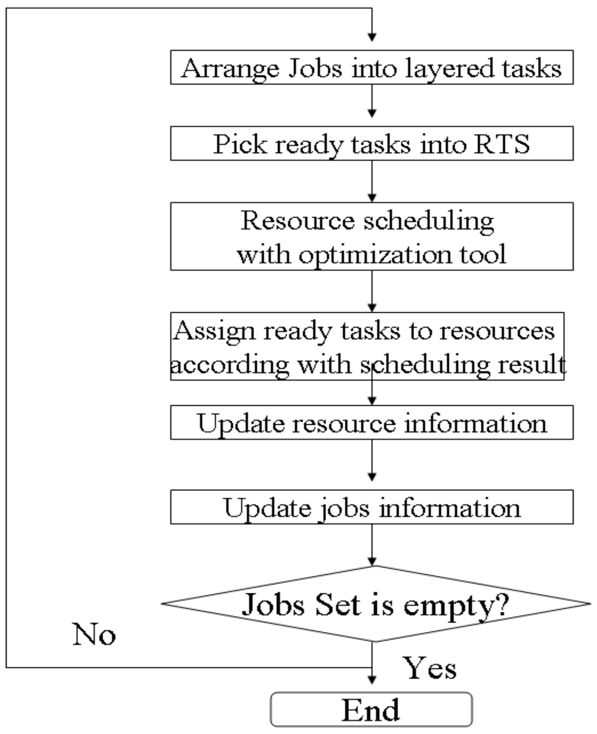
The architecture and process flowchart of this scheme is shown in Figure 6.2(a) and (b), respectively. The jobs are represented as DAGs and the information (i.e. tasks dependency) of the jobs is stored in the Jobs Set (JS). The Resource Set (RS) records the resource information including: availability, capacity and the currently allocated tasks.

The algorithm is designed to operate with dynamic scenarios where new tasks/jobs can arrive at any time. As there are typically dependencies among tasks comprising each job, the tasks are separated into different layers in accordance with their dependency constraints. The tasks in the succeeding lower layer are always compared with their preceding upper layer dependent tasks.

A task is "ready" for scheduling only if all of its upper dependent tasks are already scheduled and their completion times are known. The scheduled tasks are the yellow nodes and the ready tasks are represented in blue. The red nodes indicate "un-ready" tasks that have some incomplete upper dependent tasks and so cannot be scheduled yet. The ready tasks are input into the Ready Tasks Set (RTS). All the tasks in the RTS are independent because none of them have any un-scheduled dependent uppers, even though some of them may be associated with the same job.



(a) The Algorithm Architecture


Arrange Jobs into layered tasks
Pick ready tasks into RTS
Resource scheduling with optimization tool
Assign ready tasks to resources according with scheduling result
Update resource information
Update jobs information
Jobs Set is empty?
No
Yes
End

(b) The Algorithm Process Flowchart

Figure 6.2 The GA-Based Resource Scheduling Algorithm

A “standard DAG” is defined in this thesis as representing a DAG whose tasks can be separated into levels according to their dependency relationships. Occasionally, it happens that the job scenario can not be represented as a standard DAG. For example refer to Figure 6.3. This proposed algorithm is also feasible in this case, because only the ready tasks are put into the RTS in each of iteration. Considering the example in

Figure 6.3, the algorithm processes as follows: in the first algorithm iteration, task 1 is the ready task. After task 1 completes, task 2 is ready while task 3 is not ready until tasks 2 completes. After tasks 3 completes, tasks 4 is ready to be scheduled.

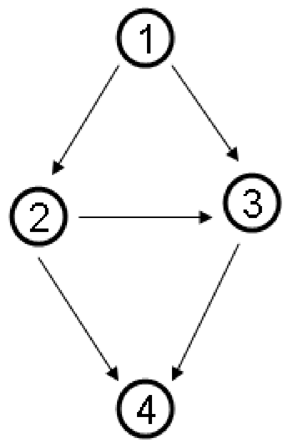


Figure 6.3 A Node-Standard DAG

The ready tasks are scheduled using an optimisation engine. In this thesis, both a GA [134] and PSO [135][136][137][138] approaches are considered as they are currently regarded as the robust and efficient stochastic searching mechanisms for various optimisation problems [135] [138].

Whenever jobs arrive they are placed within the Jobs Set and arranged into layered tasks, bearing in mind their dependencies. From these jobs, tasks that have no dependent uppers may then be transferred to the RTS in preparation for a scheduling cycle. The scheduling processing cycle is initiated when there are tasks waiting in the RTS. While it executes, trying to find a suitable allocation of ready tasks to the available resources, no further tasks may enter the RTS. At the end of a scheduling cycle, the tasks in the RTS are assigned to the identified resources along with their start time. Also at this point the algorithm re-examines the Jobs Set, which may contain additional jobs that have arrived during the previous processing cycle as well as the residual tasks of the jobs that are currently being processed.

Tasks effectively enter the RTS in a step-wise fashion. Various mapping arrangements onto the available resources are attempted until the one with the smallest *overall completion time* is found within a given number of iterations of the GA (or PSO). The task resource allocation information is updated, and then another GA scheduling cycle starts. The processing mechanism loops until all the tasks are scheduled, which means that no more tasks/jobs enter the network and the RTS is empty. The process is represented in Figure 6.2 (b).

The advantage of this approach is that the optimisation engine does not need complete information of all the jobs that will arrive; but only considers the tasks in the RTS; it possesses no foresight. The RTS updates in accordance with the current job information before each GA instance starts. The algorithm can accommodate new unexpected jobs entering the system. It is also possible that the resource availability can change whilst the algorithm is processing. The ability to accommodate these situations makes this algorithm feasible for use within the dynamic DVPN scenario.

6.2 Optimisation with GA and PSO

Alternative GA and PSO optimisation engines are considered for scheduling the ready tasks in the RTS at each resource-allocation cycle. Their performance is examined using simulations.

To illustrate the resource allocation mechanism and the role played by the optimisation algorithm, consider the simple scenario depicted in Figure 6.4. This shows an example job dependency DAG and corresponding job scheduling arrangement in Figure 6.4 (a) and Figure 6.4 (b), respectively.

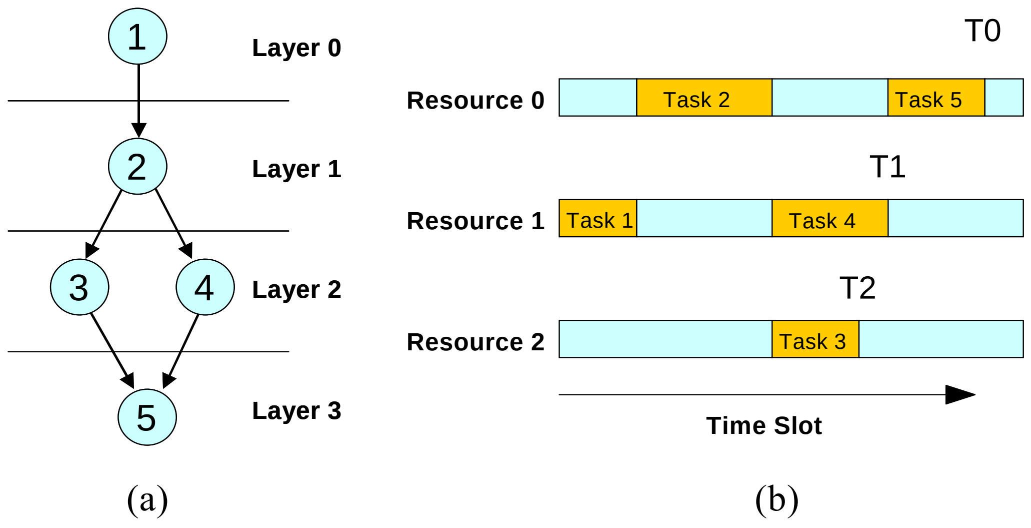


Figure 6.4 Example Job Scheduling Arrangement

In this example, a job consists of five tasks and is represented as a DAG. There are 3 network resources. The tasks are separated into different layers and numbered in accordance with their specific dependencies as shown in Figure 6.4 (a). A task's number can never be smaller than the tasks in its upper layers. A task cannot start until all its dependent uppers have finished execution. Considering the job scenario shown in Figure 6.4 (a), a possible scheduling arrangement might look like Figure 6.4 (b). T0 (T1, T2) is the completion time of Resource0 (1, 2).

6.3 GA Optimisation Engine Implementation

6.3.1 GA Chromosome

GA [92] attempts to obtain an optimal solution by simulating the natural evolutionary process. The chromosome architecture employed in the proposed algorithm is illustrated in Table 6.1. Each chromosome consists of a number of genes, where each gene represents a particular ready task awaiting scheduling. The value of the gene indicates which resource is under consideration for allocating this task to. The size of chromosome (the number of genes) equals the number of ready tasks in the RTS. In the example of Table 6.1, because there are 5 tasks in the RTS, the chromosome includes 5 genes representing the 5 ready tasks. Assuming that there are 3 available resources (resource 0, resource 1 and resource 2), the allocations of the current ready tasks are shown in Table 6.2 in accordance with the chromosome shown in Table 6.1.

Table 6.1 The Chromosome Architecture

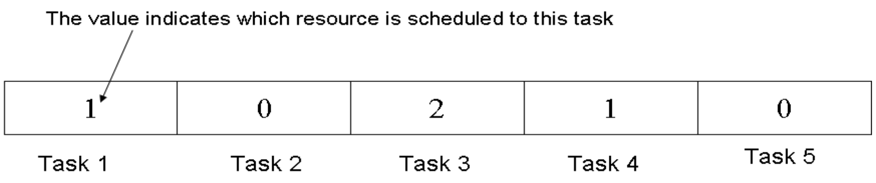

The value indicates which resource is scheduled to this task
1
0
2
1
0
Task 1
Task 2
Task 3
Task 4
Task 5

Table 6.2 The Relationship between The Tasks and the Resources Represented by Table 6.1

Tasks to be scheduled	**Available resources**
Task 1	Resource 1
Task 2	Resource 0
Task 3	Resource 2
Task 4	Resource 1
Task 5	Resource 0

The chromosome only contains the mapping information of tasks to the resources; however, the dependencies are considered when the fitness is calculated and tasks are assigned to resources. All the tasks are separated into different layers and numbered in accordance with their specific dependencies as shown in Figure 6.4 (a). A task cannot start until all its dependent uppers have finished execution. Considering the job scenario shown in Figure 6.4 (a), in accordance with the chromosome illustrated in Table 6.1, the scheduling arrangement is shown in Figure 6.4 (b).

6.3.2 Computing the Fitness of Chromosome Fitness

The goal of GA is to find the minimum *Overall Completion Time*. How a chromosome fits this requirement is represented with the *Fitness*. Here, a greater fitness value implies a lower the overall completion time. The fitness is calculated as follows:

Firstly, calculate the completion time of each resource $T_1, T_2, ...T_n$ which means the time spent by resource *n* to execute all tasks assigned to it (refer to Figure 6.4 (b)).

Secondly, set the overall completion time (T) as when all the resources have finished their tasks:

$$\mathbf{T=max}(T_1, T_2, ...T_n). \qquad \mathbf{(6.1)}$$

Thirdly, the fitness is calculated:

$$\mathbf{fitness=}\boldsymbol{C}\mathbf{\text{-}max}\ (T_1, T_2, ...T_n) \qquad \mathbf{(6.2)}$$

where C is a large constant integer.

6.3.3 Crossover Process

The crossover operator selects several chromosomes randomly from the population and “mates” them by picking genes and then swapping them between two of chromosomes randomly. In each crossover, a random gene location is picked, and then this gene and all genes to the "right" of it (those with greater location) are swapped between the two chromosomes. For example, two original parent chromosomes shown in Figure 6.5 are selected for crossover.

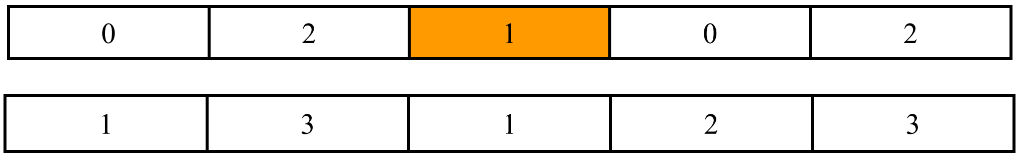


Figure 6.5 Original Parent Chromosomes

A random gene location (3rd) is picked for crossover, shown as Figure 6.6:

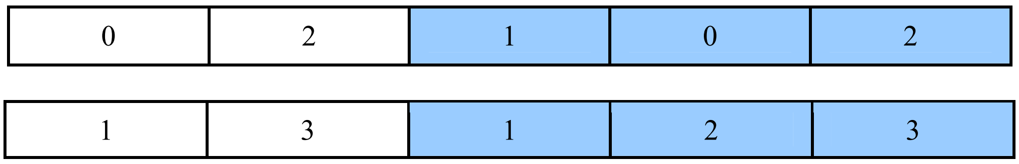


Figure 6.6 Crossover

In this example, the 3rd gene is selected as the position for crossover, the two parents chromosomes swap their 3rd , 4th, 5th genes. After the crossover, two new child chromosomes are produced:

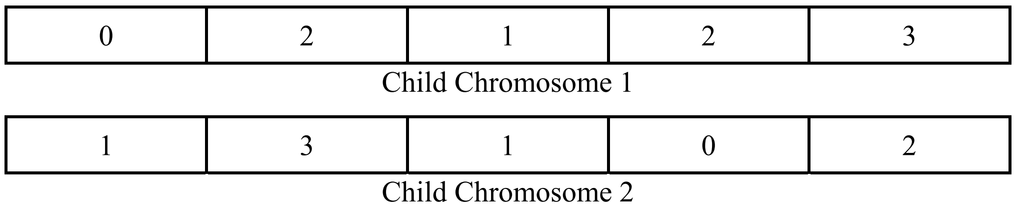


Figure 6.7 New Child Chromosomes

In this thesis, the crossover rate is defined to represent the probability that chromosome crossover will occur. Unless otherwise stated, the crossover rate is set to 50% (with is default in JGAP [139] and shows a good performance).

6.3.4 Mutation Process

The genes of chromosomes maybe mutate randomly. Because of mutation, it will cause new chromosomes to arise, possibly including ones closer to the optimal chromosome, and so avoid the undue influence of the local maxima/minima. The mutation rate is defined as representing the percentage of total genes that mutate at each of GA iteration. How the mutation rate affects this proposed algorithm performance is studied in the later sections.

6.3.5 Evolution Process

The evolution process selects a new population of P members preferentially by choosing the better members according to the fitness of each chromosome.

Let f_i be the fitness of member i

$$P_i = f_i / \sum_{k=1}^{m} f_k \quad \textbf{, where } P_i \textbf{ is the probability of choosing i.} \tag{6.3}$$

In order to ensure the excellence chromosomes with greater fitness value are passed to the next generation, roulette wheel selection [140] is used, which represents the fitness process in the natural world.

6.3.6 Simulation Scenarios

To evaluate the algorithm performance, simulations are carried under random generated job scenarios. In this thesis a job scenario consists of tasks separated into multiple levels. Each task consumes a certain amount of the CPU resource. In this thesis, the *task burden* is defined as:

Task Burden = Task Computation Time* x *CPU Capacity **(6.4)**

For example, if a task with burden 4 is allocated to a resource with “capability” 2, the task will take two time units to complete. For simplicity, we assume that tasks have no specific resource requirements and so can be potentially allocated to any of the available resources. In this thesis, unless otherwise stated there are 3 resources (R1, R2, R3); the CPU capacity of each resource is 1, 2 and 3 respectively. Every task might have zero or more upper dependent tasks located in the upper level. A task cannot start until all of its dependent upper tasks finished. Figure 6.8 shows an example of a 3 levels job scenario.

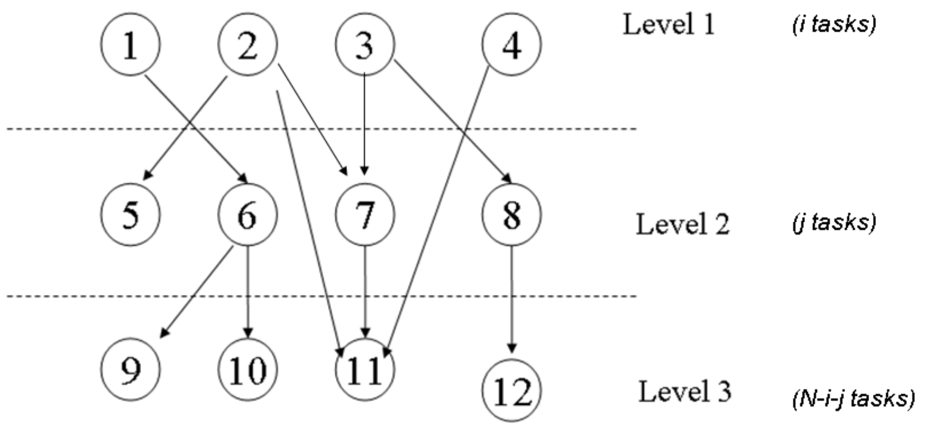


Figure 6.8 Job Scenario Example

The job scenario is generated as follows. Firstly, N tasks (for example N=60) are generated. The burden of each task is uniformly randomly distributed (i.e. between 1 and 240). Then these tasks are separated into L levels (L=3 in Figure 6.8). The tasks 1 to i are located into level 1. i is a uniformly randomly-generated number between 1 and N-2. The tasks i to j are located into level 2, where j is a uniformly randomly-generated number between i+1 and N-1. The rest of the tasks are located in level 3. Thus there are i tasks in level 1 (Numbered from 1 to i), j-i tasks in level 2 (Numbered from i+1 to j), and N-i-j tasks in level 3 (Numbered from j+1 to N). The tasks in top level have no dependent uppers, and each task in the other levels (level2 and level3) has on average D (i.e. D=5) Normally distributed (with mean=1, variance=1) dependent tasks in its preceding upper level. As a result, tasks are randomly arranged into several jobs that consist of a set of dependent tasks. The Random Number Generation (RNG) used in the simulations is provided via the Java function (java.util.Random), and the system time is used as the RNG seeds when repeating independent runs.

6.3.7 Simulation Results Analysis

The proposed algorithm is studied through simulations under different job scenarios that are randomly generated. Unless otherwise stated, the GA parameters are set as same as the JGAP's [139] default configurations, however, these parameters are also studied with simulations:

- Crossover rate: 50%
- Mutation rate: 10%
- Survival rate: 95%

Unless otherwise stated, all computation time measurements given in the simulation figures are expressed in milliseconds. However, the simulated time units (i.e. the time it takes to complete a set of tasks) are represented in multiples of a unitary none-specific time length

In Figure 6.9, the proposed dynamic GA based algorithm is compared against a static GA algorithm, both with a fixed chromosome population in this simulation. The static GA algorithm requires complete information concerning the whole scenario, specifically when each and every job will arrive and its burden. In the static case all the DAG tasks and the available resource information are read in before the GA starts. A similar chromosome arrangement is used in both the proposed dynamic algorithm and the static GA. The proposed algorithm outputs the scheduling arrangements for each set of ready tasks and terminates only when no more jobs enter the system. Conversely with the static GA the final scheduling is output only once. This static algorithm provides a useful upper bound on the performance, as this complete knowledge of the job arrival sequence would not normally be available in a dynamic scenario.

In the simulation, the number of generations for both algorithms is set to 10 (which can provide good performance and short time consuming according to the simulation results). Five job scenarios were examined. All of them contain 100 tasks while the average number of upper dependent tasks of each task is different in each case. For example, the green line demonstrates the simulation results under the scenario where each task has on average 4 dependent uppers; the black one is for the job scenario whose tasks have on average 28 dependent uppers. The results shown are the average of 100 independent trials, where different RNG seeds are used for the heuristic optimisation process.

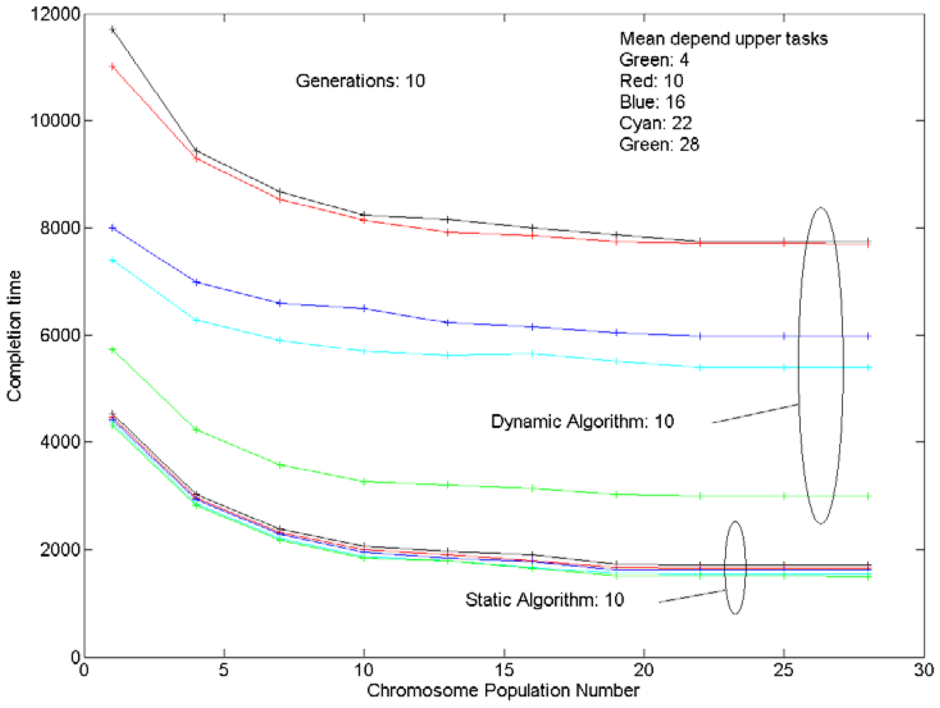


Figure 6.9 Overall Completion Time Performance

From the simulation results, we can draw the conclusion that the dynamic algorithm's performance is similar to the static one in scenarios where the upper dependent task number is small, and the dynamic algorithm performs worse when the upper dependent task number of each task increases. As expected, the dynamic algorithm works well if the tasks are not very dependent, otherwise this algorithm will lead to a worse scheduling compared to the static case, where it has the luxury of a complete job forecast. The reason is that the dynamic GA only considers the ready tasks, while the static GA schedules according to the whole scenario; the dependent relations are considered along with knowledge of when all jobs will arrive. As a result, if these tasks are very dependent, the lack of the dependency information will lead to worse scheduling performance with the proposed algorithm.

The relationship between chromosome population size and performance variability is studied with the results shown in Figure 6.10. The chromosome population is set from 1 to 40 for each scenario and the simulation with same population number is carried out 20 times (each “x” represents the simulation result of one time). According to the results we can draw the conclusion that increasing the chromosome population to a certain extent will reduce the variability in the results while a too large population does not reduce the variability further, implying that increasing the chromosome population beyond certain values is of limited or no benefit.

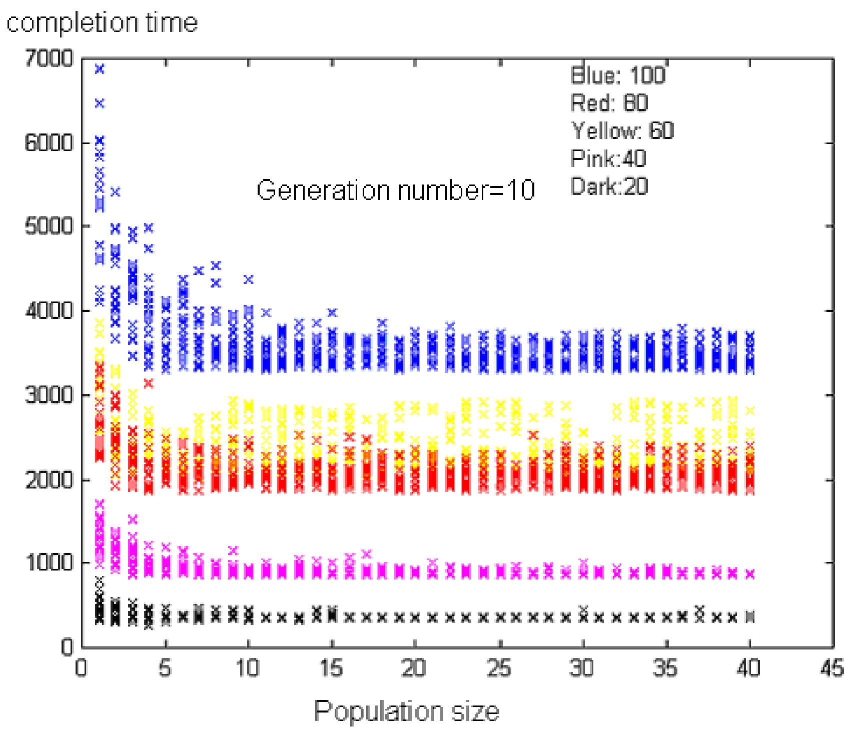


Figure 6.10 Impact of Population Size Job on Completion Time Performance-with a Fixed Chromosome Population Size

From the above simulation results, we can find that the proposed algorithm achieves a better performance when the chromosome population number increases over a certain range. As analyzed above, the task number in the RTS changes from time to time and the GA’s performance is related to the ratio of population versus the ready tasks, as implied by the Figure 6.9 and Figure 6.10. Therefore, the GA algorithm was modified to adjust its population size according the current number of ready tasks in the RTS. The *population ratio* is introduced as the parameter representing the ratio of chromosomes to the number of ready tasks. For example, with the population ratio 0.5, the chromosome number should be 10 if there are currently 20 ready tasks waiting for scheduling. This adaptation ensures the complexity of the GA calculation is kept small when the presence of fewer ready tasks makes it appropriate to do so**.** The GA with an adaptive chromosome population size was compared against the one with fixed population size. The simulation results are shown in Figure 6.11.

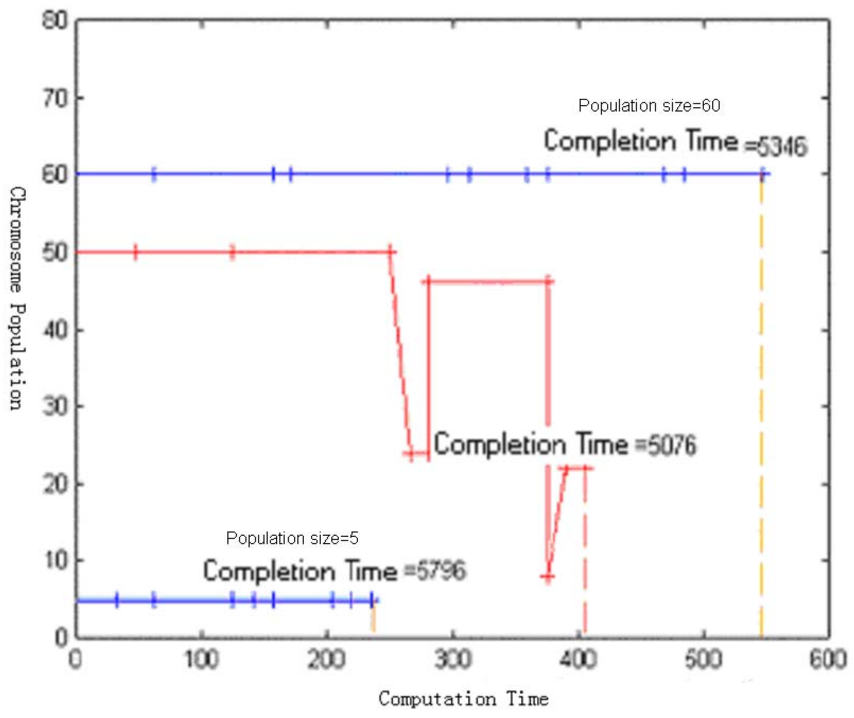
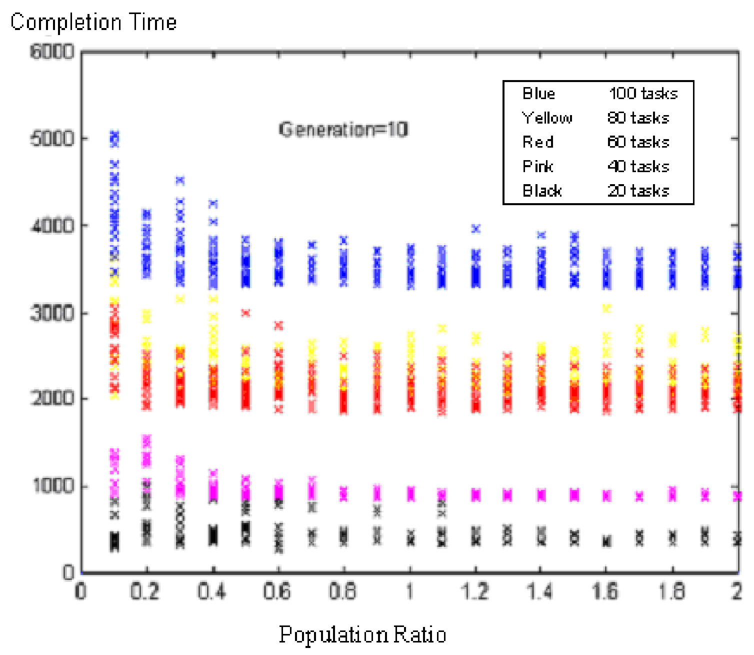


Figure 6.11 Adaptive Chromosome Population Scheme

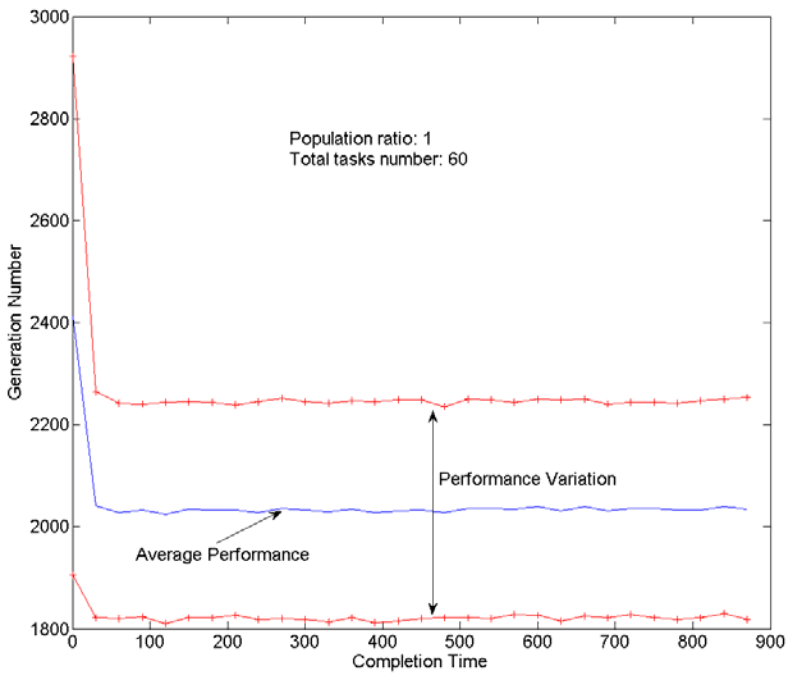
In the simulation, the job scenarios each contain 100 tasks in total and 3 resources are available. Each task has on average 10 dependent uppers. The horizontal axis is the consumed CPU time (i.e. how long the CPU spends calculating the scheduling arrangement) and the vertical axis is the chromosome population. In this thesis, all the simulations are carried on an IBM ThinkPad X61 PC Notebook with an Intel *Core 2 Duo* 2 GHz processor and 1GByte RAM. Two simulations with the fixed population sizes (5 and 60) are carried out along with one simulation where the GA has the adaptive population size. As already defined, the population ratio is the ratio of chromosomes to number of tasks in the current assignment level. The population ratio was set to 1 for the adaptive population GA in this simulation, which means that the population number always equals the current number of ready tasks.

The simulation results show for the GA with a fixed population size, too small a fixed population number leads to poor scheduling, whilst too large a population number causes the CPU to spend too much time calculating the assignment although its output scheduling is somewhat better than the GA with the smaller population number.

On the other hand, the GA with adaptive-population size (red line in Figure 6.11) consumes less CPU time whilst outputting a better scheduling result compared with both GA cases with fixed chromosome population sizes. Its population size changes over time in order to maintain an appropriate population size according to the number of instantaneous ready tasks. We can conclude that the GA with adaptive population size is more efficient than schemes with a fixed population size when considering the consumed CPU time and the resultant scheduling allocation.



(a) Impact of Population Ratio on Overall Job Completion Time Performance

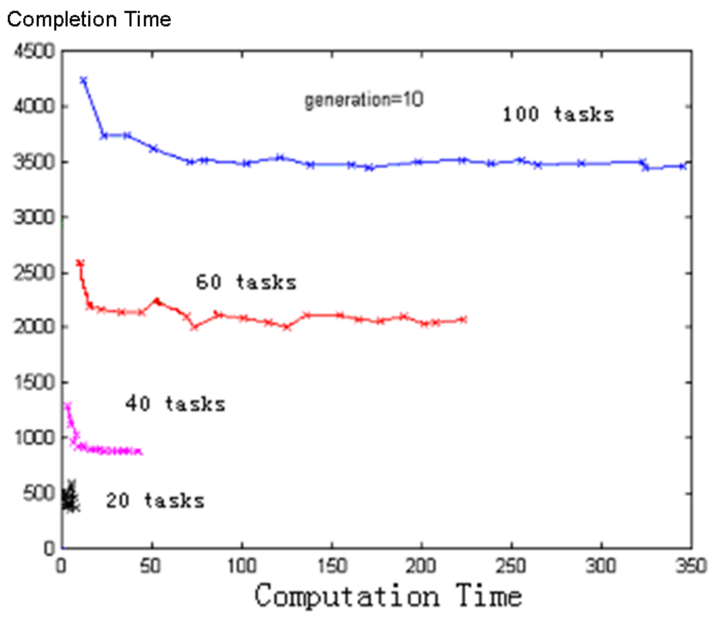


(b) Impact of Generation Number on Overall Job Completion Time Performance

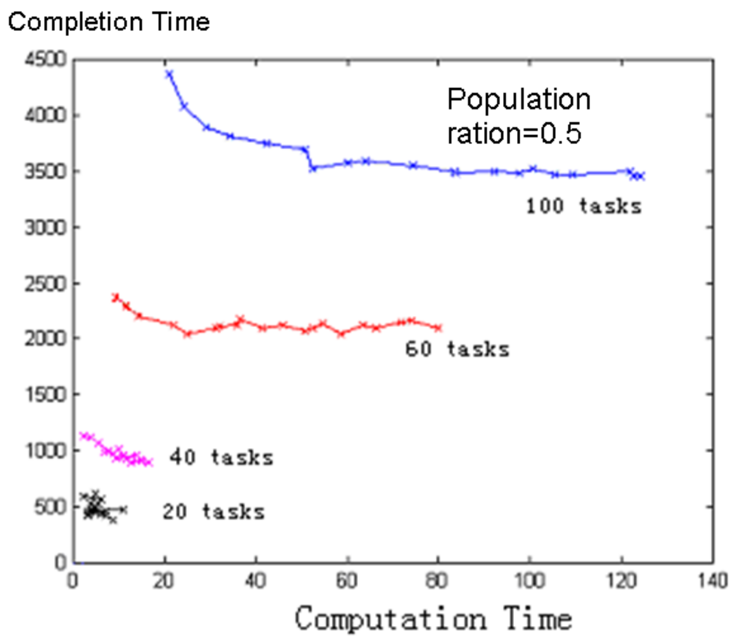
Figure 6.12 Impact of GA parameters on Overall Job Completion Time Performance

Figure 6.12 examines the relationship between the GA parameters (population ratio and generation number) and the scheduling allocation performance. Five scenarios with 20, 40, 60, 80 and 100 tasks are considered. Again there are 3 available resources. Shown as Figure 6.12 (a), in each scenario, simulations with different population ratios are performed and 20 repeats are carried out with the same population ratio (where a random number generator is used to create various job arrival permutations). Based on the results in Figure 6.12 (a), we see that increasing the population ratio over a certain range can improve the algorithm's scheduling performance and reduce the performance variation; however, further increases in the population ratio have a minimal impact. Figure 6.12 (b) represents the relationship between the Algorithm behaviour and GA generation number. Similarly, simulations with different generation number are performed and 20 repeats are carried out with same generation number under the same job scenario. The population ratio is set to 1 for all the simulations. A similar conclusion can be made that a too large the generation number (over 30) does not improve the overall completion time in the DVPN scenario. This is the reason why the generation number is set to 10 in Figure 6.12 (a) rather than a larger value.

As the population ratio or generation number increases, the time spent calculating the scheduling also increases. Figure 6.13 shows the relationship between the computation time and the overall job completion time for the proposed algorithm. In Figure 6.13(a), the population ratio varies from 0.1 with a step length 0.1 and the number of generations is kept as 10. As can be seen from the vertical axis, the range of job completion times achievable by different resource scheduling arrangements is limited in all cases. Also, especially when the RTS has more tasks for processing, increasing the chromosome population ratio can have a significant impact of the CPU time taken to arrive at a result for a given number of GA generations (i.e. from the increasing spread of values along the horizontal axis as the number of tasks mounts). This impact has little performance benefit as seen by the eventual plateau. Meanwhile, shown as Figure 6.13 (b), the proposed algorithm demonstrates a similar behaviour when the generation number varies from 1 with step length 3, with a fixed population ratio of 0.5. This result is quite useful for configuring GA parameters. As the DVM needs to achieve a good result in a limited time, we have the ability to trade off between Computation Time and GA scheduling performance.



(a) Fixed Generation Number



(b) Fixed Population ratio

Figure 6.13 Relationship Between Overall Job Completion Time Performance and Computation Time

Two of the GA parameters that affect the GA's performance are the survival rate and mutation rate. The survival rate is defined as how many chromosomes of the original population will be considered for selection to the next population. For example 0.5 survival rate means half of the chromosomes are considered. The fitness values are employed as a guide. Usually fitness is treated as a statistical probability of survival (the Normal distribution was employed in the simulations). Therefore, Chromosomes with higher fitness values are more likely to survive than those with lesser fitness values, but it's not guaranteed. When the survival rate equals to 1 it means that all of the original chromosomes will be considered for selection for the next generation population. The survival rate determines how many chromosomes will be selected into the next generation. A series of simulations are carried with different survival rates. In Figure 6.14 the horizontal axis is 1/(survival rate) and the vertical axis is the job completion time. The mutation rate is set to 0.1, which means 10 percent of the genes mutate. The results show that too small a survival rate will degrade the GA's performance. This is reasonable as the survival rate means a small ratio of the chromosomes in the current generation will be passed into the next generation and a large percentage of chromosomes will be introduced through crossover to maintain the population size. As a result, a lot of "good" chromosomes cannot be selected into the next generation, which leads the GA to be inefficient.

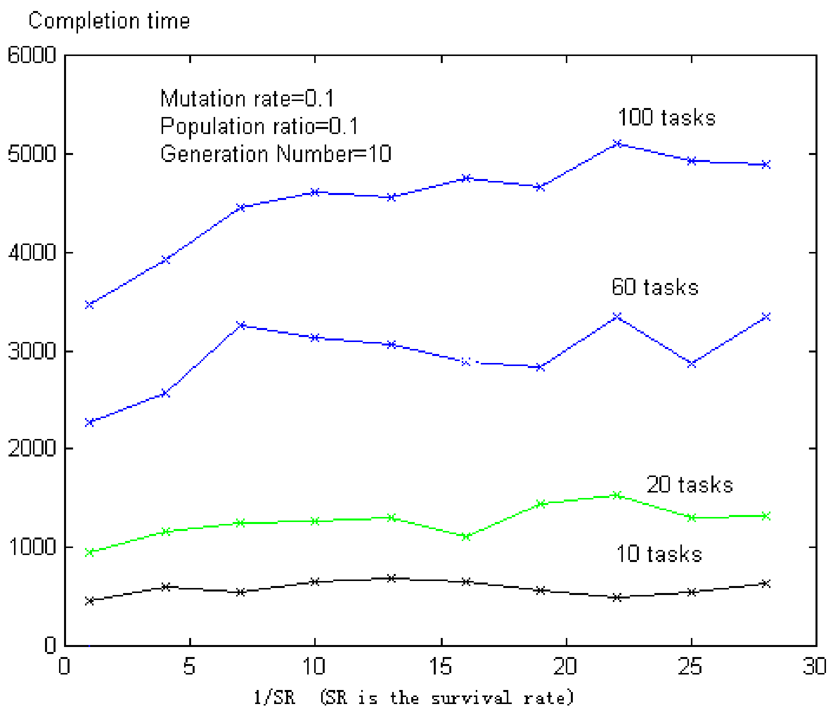


Figure 6.14 Completion Time with Different Survival Rates

The mutation rate defines the percentage of the genes that will mutate. This is also an important parameter to the GA. [141] showed that low level of mutation serves to prevent any given bit position from remaining forever converged to a single value in the entire population. A high level of mutation yields an essentially random search. The higher mutation rate also helps prevent premature convergence to local optima, which is quiet important when the population size is small and the search space is very large. The GA with the lower mutation rate converges more quickly but it is more likely to lead to a suboptimal solution.

Previous research has confirmed that the mutation rates might affect the GA performance and the optimal mutation rate should be selected in accordance with the individual problem [142][143][144][145][146][147]. Normally, the optimal mutation rates are obtained by experiments [148].

Simulations with different mutation rates are carried out to study how the mutation rate affects the GA performance. In the Figure 6.15, the horizontal axis is the 1/(mutation rate) and the right hand side corresponds to a smaller mutation rate. The result shows that the algorithm's performance is not very related to the mutation rate. As described above, the extent to which the mutation rate affects the GA performance is related to the individual problem. The conclusion that can be drawn is that the mutation rate does not affect the GA performance very much for the scenarios under study.

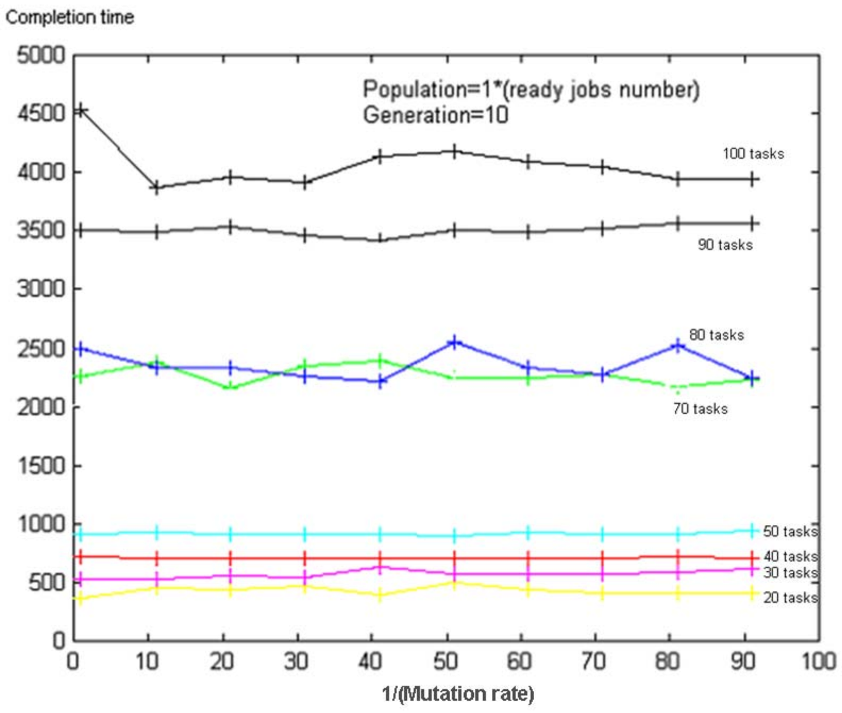


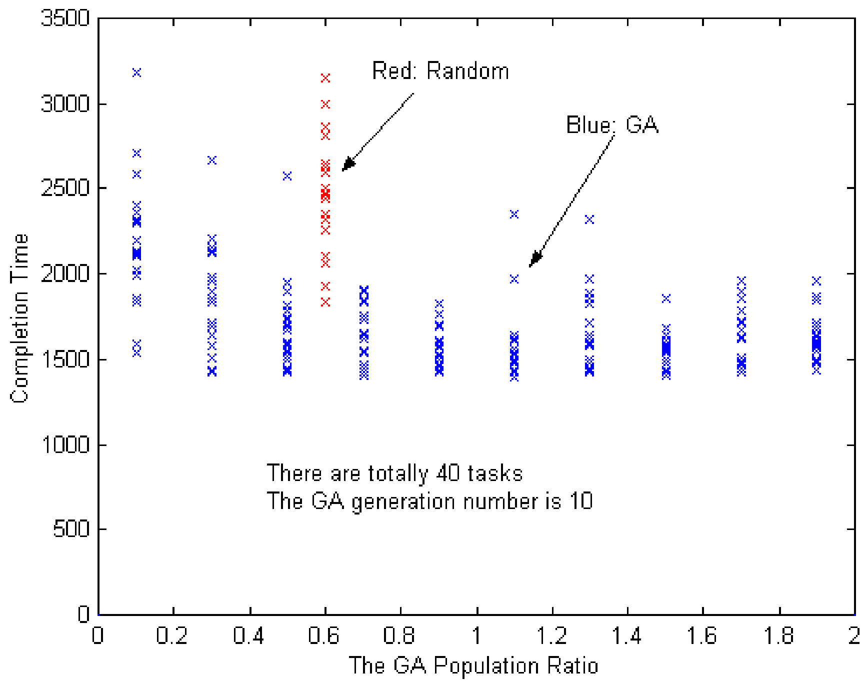
Figure 6.15 Completion Time Performance and Mutation Rate

6.3.8 Comparison with a Random Scheduler

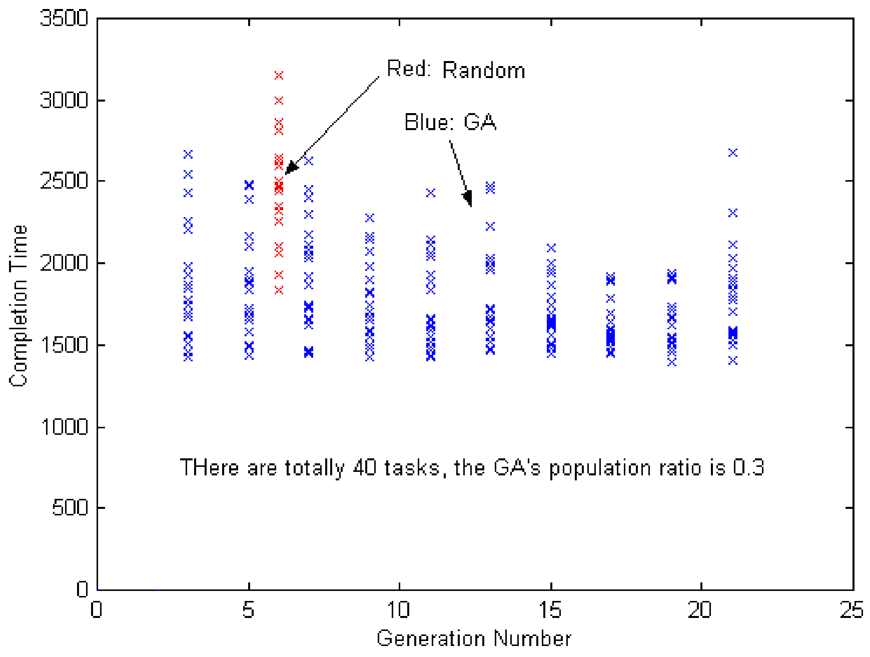
The GA's performance is also compared with a random scheduler. In the random scheduler, the ready tasks allocation algorithm is not the GA-based but a random assignment where each selected resource is chosen with equal probability, irrespective of its busyness. For example, in the simulation scenario shown in Figure 6.4, where 3 available resources exist, the chances are the same for ready tasks to be allocated onto resources 1, 2 or 3.

The simulation results are shown in Figure 6.16. In Figure 6.16 (a), the GA's generation count is set to 10 and the population ratio increased in steps from 0.1 to 1.9 (Horizontal axis), 20 repeats for each population ratio. In Figure 6.16 (b), the GA's population ratio was set to 0.3 and the generation count increased from 3 to 21 (horizontal axis), 20 repeats for each generation number. In both figures, GA demonstrates shorter completion time (vertical axis) and smaller variance compared with the random scheduler allocation.

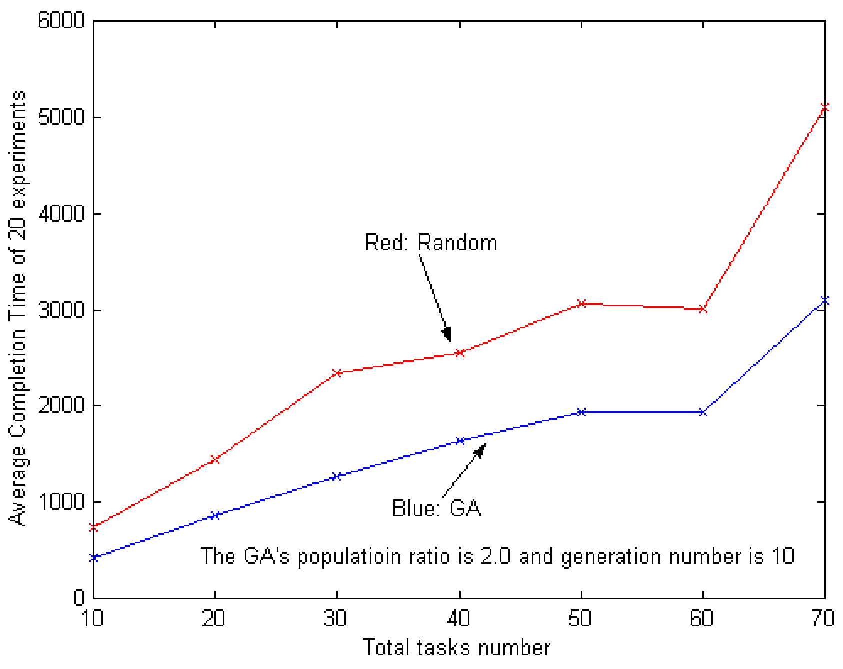
The simulation results also imply that improved Job Completion Time performance can be obtained by increasing the population ratio or the number of generations. However, both of these only provide benefit in terms of the overall job completion time up to a certain point. Further increases give rise to a worsening computation with no obvious benefits. This observation provides useful guidance for configuring the DVM, such that the GA parameters should be chosen in accordance with the job scenario and operator requirements. For example, if there are a large number of jobs waiting for scheduling and a 'quick answer' is required whilst the requirement for an efficient overall job completion time is not very stringent, the DVM might choose either a low population ratio or a reduced number of GA iterations in order to reduce the computation time.


Red: Random
Blue: GA
There are totally 40 tasks
The GA generation number is 10
Completion Time
The GA Population Ratio

(a)


Red: Random
Blue: GA
THere are totally 40 tasks, the GA's population ratio is 0.3
Completion Time
Generation Number

(b)



(c)

Figure 6.16 Performance Comparison of Proposed GA Scheme against a Random Scheduler

In Figure 6.16 (c), the average completion time achieved by GA and the random scheduler are examined in 7 different randomly generated job scenarios. The smallest one has 10 tasks while the largest one has 70 tasks in total. 20 trials are carried out under each job scenario and the average completion time of the 20 trials is recorded. As expected, the simulation results show that the GA achieves a shorter completion time than the random scheduler under these conditions. The simulation results shown in Figure 6.16 (c) indicate that the GA's completion time is about 40 percent less than that for the random scheduler. Indeed, the GA's performance could be further improved by optimizing the GA population ratio and generation number based on the preceding results, such as those presented in Figure 6.12 (a) and (b), respectively. Meanwhile, the performance of the random scheduler varies greatly according to the job scenario. Figure 6.17 shows the performance of GA and random scheduler in a particular scenario where 3 network resources exist and the capacity of the 3 resources are 1, 1, and 4 (The capacities were set to 1,2,3 in the former experiments shown in Figure 6.16). The results show that GA achieves around 70 percents shorter completion time than the random scheduler case.

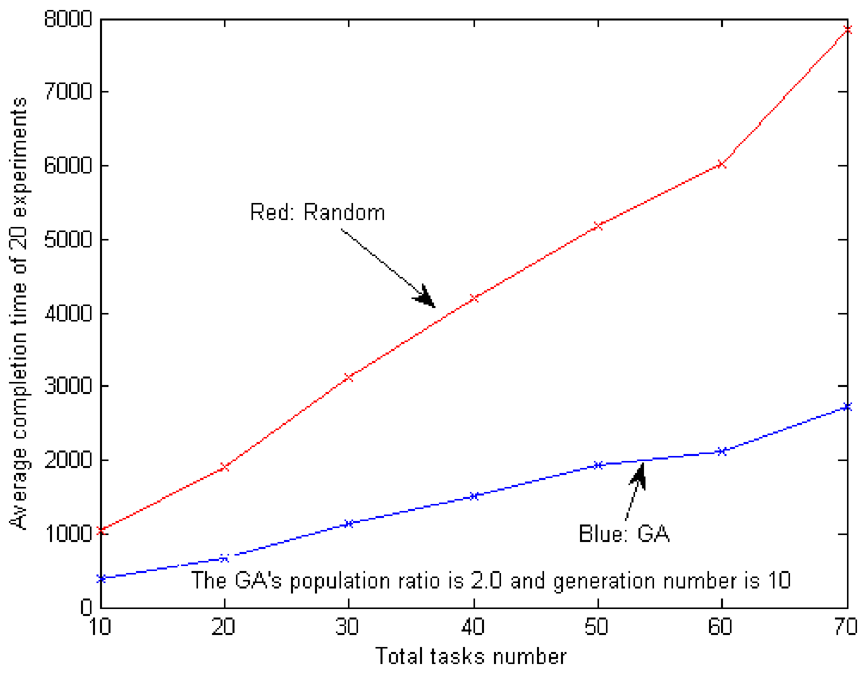


Figure 6.17 Comparison between GA and Random Scheduler in Particular Scenarios

Additional tests, not detailed in the thesis, confirmed that the performance difference between the GA approach and that of a random assignment scheme became more pronounced as the imbalance between processing capability levels of the available resources increased.

6.4 Alternative PSO Optimisation Engine Implementation

The GA scheduler can also be replaced with alternative optimisation algorithms. Particle Swarm Optimisation (PSO) [136] [135] is one of the latest evolutionary optimisation techniques. As analysed in Chapter 2, it simulates the process of a swarm of birds preying and has worked well on many global optimisation problems [138].

6.4.1 Particle Architecture

PSO is employed as the alternative scheduler in this thesis. The overall dynamic resource scheduling algorithm is still setup as in Figure 6.2, while the PSO engine replaces GA as the optimisation tool. The architecture of the particle is similar to the GA's chromosome. A particle has n vectors, where n equals to the number of the tasks waiting for scheduling. Figure 6.18 is a particle example. Each vector maps a certain task and the value implies which resource this task is allocated to. A Job scheduling arrangement shown in Figure 6.4 (b) represents the particle of Figure 6.18 and the mapping relationship between the tasks and resources is demonstrated in Table 6.3. In this example, each element of the vector might be 0,1,2 which maps to available resource1,2,3. As with the GA implementation, PSO particles only imply the relationship among the tasks and resources, the dependences among tasks are considered when the fitness value is calculated.

$\{1,0,2,1,0\}$

Figure 6.18 A Particle Example

Table 6.3 The Relationship between the Tasks and the Resources Represented in Figure 6.18

Tasks to be scheduled	Available resources
Task 1	Resource 1
Task 2	Resource 0
Task 3	Resource 2
Task 4	Resource 1
Task 5	Resource 0

The goal of PSO is also to find the minimum *Overall Completion Time*. How a particle fits this requirement is represented by its *Fitness*. The greater fitness value the less the overall completion time is. The fitness of a particle is calculated as:

$$fitness=C\text{-}max\,(T_1, T_2, \ldots\ldots T_n) \tag{6.5}$$

where C is a large constant number and $T_1, T_2, \ldots\ldots T_n$ means the completion time of resource *n* executing all tasks assigned to it. When the fitness is calculated, the dependences among tasks must be satisfied, a task can start only when all its upper dependent tasks were completed.

6.4.2 Performance Evaluation

The PSO performance as an optimisation engine in the dynamic resources scheduling algorithm is studied with a series of simulations. Three swarm parameters regulate the PSO behaviour:

- C0, Particle's inertia (to keep prior velocity)
- C1, used to update particle's velocity towards global best position
- C2, used to update particle's velocity towards local best position

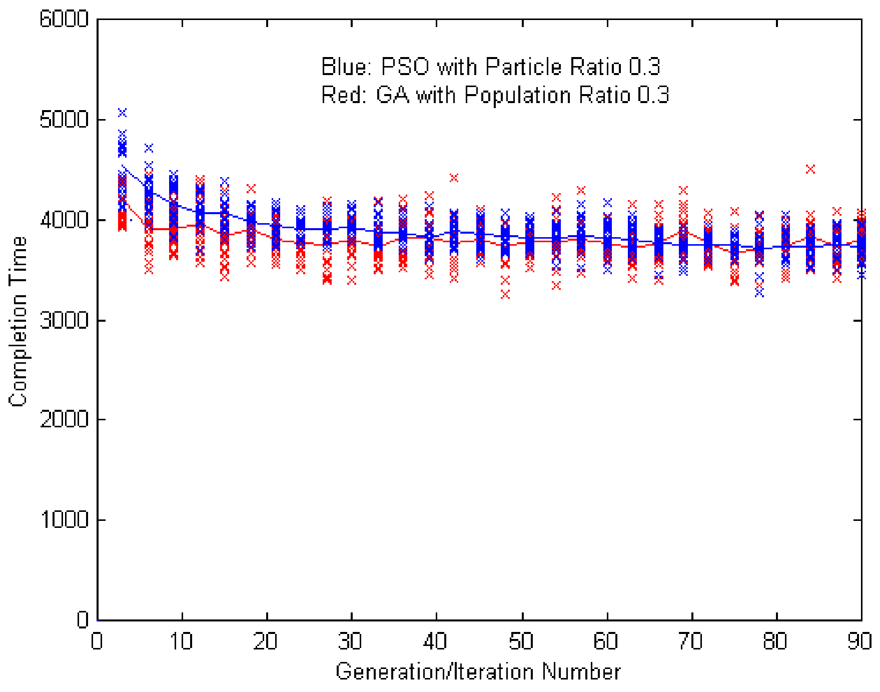
Unless otherwise stated, the three parameters are set as follows in this thesis [137]:

- C0=0.95; C1=0.9; C2=0.9

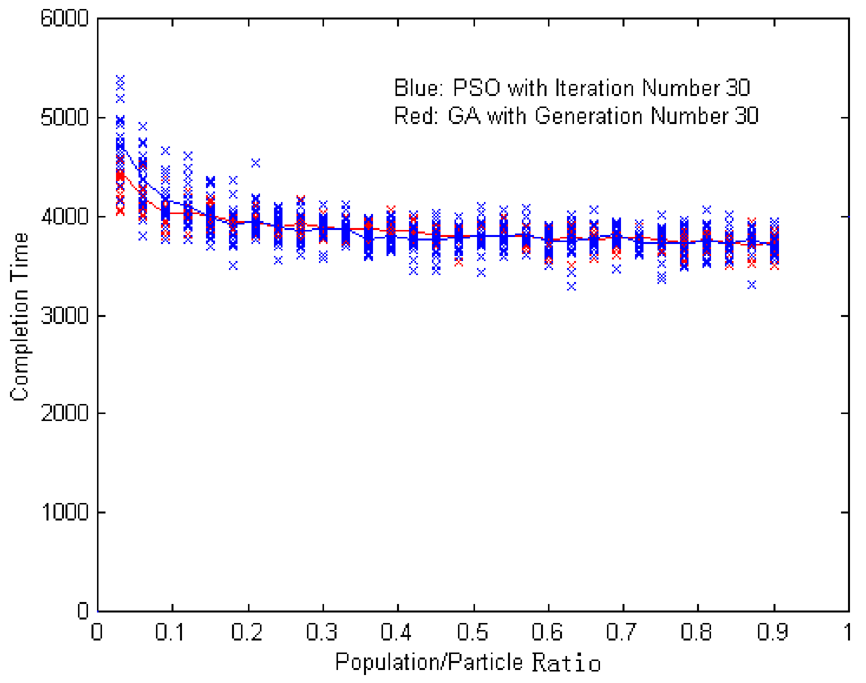
However, the above parameters are also studied in this section.

The author compares the PSO performance against GA as the optimisation engine with the same job scenario. 20 trails are carried out with the same population/particle ratio and generation/iteration number in the same job scenario. This job scenario consisted of 100 tasks and 3 resources, each task has average 5 dependent upper tasks. The PSO also adjusts its particles number according to the current ready tasks number and *Particle Ratio* is defined as the ratio of particles to the number of ready tasks. The simulation results are shown in Figure 6.19.

The PSO approach shows similar performance to the GA scheme. A shorter overall completion time is obtained by increasing the iteration count or the particle ratio over a certain range. Both GA and PSO converge to similar optimal/near optimal solutions (shortest overall completion time) if the population/particle ratio and generation/iteration number are large enough. However, GA can achieve a better solution when the population/particle ratio and generation/iteration number are small.

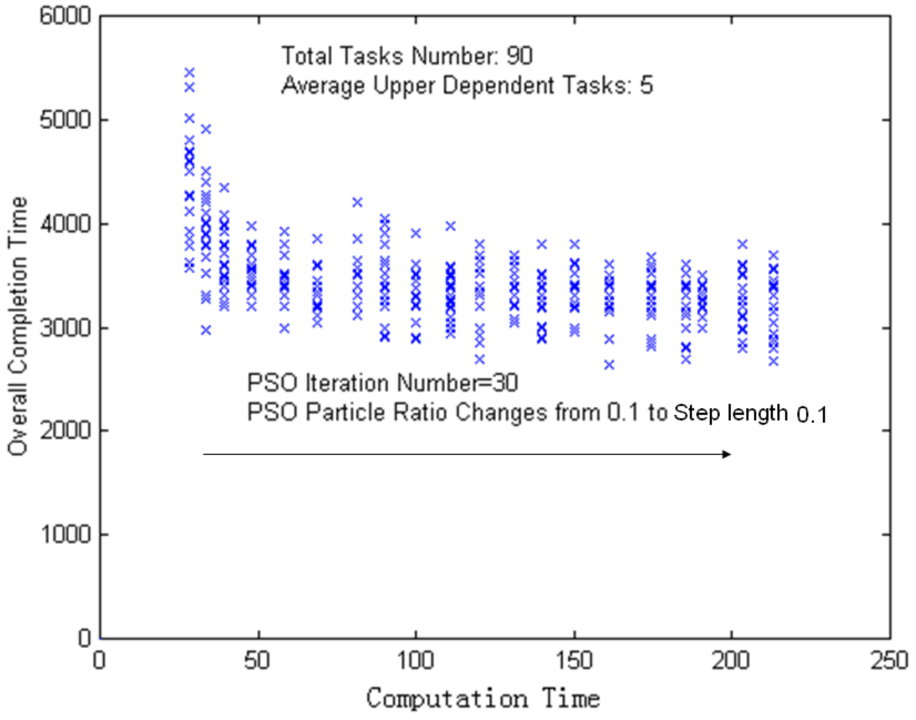


(a)

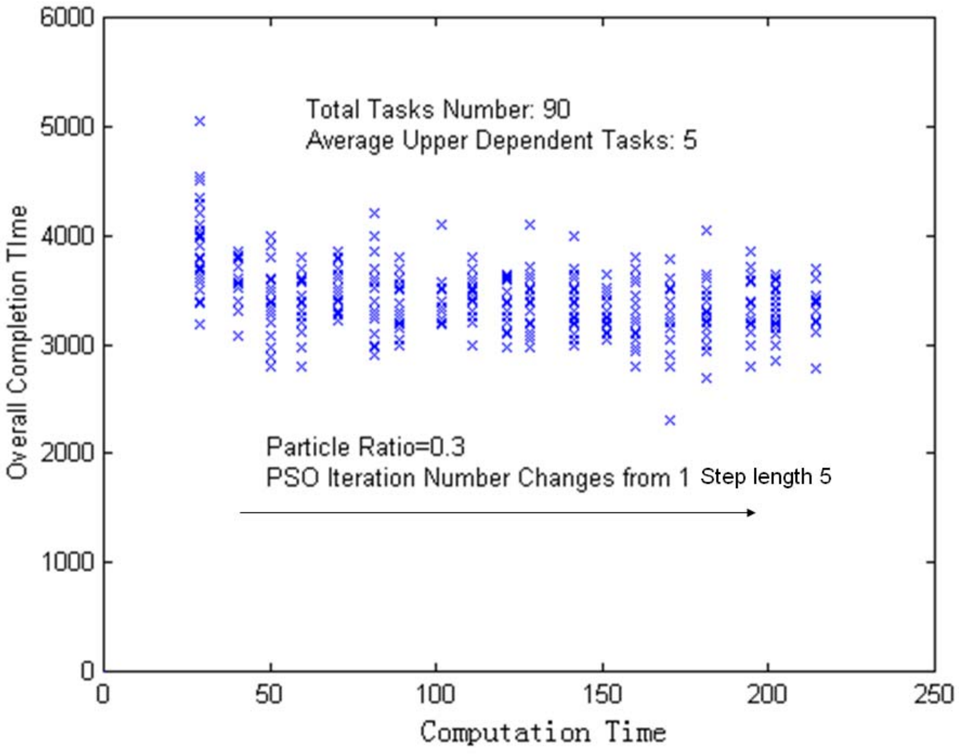


(b)

Figure 6.19 Performance Comparison of Static GA and Static PSO Resource Scheduling Algorithms



(a) Completing time and overall completion time while PSO iteration number=30 and particle ratio changes



(b) Completing time and overall completion time while PSO iteration number changes and particle ratio fixed

Figure 6.20 Relationship Between Completion Time and Computation Time

Considering Figure 6.19, we find that increasing particle ratio and iteration number can improve the completion time performance, although more computation time has to be spent. The relationship between the

PSO performance and particle and iteration number is represented in Figure 6.20. The simulation job scenario contains 90 tasks and each task has an average of 5 upper dependent tasks. 20 trials are carried under same particle ratio and iteration number. In Figure 6.20 (a), the iteration number is 30 and particle ratio changes from 0.1 with step length 0.1. In Figure 6.20 (b), the particle ratio is 0.3 and the iteration number changes from 1 with step length 5. According to Figure 6.20 (a) and (b), the conclusion can be drawn that to improve the performance (shorter completion time and smaller confidence interval) more computation time will be spent on calculating the resource scheduling by PSO. Yet too large a particle ratio or iteration number will result in huge computation times consuming but bring no significant performance improvement.

As the scheduling algorithm must operate in a dynamic environment, the speed of the optimisation engines must also be considered. The computation time of GA and PSO approaches are compared under the same job scenario. According to the simulation results given in Figure 6.21, with same population/particle ratio and generation/iteration number, it can be seen that GA requires considerably more computation time than the equivalent PSO scheme.

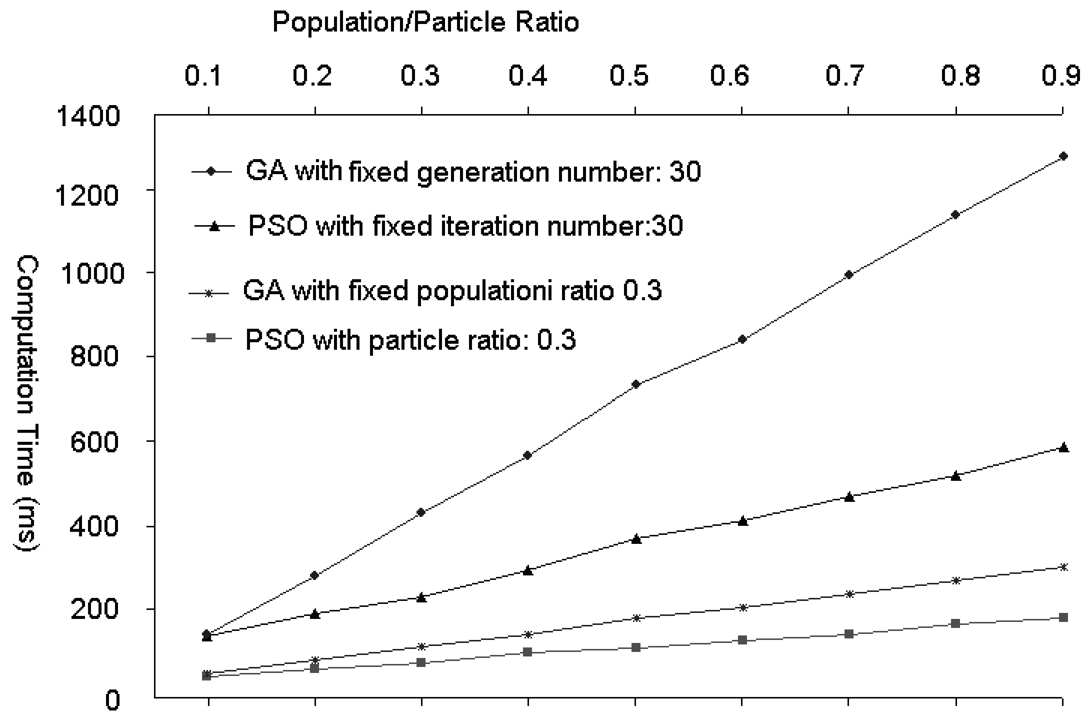


Figure 6.21 Computation Time Comparison between GA and PSO

6.4.3 Swarm Parameters

As stated above, there are three parameters that can affect the particle behaviour in the swarm. The relationship between these parameters and PSO behaviour are studied with simulations where the iteration count is fixed to 10. The results are shown in Figure 6.22, Figure 6.23 and Figure 6.24. For all these simulations; the job scenario contains 90 tasks and each task has an average of 5 upper dependent tasks. 20 independent trials are carried out under the same conditions due to the heuristic nature of the PSO optimisation algorithm.

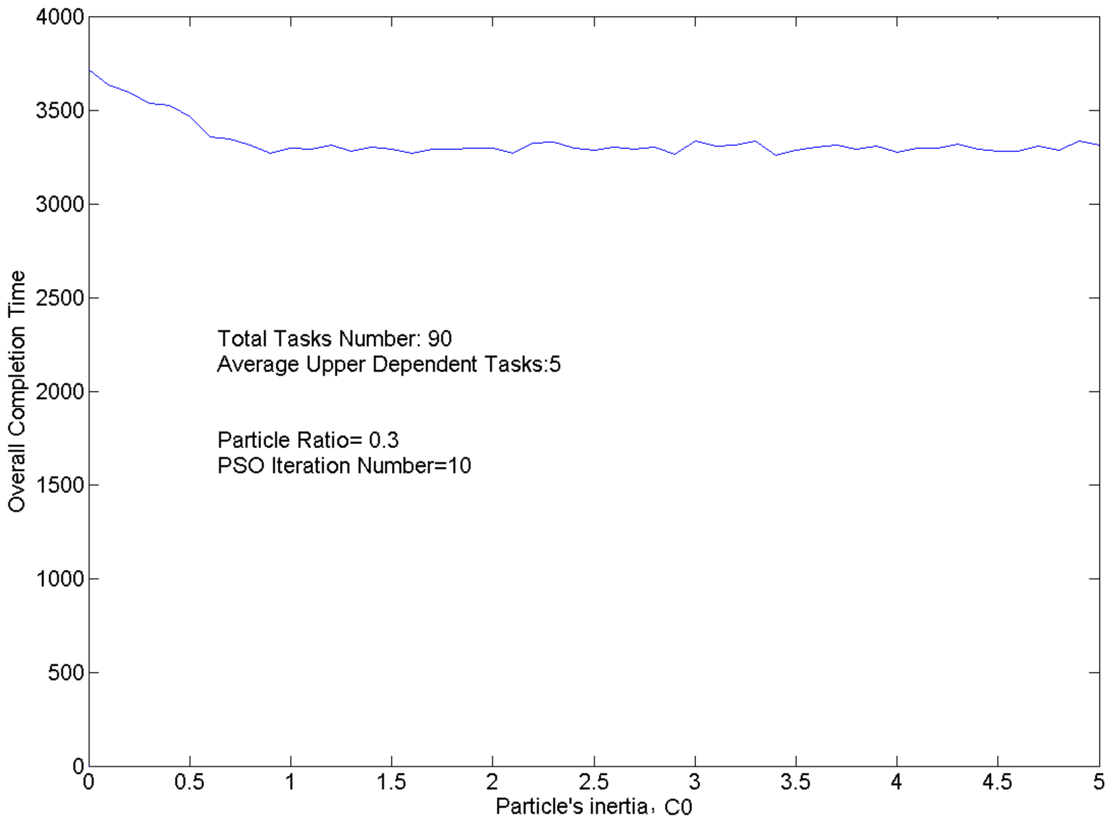


Figure 6.22 Relationship between Particle Inertia (C0) and PSO Performance

According to Figure 6.22, too small a PSO Inertia (C0) leads to worse performance. The reason is that a small inertia means that a particle's current vector is considered little when the particle's new vector is calculated. As a result, the PSO has insufficient "memory" of the results of former iterations. In other words, the algorithm does not depend much on the outcome of former iterations; the former iterations have little effect on the following iterations, which leads to the poor convergence. However, beyond an inertia value of 1, the inertia value is sufficient. Further increases in inertia have minimal effect on the PSO performance[5].

The results shown in Figure 6.23 imply that when the PSO Global Velocity (C1) is set to an appropriate value (i.e. between 2 and 3 in the scenarios considered) good performance can be obtained; meanwhile, too small or large a C1 value will lead to inferior performance. This is because a particle's vector is decided by the current

[5] Given the limited iteration count, increases in inertia above 1 appear to have no effect. However running simulations with a very large inertia, not shown, do yield a deterioration in PSO performance. However, if we run more iterations, the PSO performance can compensate for this sluggish convergence to good results.

particle vector, gbest and lbest. If C1 is very small, the effect of gbest will be too small, which leads to the global best location being considered very little when the next particle vector is calculated. On the other hand, if the C1 is too large, the effect of gbest will be dominant and the local best location will not be considered sufficiently for the next particle vector calculation, which also degrades the PSO's performance.

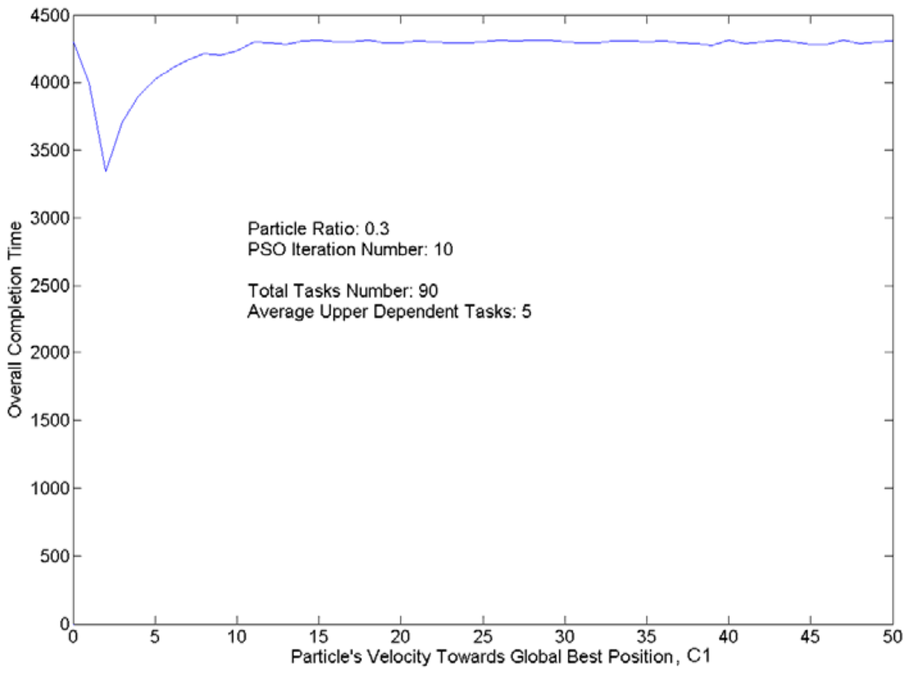


Figure 6.23 Relationship between Global Velocity (C1) and PSO Performance

Similarly, according to Figure 6.24, when the PSO Local Velocity (C2) is set to an appropriate value, a good performance can be achieved. Too large or small a C2 value both lead to an imbalance between the gbest and lbest influence and result in the degraded PSO performance.

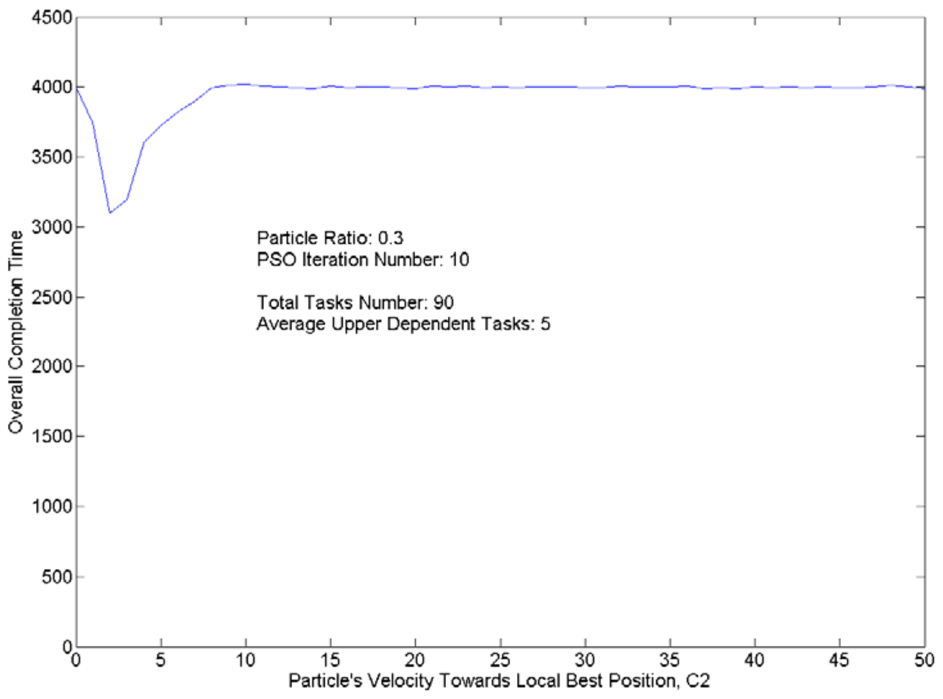


Figure 6.24 Relationship between Local Velocity (C2) and PSO Performance

The performance of the PSO is seen to vary with different swarm parameters setting. The swarm parameters should be set to appropriate value to achieve better performance according to the above simulation results. The optimal PSO parameters may well differ for different scenarios. Appropriate PSO parameters might be obtained by introducing the machine learning technologies; however, this is considered a potential topic for future work.

6.5 Inter-Domain Resource Allocation

As analysed in Section 4.1, the DVPN architecture is designed to provide inter-domain services for orchestrated computing. Through negotiation among DVMs, available resources can be shared across the domain borders.

Normally, using the inter-domain resources will incur extra cost, which is different from the local scenario. To make the dynamic resource-scheduling algorithm proposed in this thesis also applicable to the inter-domains case, the fitness function needs to be extended to consider the inter-domain cost. In the following paragraphs, the inter-domain cost is introduced into the fitness function and the behaviours of the proposed algorithm under multiple domain scenarios are studied through simulations.

6.5.1 Simulation Scenario

The simulations are carried under a two-domain scenario shown in Figure 6.25. This is the same as for former Sections (i.e. Figure 6.4) differing in only that processing resources CPU 1, 2, 3 are located in AS1,and CPU 4, 5, 6 are located in AS2. The capacity of each CPU resource is listed in Table 6.4:

Table 6.4 The Resources

Resource Name	Capacity
CPU1	1
CPU2	2
CPU3	3
CPU4	1
CPU5	2
CPU3	3

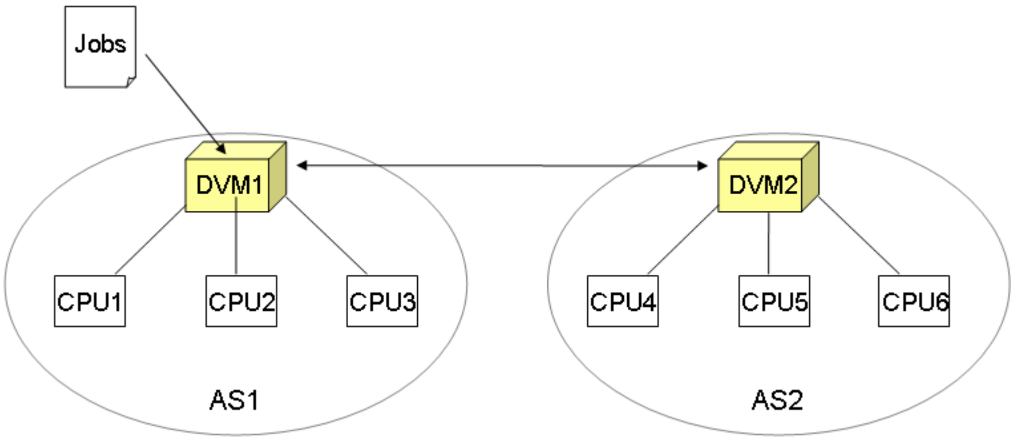


Figure 6.25 Two-Domain Scenario

During the simulation, DVM1 and DVM2 exchange resource availability information together with resource regular usage updates. For simplicity, the jobs are only submitted to DVM1 (the DVM of AS1). Tasks allocated to CPU1, 2, 3 are performed locally; tasks allocated to CPU 4, 5, 6 are performed remotely, which incurs an inter-domain cost. The inter-domain cost might take into account various factors, such as latency, inter-domain charging (fees charged for using resources); however, only the inter-domain charging fee is considered in these simulations and is represented as the cost parameter in the simulation experiments.

6.5.2 Fitness Function

As analysed above, if a task is carried out at a resource outside the local AS domain, an extra fee will be charged. For this reason, the optimal resource-scheduling scheme needs to achieve both a low overall completion time and total cost. The fitness function is therefore extended as follows:

$$Fitness=a*(Overall\ Completion\ Time)+b*(Overall\ Cost)$$

Where *a* is a time factor and *b* is a cost factor. Various values of *a* and *b* reflect how important the *Overall Completion Time* and *Overall Cost* are. For example, the bigger *b* is, the more important the inter-domain cost is regarded by the fitness function.

In the simulation, all the local resources cost 0 units and all the inter-domain resources cost 1 unit / burden. In this thesis, the cost is calculated as

Cost=Resource Capacity x *Occupied Period*

Where the *Occupied Period* is the time that the resource spends carrying out this task.

Another assumption is that all the jobs are submitted to the DVM in AS1, so if a task is carried out at any CPU resource in AS1, the cost will be 0; however, if a task is allocated to a CPU resource of AS2, an inter-domain cost is charged. For example, assuming the burden of a task is 1 the cost will be 1x1=1, if the burden is 10, then the cost for this task is 10x1=10 credit units.

6.5.3 Simulation Results

In the simulation, the total task number is set to 100, the mean task burden is 240 (with normal distribution, variance 100) and each task has 5 uppers on average (normal distribution, variance=1). In each simulation trial, the generation number is set to 30 and the population ratio is set to 1.

In the first simulation experiment, the time factor "a" is always set to 1, the cost factor "b" is set to 0, which simulates the scenario where there is no inter-domain cost. The simulation results are shown in Figure 6.26 10 independent trials are carried out. The total task number located within AS1 and AS2 are the same on average. The conclusion can be drawn that the algorithm treats the local and remote resources the same in this scenario where the inter-domain cost is 0, as expected.

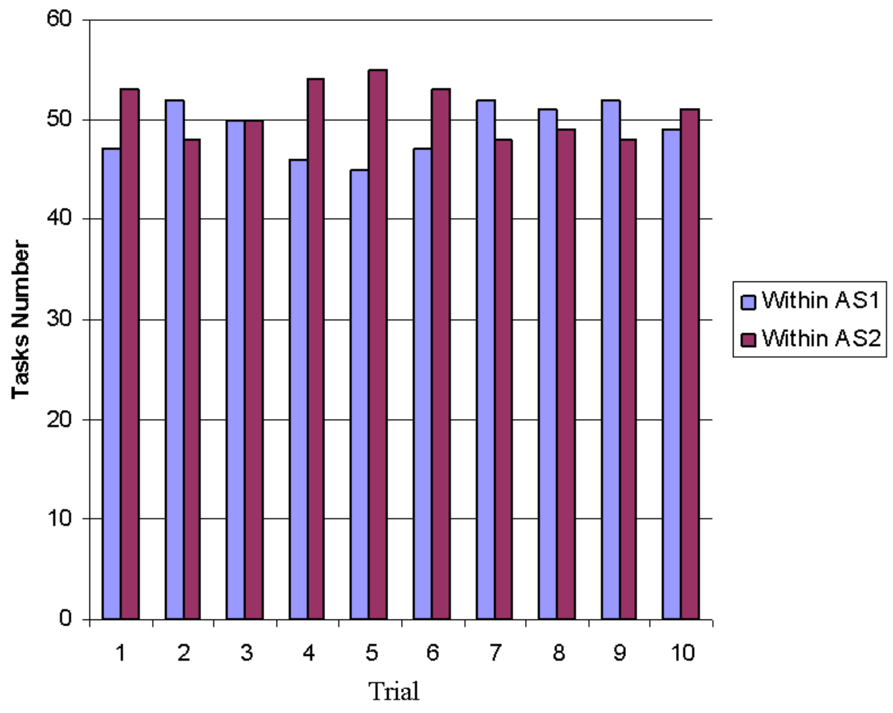


Figure 6.26 Task Distribution in the Scenario without Inter-Domain Costs

In the second experiment, the inter-domain cost is considered. The cost factor b is set to 1. The simulation results are shown in Figure 6.27. As the inter-domain cost is considered when the fitness is calculated, the number of tasks located within AS1 (local AS) is more than the tasks located within AS2. This is because that the algorithm tries to avoid using the inter-domain resources, which incur the extra cost. The average number of tasks is 65.300 for AS1 and 34.700 for AS2 across 10 experiments.

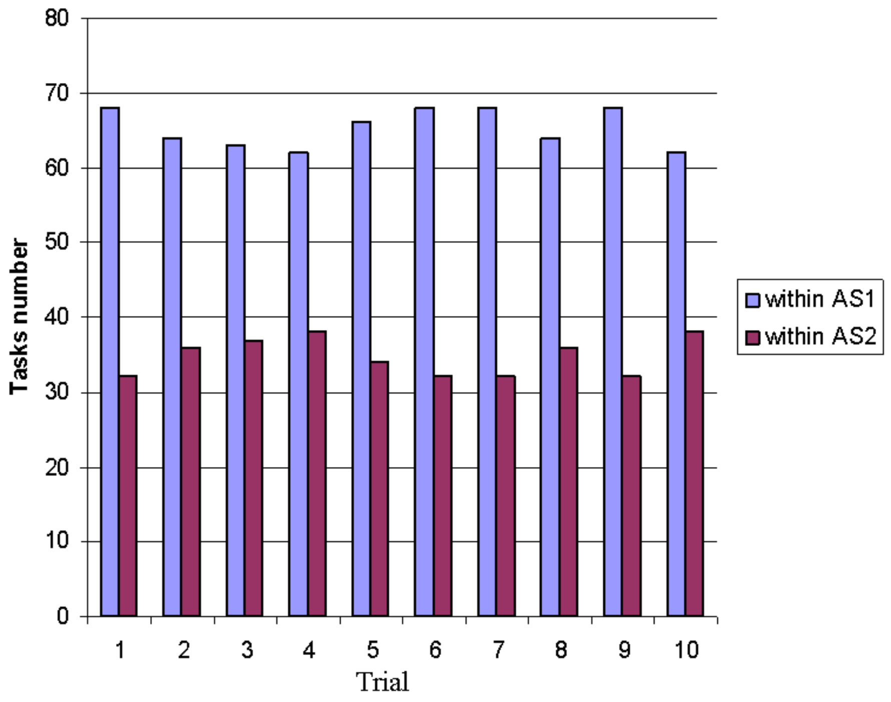


Figure 6.27 Task Distribution in the Scenario where the Inter-Domain Cost is Considered (b=1)

The relationship between the overall completion time and the inter-domain cost factor b is also studied through simulations. Each simulation is repeated 20 times with same cost factor and the average overall completion time is recorded. The results are shown in Figure 6.28. As the cost factor increases, the overall completion time also increases. This is because a higher cost factor makes the cost a more significant factor for the fitness value. The algorithm is more likely to allocate tasks locally when the inter-domain cost is higher. Meanwhile the upper bound of the overall completion time is same as the scenario where only local resources are available, because only the local resources will be employed if the cost factor is very large.

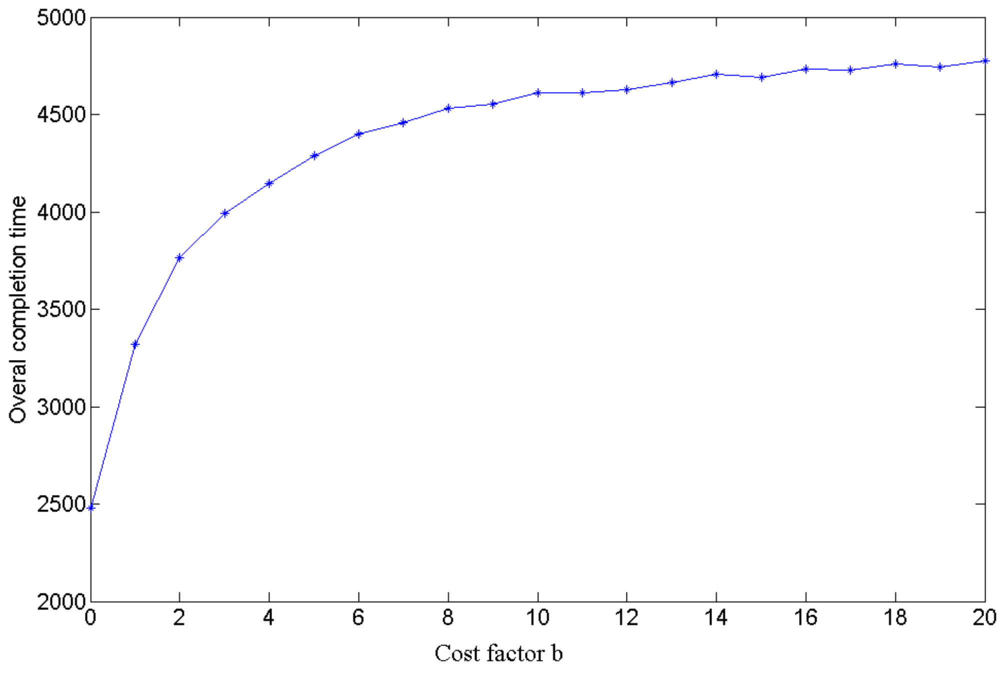


Figure 6.28 Relationship between Overall Completion Time and Cost Factor “b”

According to the above simulation results, the conclusion can be drawn that the dynamic resource-scheduling algorithm is also appropriate for the inter-domain scenario. By introducing the inter-domain cost into the fitness function, a DVPN operator can regulate the extent to which inter-domain resources are employed for a given offered task load relative to the estimated overall completion time.

6.6 Verification

6.6.1 Verification of the Algorithm Architecture

The whole dynamic resource scheduling architecture is verified by tracking the algorithm behaviour in a simulation under a very simple job scenario. The job scenario employed for the verification is show in Figure 6.29. There is only one job consisting of 5 tasks. The number above the each task implies the burden of this task. For example, the T1’s burden is 10, which means that if the task T1 is carried out a resource whose capacity is 1, 10 time unit will be spent to completion this task. In this scenario, there are two resources, R1 and R2. R1's capacity is 2 and R2's capacity is 1.

Because the job scenario is small (only five tasks and two resources), it is feasible to examine all the possible arrangements and pick out the best one in each iteration. Meanwhile, the verification that focuses on testing the whole algorithm architecture and the optimisation engines are not studied here. For this reason, in each algorithm iteration the completion time of each possible arrangement for the ready tasks is calculated and the one with the shortest completion time is picked as the best arrangement.

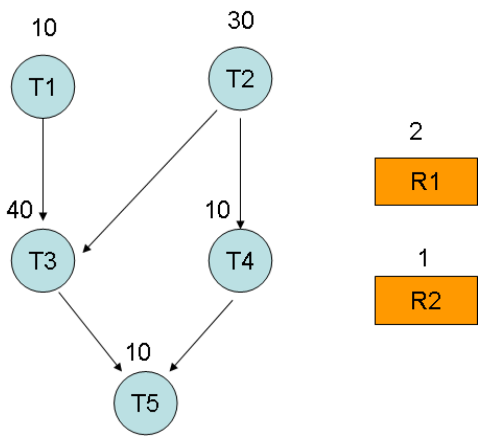


Figure 6.29 verification Job Scenario

The tracking result is recorded in Table 6.5. At the first iteration, T1 and T2 are the ready tasks, R1 is allocated to T2 and R2 is allocated to T1, the R1's completion time is 15 and R2's completion time is 10. At the second iteration, T3 and T4 are the ready tasks and scheduled to R1 and R2 separately. Finally, T5 is scheduled to R1. The overall competition time is 40. These arrangements are illustrated in Figure 6.30.

Table 6.5 Algorithm Process Tracking

Iteration	R1 Completion Time	R2 Completion Time	R1	R2	Completion Time
1	15	10	T2	T1	10
2	35	25	T3	T4	35
3	40	25	T5	--	40

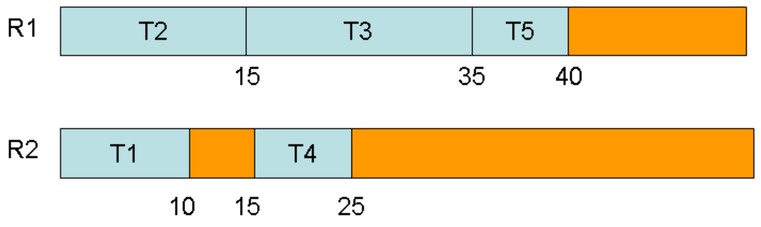


Figure 6.30 Scheduling Arrangement

By tracking the behaviour of the algorithm, it is proven that the ready tasks are picked out and the completion time is calculated correctly at each iteration. The dependency relations are also satisfied when the tasks are allocated to the resources. The algorithm architecture works in accordance with the author's proposal.

6.6.2 Behaviour Verification of the Algorithm for Particular Scenarios

To study the relationship between the job scenarios and the proposed algorithm's performance, simulations are carried in a scenario where the tasks are linked in a single line as 1→2→3→4→5 and each task has only one dependent upper except the first task. There are totally 100 tasks. The dynamic algorithm's completion time performance is compared with that of the static one. To compare with the static algorithm, the proposed algorithm employs a fixed chromosome population size for each simulation. The average of 20 simulations results is recorded as the result for each selected chromosome population size. The results are shown in Figure 6.31. The proposed dynamic algorithm and the static one perform similarly in this job scenario, as expected

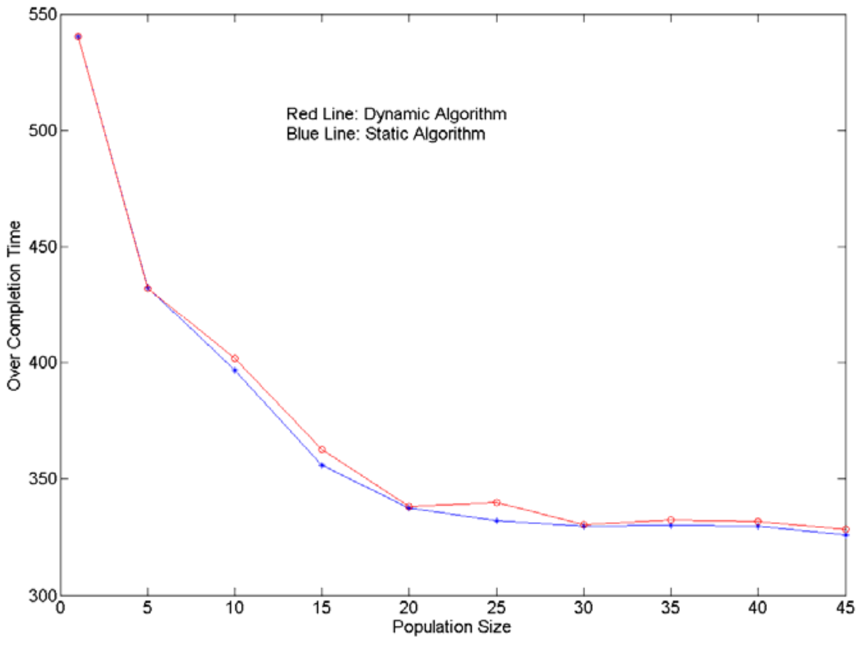


Figure 6.31 Completion Time Performance for a Simple Tandem Task Job Scenario

Figure 6.32 represents the performance of the two algorithms under a scenario where all the tasks are independent, i.e. there are no dependent relationships among the tasks. The total task number is 100. The result shows that both algorithms perform similarly.

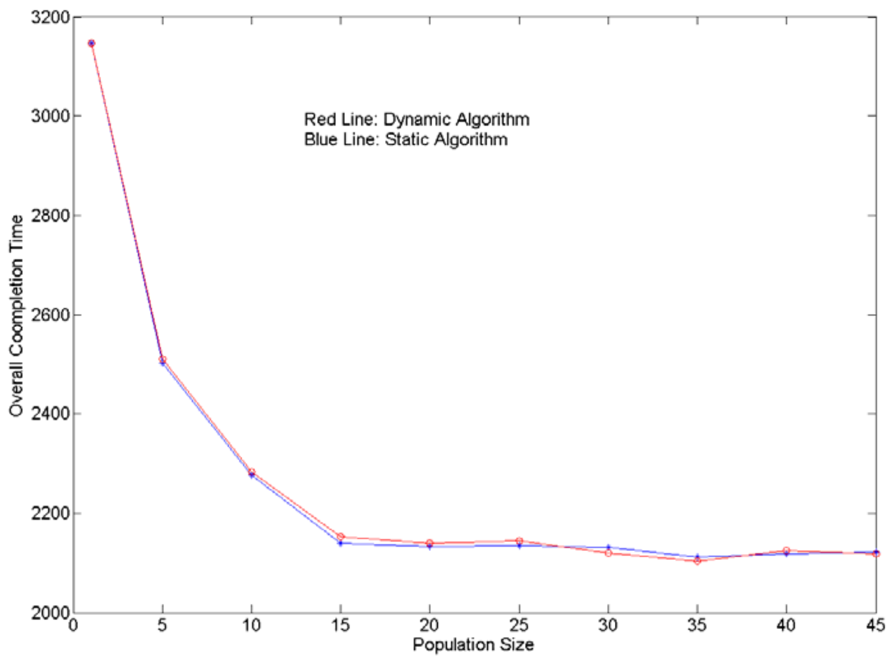


Figure 6.32 Completion Time Performance for an Independent Job Scenario

The results shown in Figure 6.31 and Figure 6.32 represent the performance of the dynamic algorithm under two particular job scenarios. The goal of carrying the simulations in these two scenarios is to examine whether the proposed algorithm performs in the way we expect. One explanation of the similar behaviours of the proposed algorithm and the static one is because that both of these scenarios are "lightly" dependent scenarios. In the Figure 6.31, each task has 1 dependent upper task while in the Figure 6.32 each task has 0 dependent upper task. Based on the conclusion drawn from the Figure 6.9, the proposed algorithm's completion time performance should be similar with the static algorithm when the number of each task's dependent uppers is small, which is verified by the above experimental results.

To test whether the GA performs differently under different task relationships, simulations are carried out using different scenarios. Among these scenarios the total number of tasks are the same, the dependent uppers number of each task and the task's burden, the total layers and the task number in each layer are also the same. Meanwhile, the relationships among the tasks are different. For example task 5 might have dependent uppers task 1 and 3 in one scenario while task 5 might have dependent uppers 2 and 4 in another scenario. In Figure 6.33, simulations are carried out in 8 different scenarios with 40 tasks with same average number of dependent uppers. 20 simulations are carried out for each scenario and the average results are recorded (each scenario is shown in a different colour). The results show that the GA's performance is similar in these different scenarios, so that we can be reasonably confident about the consistency / generality of the previous results and conclusions obtained.

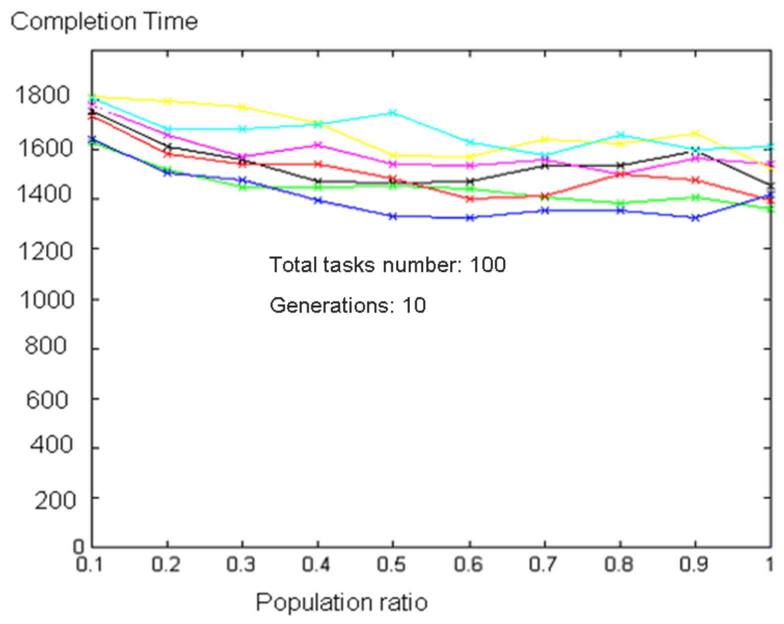
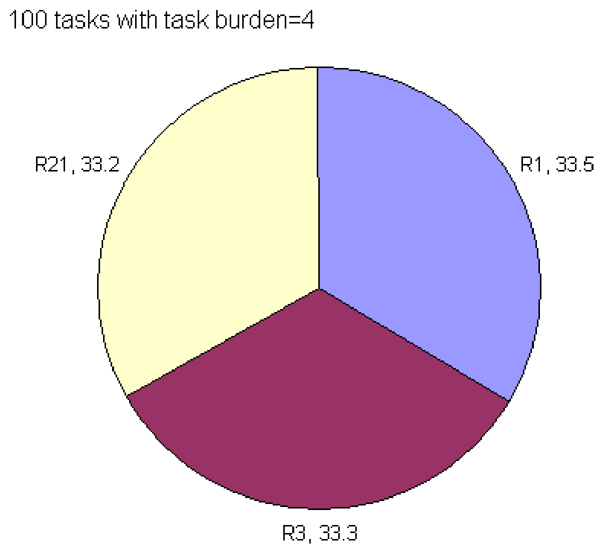
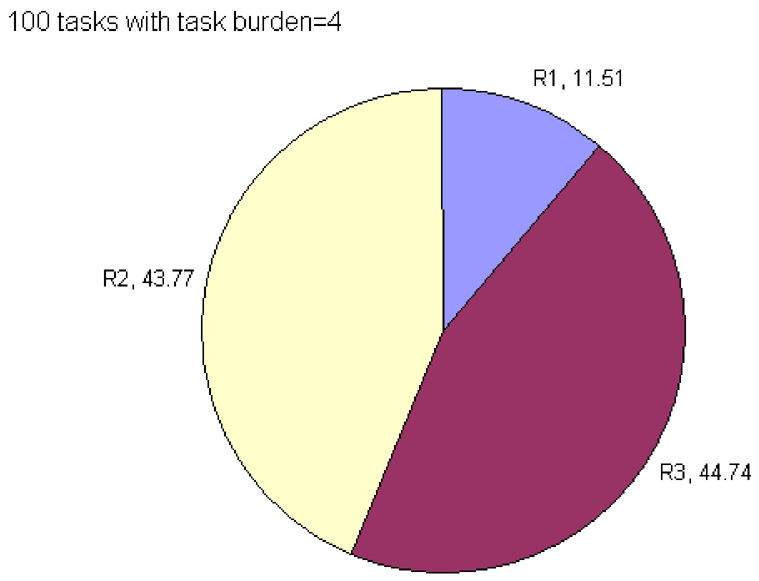


Figure 6.33 Completion Time Performance Scenarios with Different Tasks Relationships but the Same Scenario Size

The algorithm behaviour when the resource capacities are varied is also studied with simulations. Figure 6.34 shows the task numbers allocated to each resource at each simulation. The job scenario contains 100 independent tasks and the burden of each task is 4 units. In Figure 6.34 (a), the capacity of each resource is 1. The 100 simulation results show that the average number of tasks scheduled to each resource are similar. In Figure 6.34 (b), the capacity of resource R1 is 1 while R2 and R3's capacities are 4. The 100 simulation results show that more tasks are scheduled to the resources R2 and R3 whose capacities are bigger and that the tasks are allocated in proportion to the resource capability. This experiment also verifies that this resource scheduling mechanism operating properly.



(a) Mean Number of Tasks: R1=33.5, R2=33.2, R3= 33.3, Resource Capacity: R1=1, R2=1, R3=1



(b) Mean Number of Tasks: R1=11.51, R2=43.77, R3=44.74, Resource Capacity: R1=1, R2=4, R3=4

Figure 6.34 Resource Capacity Verification

6.6.3 Verification of the Random Number Generator

In the simulation studies, the random number generator initially included in Java (java.util.Random class [149]) is employed to produce random numbers. An instance of this class is used to generate a stream of pseudorandom numbers. The class uses a 48-bit seed, which is modified using a linear congruential formula [149]. The system real time is used as the seed for this generator. Figure 6.35 represents the distribution of the 1,000,000 numbers. The results imply that the Java pseudo-random number generator appears to produce the quasi-uniform distributions.

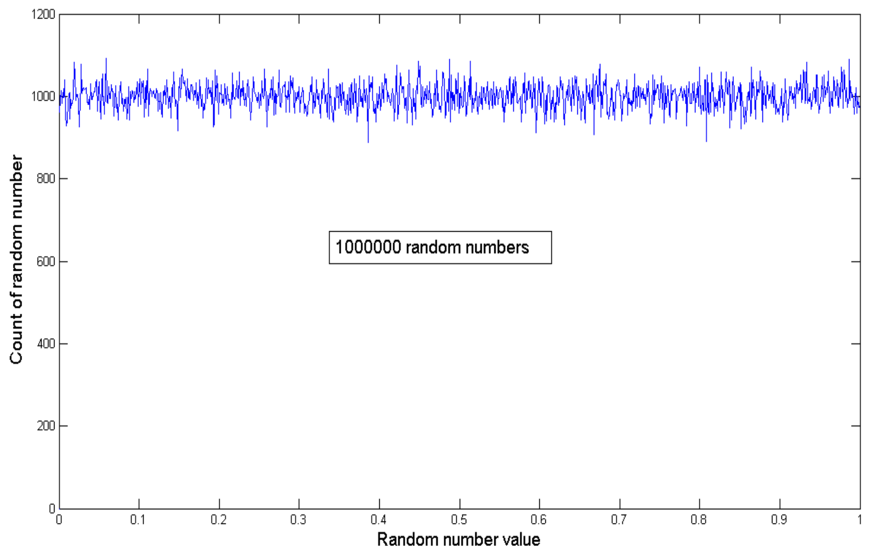


Figure 6.35: Distribution of Random Numbers

6.7 Summary

The DVPN architecture is designed to provide value-added network resources to customers' applications such as Grid Computing. The complex and dynamic DVPN job scenarios require that the network resources to be scheduled dynamically and efficiently. To achieve this a novel dynamic resource-scheduling algorithm is proposed, GA and PSO are employed as the optimisation engines. The algorithms are examined using simulations. The results show that this algorithm is feasible and should operate efficiently within a typical DVPN scenario. The GA and PSO optimisation engines are also compared. Both of them converge to similar overall completion times when a large generation/iteration number and a sufficient population/particle ratio are used. However, with a small generation/iteration number and population/particle ratio, GA can typically obtain better results although the PSO approach requires a much shorter computation time. The author also studied the relationship between the parameters of GA and PSO and the algorithm performance. The results can provide an engineering guideline for parameter value selection.

Chapter 7 The Credit-Based Inter-Domain MPLS Mechanism

7.1 The Credit based inter-AS management Mechanism

Applications such as global grid computing benefit from global resource sharing (i.e. bandwidth, processors). To achieve this, it is expected that cooperation is necessary between Autonomous System (AS) domains. Considering a MPLS VPN service from AS1 to AS3 in the scenario of Figure 7.1, if the direct connection is impossible for this service, an alternative AS-path will be necessary, for example AS1-AS2-AS3. To use this path, AS2 needs to provide a transit service for this VPN link. However, an AS operator normally tends to consider only their own situation and refuses to provide such transit services because the transit service provider does not use this service while its network resources are occupied. As a result, customer service applications will be declined if the transit service is necessary and the overall network resource utilisation efficiency will degrade.

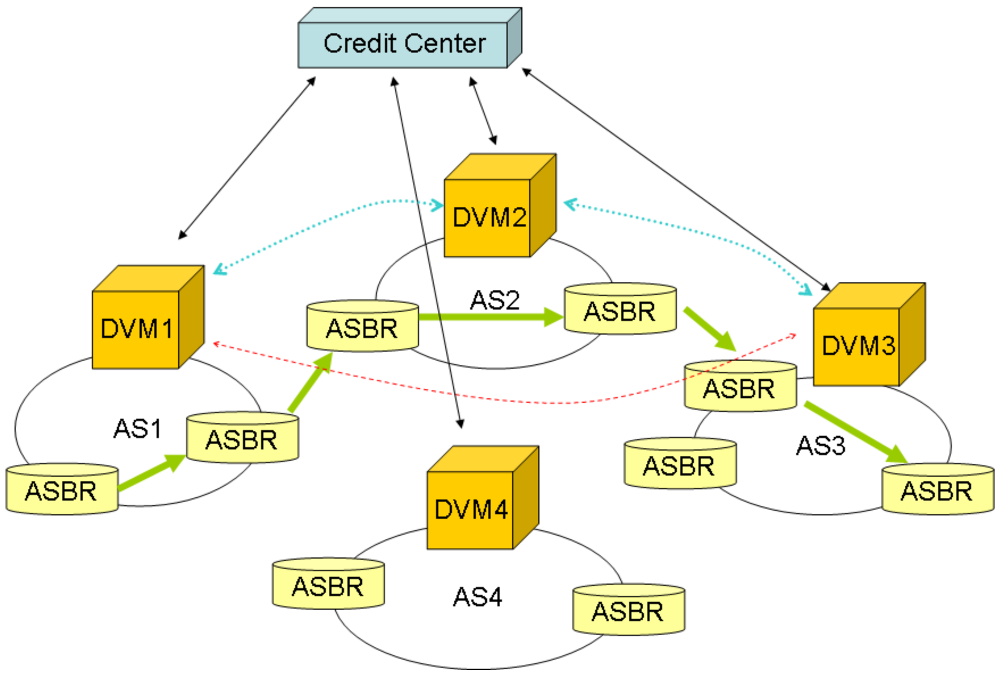


Figure 7.1 Example of How the Credit Based Inter-domain Management Mechanism Works

A mobile ad hoc network (MANET) is a wireless network that has neither fixed infrastructure nor fixed basestations. There is also the “selfish behaviour” problem in MANETs with the Internet. Because ad hoc networks were originally designed to be used in dangerous places such as rescue operations and battlefields, thus most of routing schemes were designed based on the assumption that nodes belong to a single authority. The most prominent routing protocols are AODV [150] and DSR [151], and there are also some routing schemes for load balancing: Load-Balanced Ad hoc Routing (LBAR) [152], Dynamic Load-Aware Routing (DLAR) [153], and Simple Load-balancing Approach (SLA) [154].

Nowadays, however, MANETs are expected to support various civilian applications, and in that situation nodes may not belong to a single authority. Thus, some selfish nodes may not cooperate with others and may attempt not to forward others' control and data packets in order to save their own energy [150]. The selfish behaviours of individual nodes might degrade overall network performance. Several research efforts have focused on how to remove this problem as analysed in section 2.7. In [155] Packet Purse Model (PPM) and Packet Trade Model (PTM) were proposed, where nodes need to pay for packet forwarding in virtual currency named *nuglet*. Since nuglets are required to originate data packets. Nodes have to earn nuglets by forwarding others' packets. Sprite [69] also utilizes credit to give incentives for nodes to forward packets. The Protocol-Independent Fairness Algorithm (PIFA) was proposed [150]. It is also the same type of credit-based scheme as PPM/PTM and Sprite. In PIFA, all nodes periodically send a central server reports about their contribution to packet forwarding, and the server manages credit information for each node after checking the report messages.

The reported results show that the above solutions can effectively induce selfish nodes to participate in packet forwarding, resulting in the prevention of network performance degradation due to selfish behaviour [69] in MANETs. However no similar solution has proposed for inter-domain wire networks.

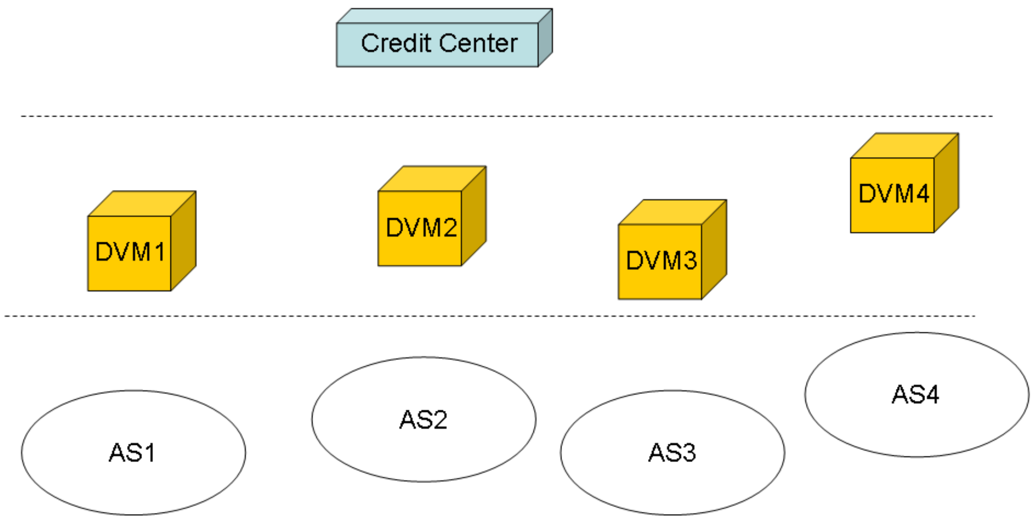


Figure 7.2 The Credit Based Inter Domain Management Architecture

The DVM is the manager of the AS while it also communicates with other DVMs regarding network resources and services. A novel Credit Based Inter-AS Management (CIM) mechanism is designed to work in the DVM based DVPN system to stimulate cooperation among the ASes and a new entity named the Credit Centre is introduced.

In the CIM system, if an AS uses the transit service, its credit will be charged while the credit will be paid to the ASes who provide this transit service. The overall framework of this credit based inter domain management mechanism is shown in Figure 7.2. The CC is the credit management centre equipped with the network interfaces allowing it to communicate with DVMs. The credit of each AS is managed by the CC, the DVM of every AS also keeps its own credit information. When credit payment or charge occurs, the relevant DVMs will update their credits through signalling between themselves and the CC. ASes using the transit service will incur a credit charge. A DVM will decide whether to try to employ such a service or not according to how much credit it has. An AS is not going to use transit service if it has not enough credit. Alternatively, the decision of whether to provide a transit service for other ASes is made by the DVM of the transit AS. If this AS has little credit, it will try to increase its credit by providing transit services, and it will probably refuse to provide transit services if it already has a large amount of credit. In CIM, there are two methods for a AS to get more credit. First, the AS can buy credit with real money. The second, is for an AS to obtain credit by providing transit services for others.

Figure 7.1 demonstrates an example of how this proposed mechanism works. If DVM1 needs to build an MPLS tunnel to AS3, it will try to access this service from DVM3 directly. If it is not possible to establish a direct link to the destination AS, DVM1 will turn to DVM2 (or DVM4). Assuming DVM2 (or DVM4) agrees to provide this transit service, DVM2 will then require DVM3 to establish an MPLS tunnel from AS2 to AS3. If this request is in turn accepted by DVM3, an MPLS tunnel from the ingress in AS1 to the egress in AS3 will be built. Due to the fact that AS2 provides a transit service for AS1, its credits will increase after this service completion in accordance with the QoS parameters (e.g. Bandwidth) and service duration while the AS1's credits will reduce because it has used the transit service. Because AS3 is the termination point AS of the MPLS tunnel, there is no change to its credit status. All the credits are managed by the Credit Centre.

Figure 7.3 demonstrates the credit scheme operating in a scenario with 3 ASes. After service completion, DVM2 will send a receipt to CC to notify CC that it has provided a MPLS VPN tunnel for its upstream AS (AS1) and this is a transit service for the VPN link from AS1 to AS3. The successor of DVM2, DVM3 will also send a receipt to CC for the service it supplied and confirm that its upstream is AS2. DVM1 does not need to send a receipt to CC because it is the consumer of this service.

CC performs the credit payment and charge. With the AS-path information in the receipts, the CC knows which ASes need to be charged or paid. In this example, because AS1 used transit service from AS2, credit will be charged from AS1 and paid to AS2 as AS2 provided transit service. For AS 3, there will be no credit payment/charge for the reason that AS3 is the last AS, it did not use the transit service nor provide any transit service for others. The payment/charge amount is calculated according to QoS requirements and service duration.

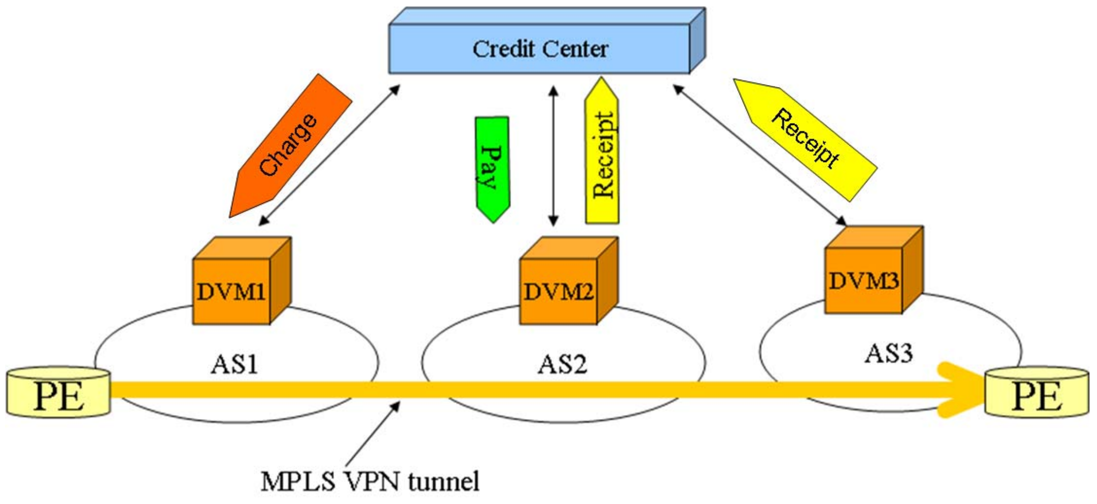


Figure 7.3 The Credit Charge/Payment Process

7.2 Simulation Scenario

This CIM approach has been evaluated with simulations under Matlab [156]. Simulations are carried out in the scenario shown in Figure 7.4. There are four fully-meshed Autonomous Systems (i.e. AS1, AS2, AS3, AS4) in total. In the simulations, the bandwidth, credits and time are represented as integer units.

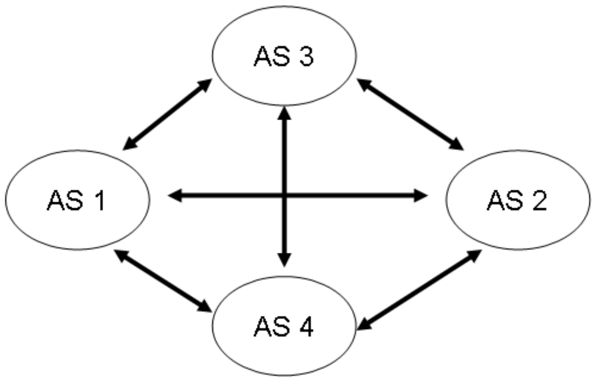


Figure 7.4 Example Simulation Scenario

The bandwidth of all the links among ASes are set to 5 units, the initial credits of each AS are set to 5 units. Each AS generates job applications randomly (i.e. MPLS tunnels with bandwidth requirements and specific durations). The format of a job is shown in Table 7.1. The destination AS is the egress AS of the MPLS tunnel. The bandwidth expresses the bandwidth requirements of a given job and the time is the duration over which this bandwidth will be occupied by this service. Table 7.1 shows an example of an AS building an MPLS tunnel from its own AS to AS3. The bandwidth required is 1.67 units and the duration is 22 time units.

Table 7.1 The Job Format

Destination AS	Bandwidth	Duration
AS3	1.67	22

In the simulation, the destination AS is selected randomly. In the scenario of Figure 7.4, the probabilities are same for each AS (1,2,3,4) to be selected as the destination. If the destination is the AS itself, the DVM will not require any service from other ASes. The bandwidth and duration are generated randomly according to a normal distribution. The mean value of the bandwidth requirement is set to 1 unit and the variance is 1. The mean value of the duration is set to 30 time units and variance is square root of 5.

Firstly, the ingress AS tries to create the MPLS tunnel to the egress AS (destination) directly. If a direct path to the destination AS is not possible, the ingress AS will try to create the MPLS tunnel to the destination AS via transit AS(es), which means that this AS will try to use a transit service. An AS decides whether to accept or refuse a service application based on the available bandwidth between the two ASes and how much credit it has currently.

7.3 Decision Policy

More available bandwidth and fewer credits will make an AS more likely to accept the transit service applications. Two factors affect the decision in this simulation:

- *The current available bandwidth*

  With more bandwidth available, the transit service is more likely to be accepted, because an AS can increase its credits using the unoccupied bandwidth without concerns that its bandwidth is not enough for other services.

- *The current credits*

  The more credits an AS has, the more likely it is to refuse a transit service request, because additional credits bring little obvious benefits since it has a surfeit of them.

In the simulations, 3 decision policies are examined.

a) Decision Based on Exponential Function

In this simulation, an exponential function is employed in the decision policy because this is regarded as a function that simulates certain human behaviours [157]. R, the *Transit Service Acceptance Probability*, reflects the probability of accepting a service application. It is defined as:

$$R= L \times ((\exp(B \times \text{bandwidth})) / (\exp(C \times \text{credits}))) \tag{7.1}$$

Hence

$$R=L \times \exp(B \times \text{bandwidth} - C \times \text{credits}) \tag{7.2}$$

where B is defined as the *bandwidth factor* and C is defined as the *credit factor*. L is defined as the *transit service acceptance factor.*

Figure 7.5 shows the relationship between the Transit Service Acceptance Probability, the available bandwidth and owned credit when L=0.05. B=0.5 and C=0.5

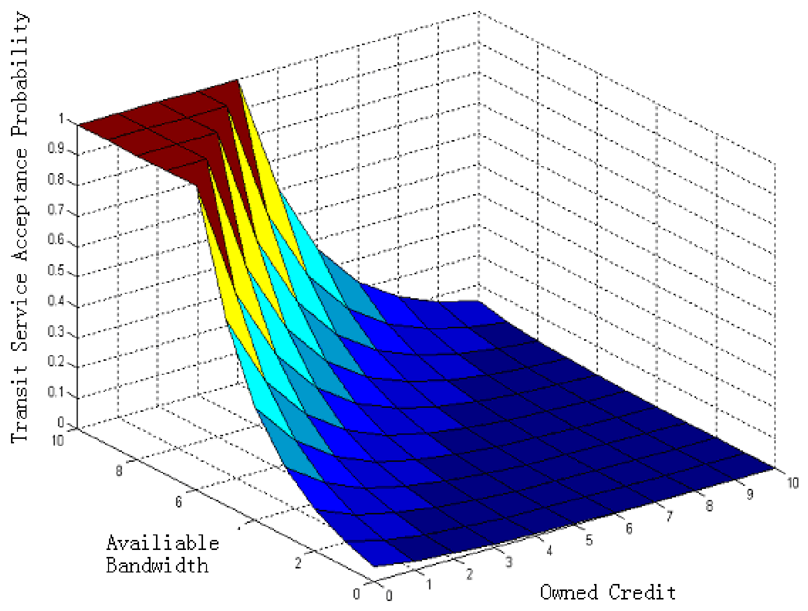


Figure 7.5 Relationship between Transit Service Acceptance Probability, Available Bandwidth and Owned Credit.

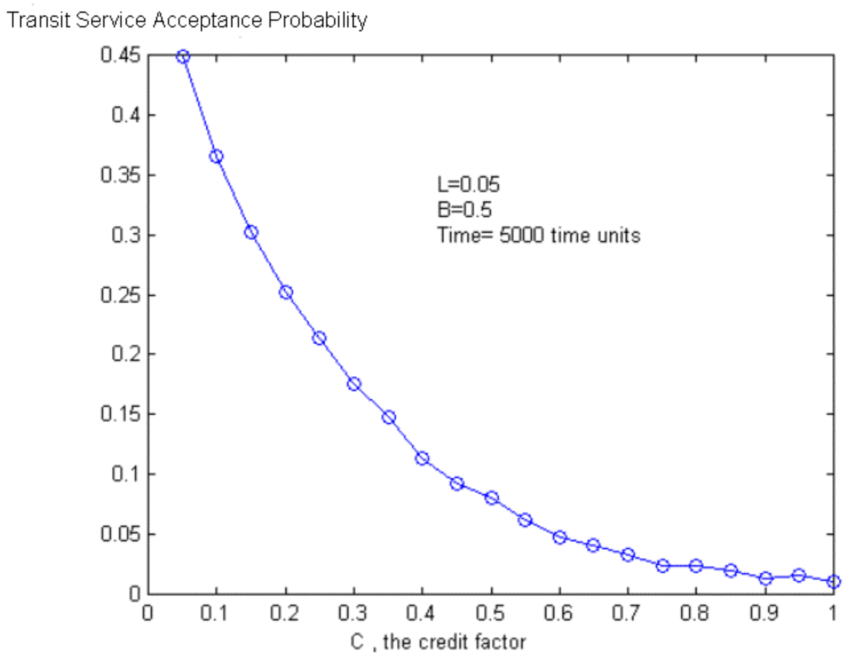


Figure 7.6 Relationship between Transit Service Acceptance Probability and the Credit Factor

The credit factor reflects how the amount of credits affects the AS decision making process, which is examined through simulations shown in Figure 7.6. A bigger credit factor leads to less accepted transit service applications in accordance with equation (7.2). In Figure 7.6, 10 simulations with different credit factors (horizontal axis) are undertaken, where each simulation lasts for 5000 time units. As expected, the results confirm that increasing the credit factor will reduce the Transit Service Acceptance Probability.

Figure 7.7 shows the relationship between the Transit Service Acceptance Probability and the bandwidth factor, B (horizontal axis). Increasing B causes the DVM to be more likely to accept transit service applications.

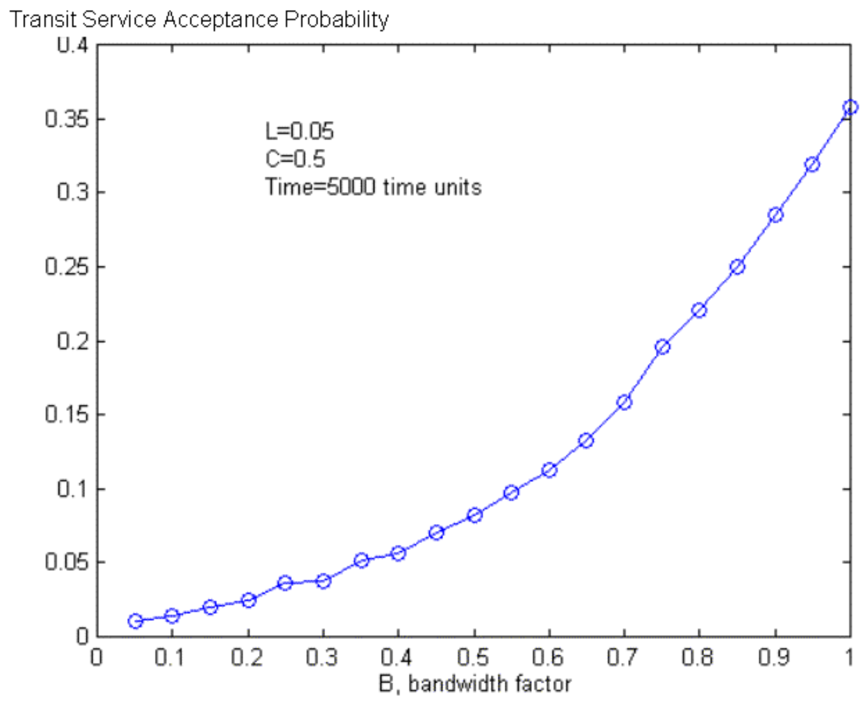


Figure 7.7 Relationship between Transit Service Acceptance Probability and the Bandwidth Factor

Increasing the transit acceptance probability means more transit service applications will be supported, which should lead to an increase in the bandwidth utilisation across the whole network.

b) Decision based on bandwidth and credit limits

The decision scheme is represented in Table 7.2 where *LB* is the bandwidth low bound, *UC* is the credit upper bound and *LC* is the lower bound and UC>LC. If the available bandwidth is lower than *LB,* the DVM will refuse to provide the transit service to protect itself from resource shortage. Meanwhile, if a DVM has more than *UC* credit, it will also refuse any transit service application since it has enough credit. The DVM will only agree to provide the transit service under the condition where the *Credit <=LC* and *Available Bandwidth>LB.*

The relationship between the credit and available bandwidth is represented in Figure 7.8. A similar decision policy is used in [69] for a MANET. In [69], “a node is power-conservative if its remaining power allows it to send (and forward) only a limited amount of messages; a node is credit-conservative if it refrains from sending any new message when its credit balance is insufficient to cover the charge for sending a message.” The author’s simulation results prove that it is an effective decision policy. For this reason, the author also studies this decision policy.

Table 7.2 The Decision Scheme

Bandwidth & Credit	Decision
Available Bandwidth<=LB	Refuse
Credit>UC	Refuse
Credit<=LC & Available Bandwidth>LB	Accept

Figure 7.8 gives the transit service acceptance probability distribution of the decision policy *b)* (UC=3, LC=2 and LB=3).

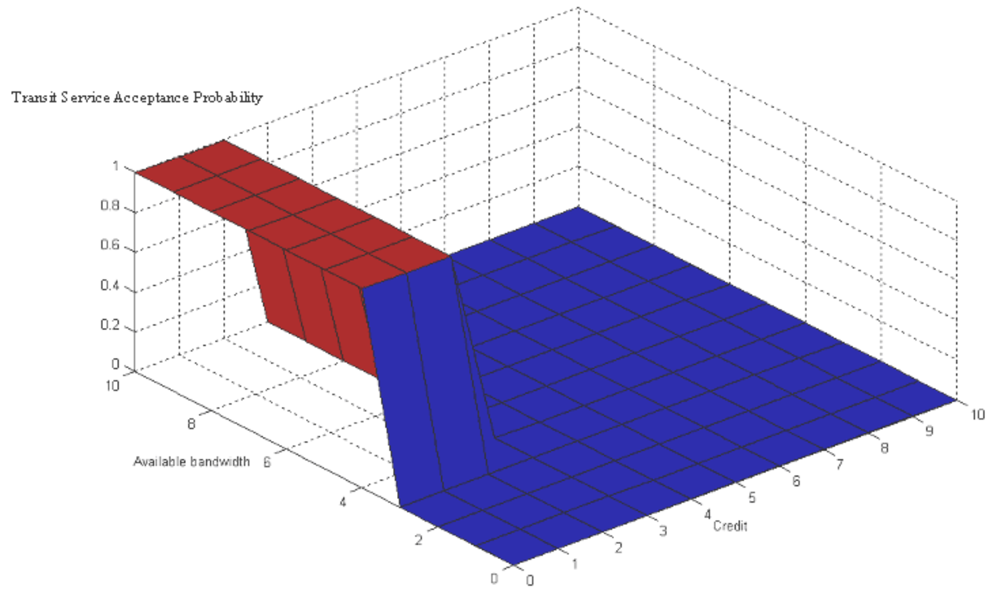


Figure 7.8 Relationship between Transit Service Acceptance Probability, the Credit and Available Bandwidth

Increasing the LB will reduce the transit service acceptance probability. Figure 7.9 shows the relationship between the transit service acceptance probability and the bandwidth lower bound (LB).

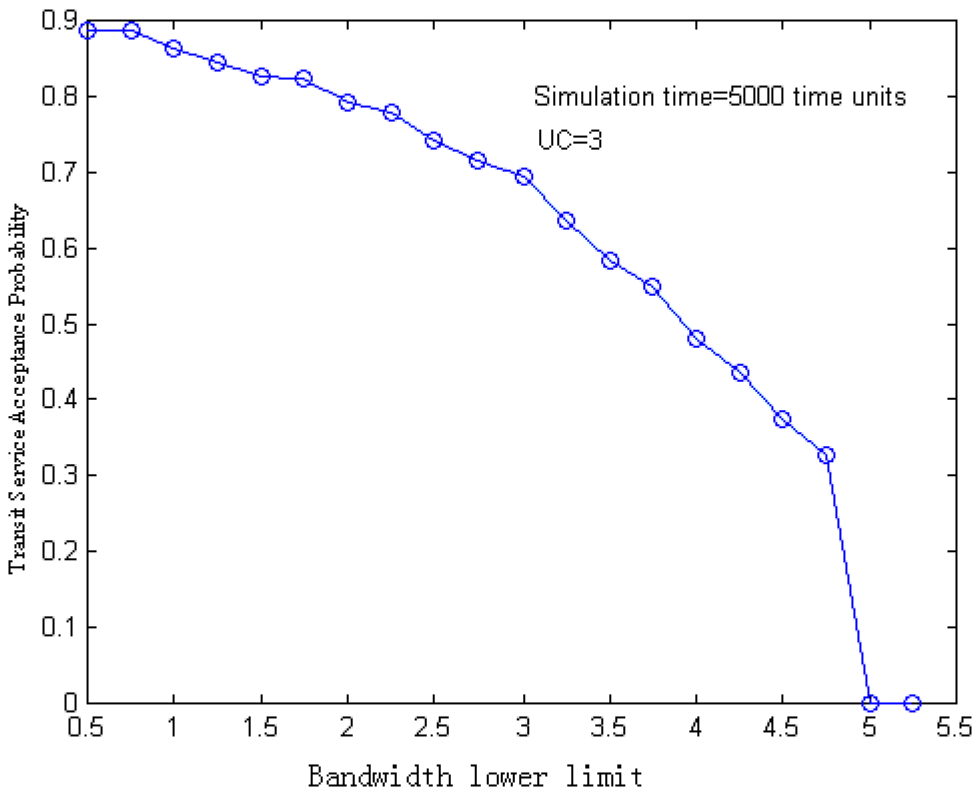


Figure 7.9 Relationship between Transit Service Acceptance Probability and Bandwidth Lower Limit

At the same time, the higher the credit upper limit (UC) the bigger transit service acceptance probability is. Figure 7.10 shows the relationship between the transit service acceptance probability and the credit upper limit (UC).

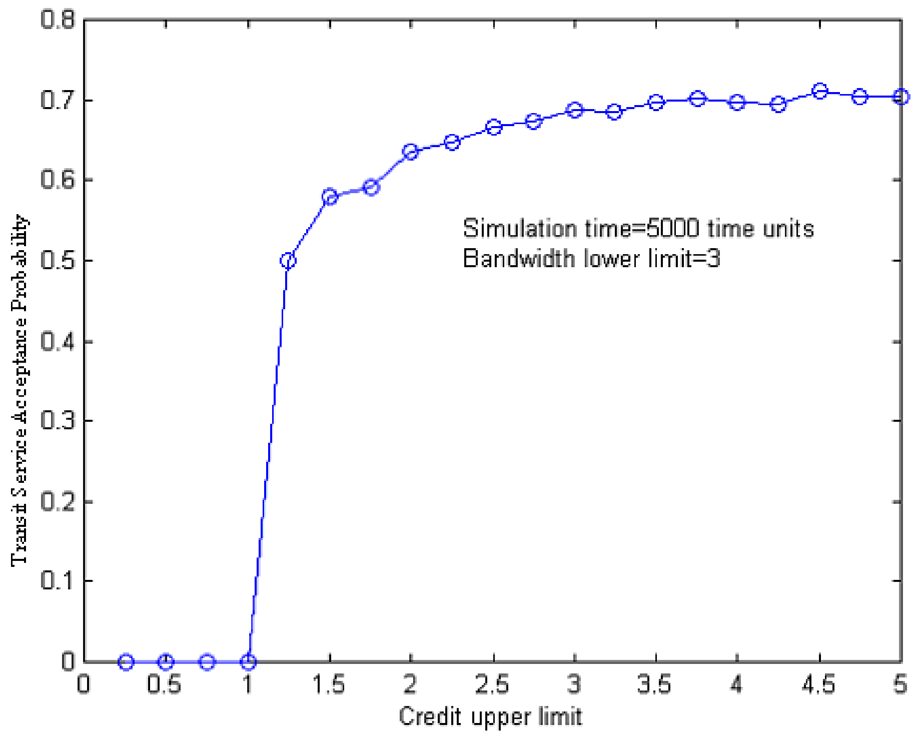


Figure 7.10 Relationship between Transit Service Acceptance Probability and Credit Upper Limit

c) *Fixed transit service acceptance probability*

If the fixed acceptance transit service acceptance probability is employed, the transit service will be randomly accepted with a constant probability when the available bandwidth is enough. For example, if the transit service acceptance probability is set to 0.1, 10 percent of the transit service applications will be accepted if there is the enough bandwidth. The credit does not affect the DVM's decision under this policy and the system works as a normal multiple domain wires network.

7.4 Performance Analysis

Increasing the transit service acceptance probability means more transit service applications will be approved, which should lead to an increase in the bandwidth utilisation across the whole network. Figure 7.11 illustrates the relationship between the transit service acceptance probability and the mean available bandwidth. A reduced mean available bandwidth means more bandwidth is in use, which is generally considered desirable in this scenario.

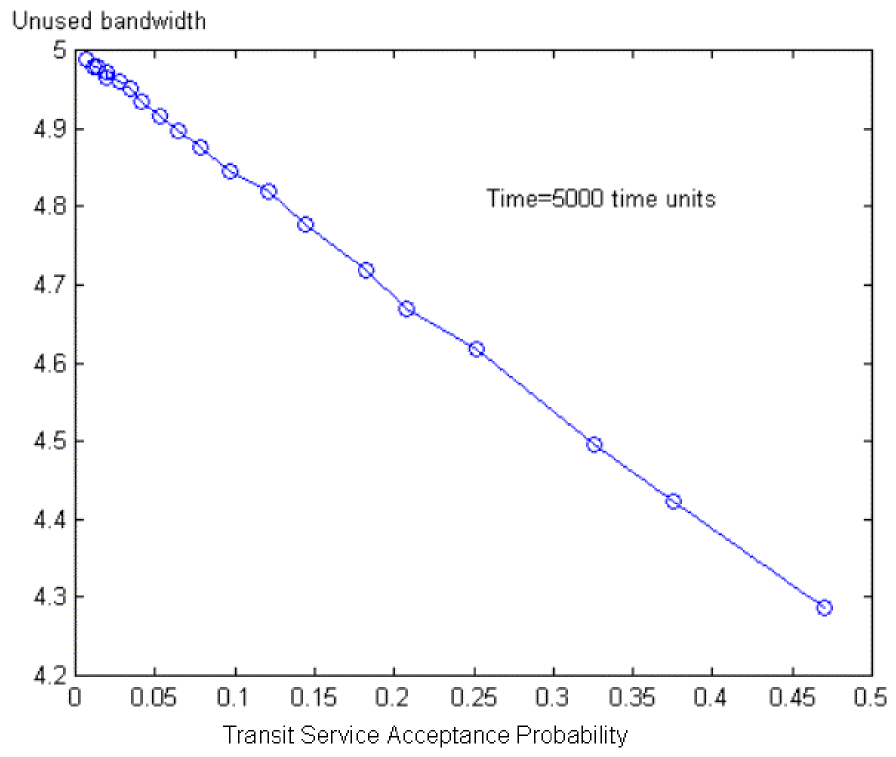


Figure 7.11 Relationship between the Mean Available Bandwidth and the Transit Service Acceptance Probability

The AS will try to use a transit service if its attempt to create MPLS tunnel directly to the destination AS fails. If the Transit Service Acceptance Probability is 0, the network acts as a traditional multi-AS scenario where the credit mechanism does not operate and no AS provides transit services between other ASes due to its selfish behaviour. In Table 7.3, simulation results are listed for a traditional network and the network with the credit mechanism operating. The bandwidth usage is 21.9% for the network with the Transit Service Acceptance Probability of 0.5448 whilst the traditional network only provides 17.68% bandwidth usage.

Table 7.3 The Bandwidth Utilization under Different Transit Service Acceptance Rates

Transit Service Acceptance Probability	Bandwidth Utilization
0.5448	21.9%
0	17.66%

The credit mechanism's performance is also studied under different Direct Connection Probabilities. The Direct Connection Probability is defined as the probability of a direct connection being available. The exponential decision policy (Policy a) and the random decision policy (Policy c) were simulated. (Policy b is not studied at this point. The reason why Policy b is also introduced in the former section is to show that different policies could be employed in this proposed architecture). During the simulation of Policy a, the L=0.05. B=0.5 and C=0.5. The Transit Service Acceptance of the Policy c is set to 0.4.

The simulation results shown in Figure 7.12 indicate that when Policy a was employed, the Transit Service Acceptance Probability will increase in the credit-based architecture if the Direct Connection Probability decreases, which means that the credit architecture provides some compensation when direct connections are not available. However, the random decision approach (Policy c) does not provide such benefit.

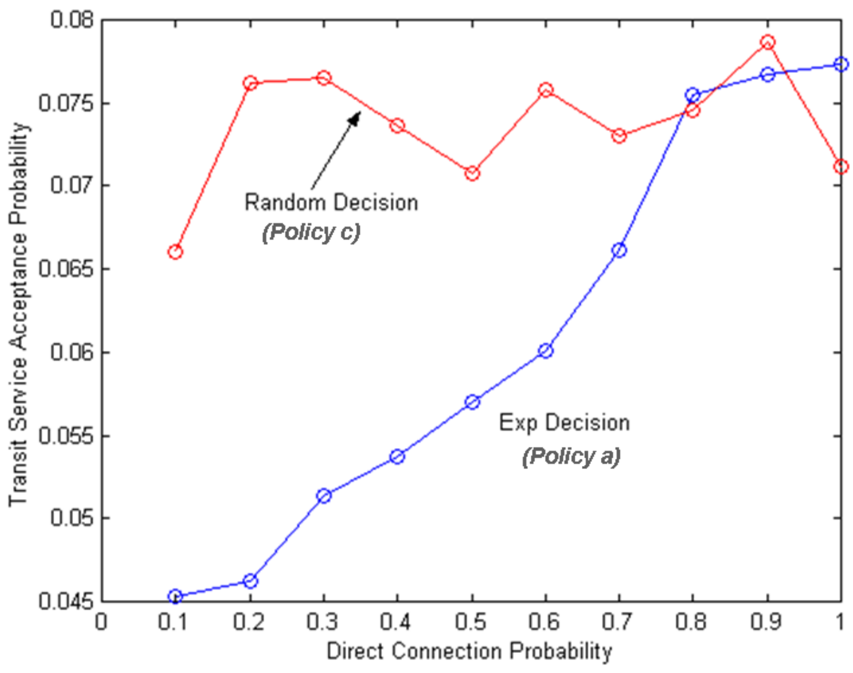


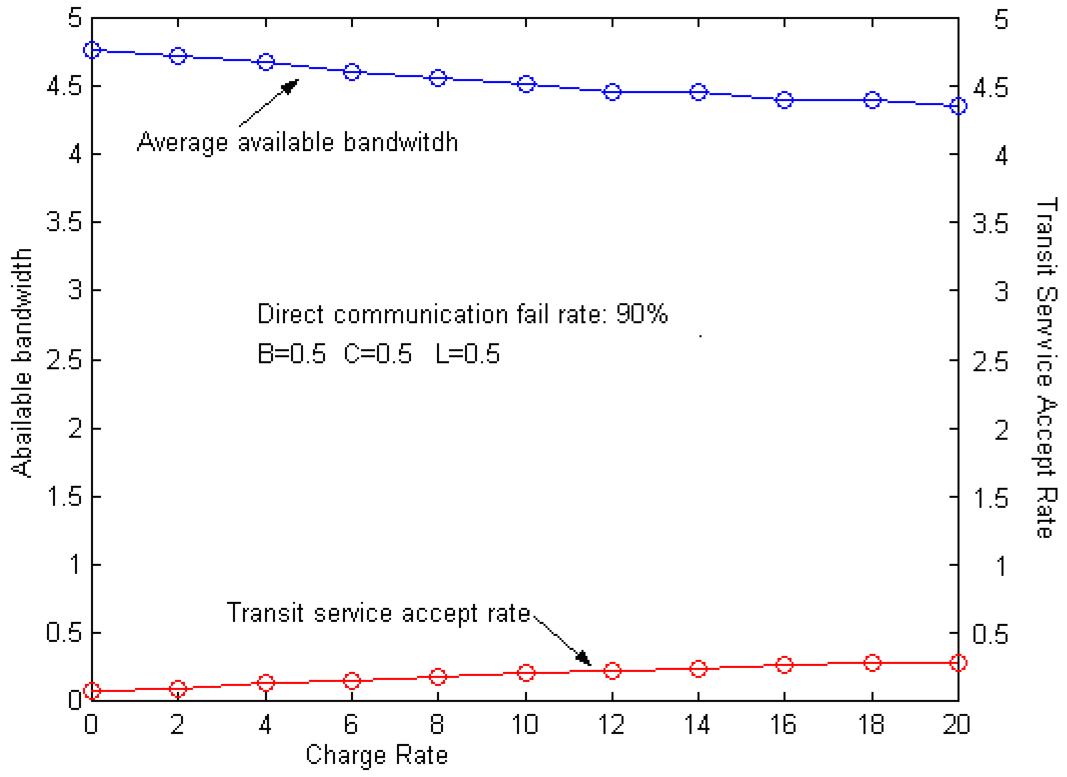
Figure 7.12 Relationship between Transit Service Acceptance Probability and Direct Connection Probability

7.5 Payment/Charge Rate

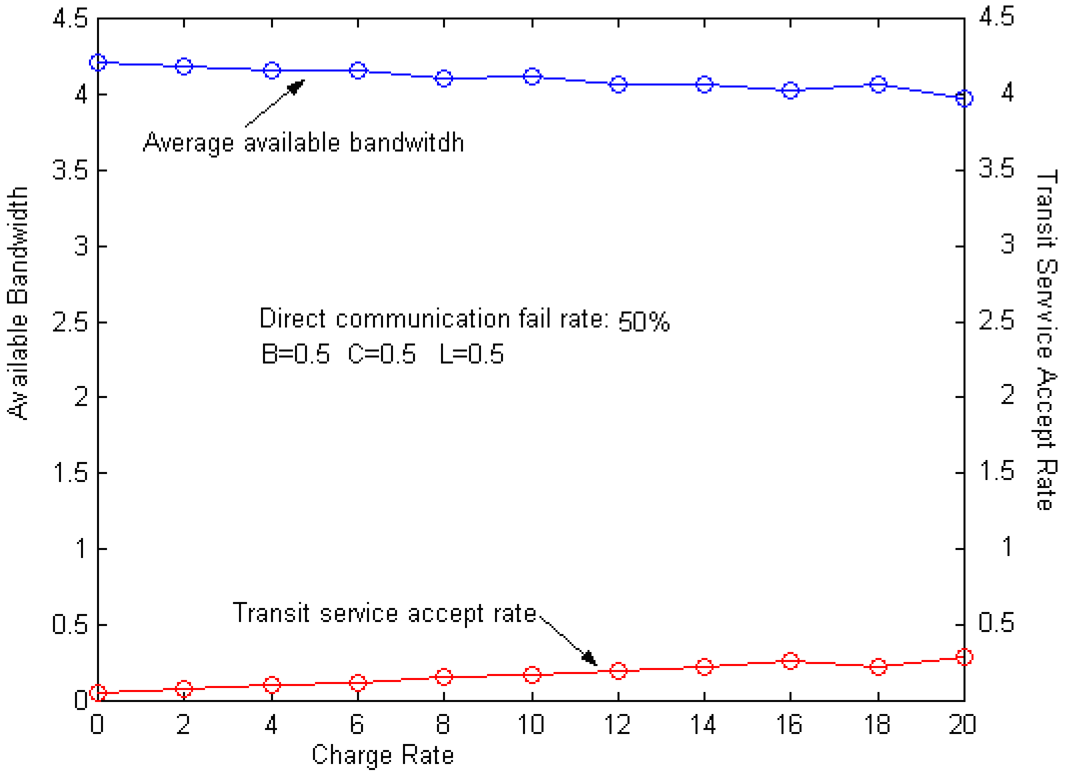
The Credit Centre can also affect the network behaviour through changing the payment/charging rate. The payment/charging rate is defined as how much credit will be paid/charged for per transit service *bandwidth x time*. Simulations were carried out with different payment/charging rates for the same scenario and decision policy under different Direct Connection Probabilities. The Policy a was employed and B=0.5, C=0.5, L=0.5 for all the simulations. The direct connection fail rate is 90 percent in the Figure 7.13 (a). The result indicates that, the Transit Service Acceptance Probability and the Bandwidth Utility both improved while the payment/charging rate increased. In other words, the Credit Centre has the ability to control the extent to which transit services are used across the overall network by simply adjusting the payment/charging rate.

Simulations were also carried with the 50 and 10 percents direct connection fail rate shown in Figure 7.13 (b) and (c). The results suggest that increasing payment/charging rate can improve the Transit Service Acceptance Probability under high Direct Connection Probabilities while the bandwidth utility will not improve significantly. The explanation is that most of the bandwidth usage came from the direct connection and the transit service only occupied a small portion of the total bandwidth. As a result, increasing the Transit Service

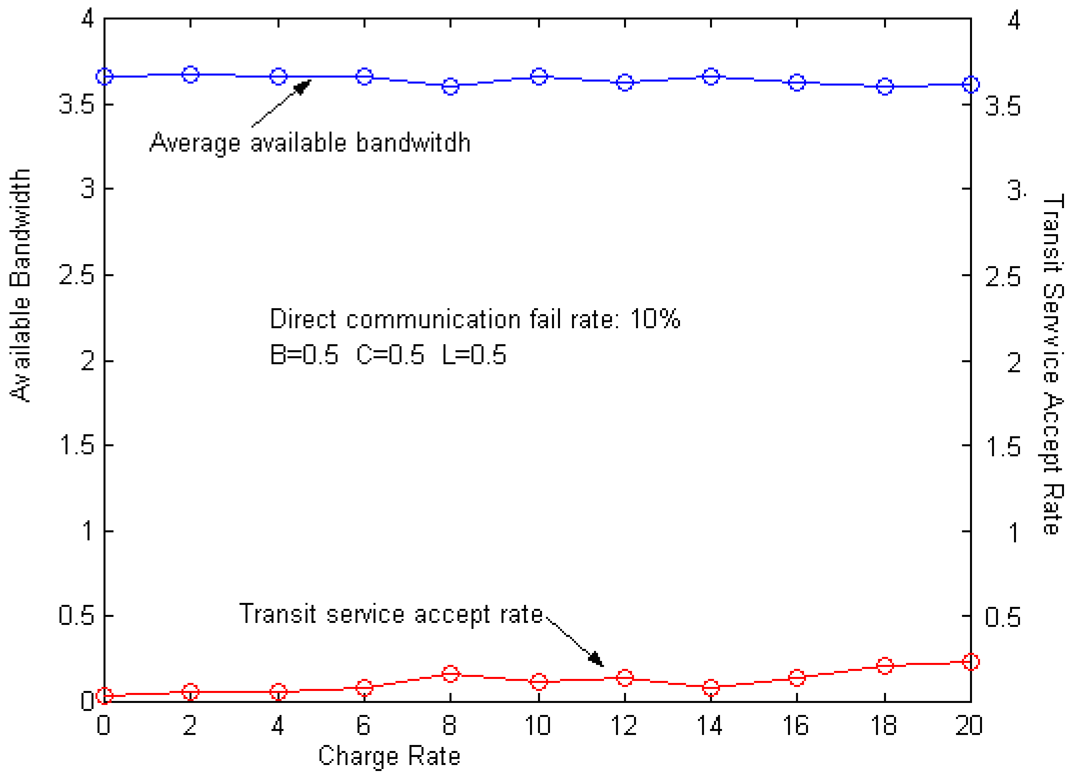
Acceptance Probability does not affect the total bandwidth usage when the Direct Connection probability is high.



(a)



(b)



(c)

Figure 7.13 Relationship between Transit Service Acceptance Probability and Payment/Charge Rate

7.6 Summary

In a multi-domain system, an individual AS tends to demonstrate selfish behaviours, which might degrade overall performance and network resource utility. To encourage the cooperation among the ASes, a novel credit based inter-domain management mechanism is proposed and is designed for the DVM based DVPN architecture, where the DVM provide negotiation and communication among ASes. An independent equipment entity called the “Credit Centre” is introduced to perform the credit management. Its performance is studied through simulations. From the preceding results we can conclude that the credit based inter-AS management architecture is more efficient, controllable and robust than without it.

Chapter 8 Discussion, Conclusions and Further Work

8.1 Discussion

The author proposes a new service architecture (in Chapter 4) that extends the use of a Multi-Protocol Label Switching (MPLS) infrastructure for the formation and operation of dynamic communities, incorporating value-added resources allied to the tasks being performed. The approach introduces a new operator-owned control-plane entity called the Dynamic VPN Manager (DVM) for managing customer communities and liaising with the operator's infrastructure. With the help of the DVM entity, an inter-AS BGP/MPLS VPN can be built and managed automatically. This architecture has more merits than traditional BGP/MPLS VPNs. In this network, it is possible to provide QoS guaranteed MPLS VPN links among different ASes, because before an inter-AS routing decision is made, there is the negotiation among the DVMs. The route is built based on the result of the negotiation. At this same time this architecture is able to provide dynamic failure recovery for inter-AS path. The BGP edge routers can act as proxy ingress points to the adjacent domains and may have pre-established backup LSPs that are AS diverse. In the event of a failure, the DVM receives the failure report through information exchanges with other ASes and sends commands to the CM to tell the egress BGP router to perform path switch-over, though this may isolate small clusters of the VPN community within the affected AS.

The communities take the form of dynamic VPNs that can be used to support extranet-based on-demand services compatible with "grid computing" and other distributed, collaborative activities. The significance of the approach is further strengthened by supporting methods that allow automated business processes to dynamically request communication infrastructure and processing resources that are compatible with state-of-the-art commercial standards in distributed computing.

The potential argument to this architecture might focus on the security. To satisfy the security requirements, in the proposed DVPN architecture the control planes are separated. Users can only exchange message with DVM and network edge equipment. The core network routers and equipment are never visible from the view point of end users. Meanwhile, a DVM cannot see any SP's equipment of other ASes. It can only communicate with DVMs regarding the inter-domain service requirements. An AS looks like a "black box" to the DVMs of other ASes. As a result, this DVPN architecture does not add security risks. Another issue is the scalability. Being an AS manager, a DVM needs to deal with a lot of requests coming from users and other DVMs. It also needs to store a lot of information, for example, the network resource availability. It also needs to do intensive computation to calculate the VPN allocations, which requires considerable CPU capacity. The failure tolerance issue is also necessary to be considered. Once the DVM does not work, the DVPN system will fail. Considering this point, the DVM can be a cluster of sub-DVMs, the sub-DVMs cooperate to provide the DVM management functions, if one sub-DVM dies, others can continue working. However, there is still one DVM to the end users and other DVMs. The available network resources can also be utilised by the DVM for intensive computation and information storage.

To demonstrate the DVPN idea, a test-bed was developed and described in Chapter 5. Prototypes of the DVM, PE, CE were developed over Linux PCs. In this thesis, a simple experiment is carried out to demonstrate the basic DPVN actions and the feasibility of the DVPN architecture. This test-bed provides a platform to perform further experiments and developments. It also shows the potential industry utilisation of this DVPN architecture. BT (British Telecom) has shown a great interest in it.

Meanwhile, the author also studied the resource scheduling problem for the reason that the orchestrated computing is one of the most important potential applications. A dynamic resource scheduling algorithm is proposed in Chapter 6. The advantage of this approach is that the algorithm does not need of complete information when jobs arrive, but only considers the tasks in the RTS, and possesses no foresight.

Compared with a static scheme where full knowledge of the jobs is known at the time when their efficient placement is being determined, this proposed dynamic algorithm has shown that the lack of foresight does have a negative impact on the efficiency of the job placement. This problem is notably worse when there is considerable dependency between the tasks that make up each job. A potential solution is to make the algorithm to take into account foresight for several steps, which needs to be trade off with the dynamic ability. Predication technologies can also be used to improve the algorithm efficiency.

Both GA and PSO use the “fitness” to represent whether a potential arrangement is close to the optimal solution. For simplicity, this thesis assumed that tasks have no specific resource requirements and so can be potentially allocated to any of the available resources. However, various constraints and requirements can be applied by modifying the fitness function so that an appropriate solution can be selected by the algorithm.

In Chapter 7, a credit based Inter-AS management architecture is proposed. The “credit” is introduced as “virtual money” for the transit service. The simulation results show that this CIM system improves cooperation among ASes and increases the overall network efficiency while the self benefits are also considered. The potential issue is the overhead of the CIM signalling. The CIM is built over the DVM based DVPN system. The CIM signalling messages are exchanged among the DVMs and the Credit Centre. The CIM signalling is separated from the underlying network. As a result, the CIM signalling will not affect the bandwidth competition communications among the underlying network equipment. There might be also argument concerning the scalability. At this point, the Credit Centre can also use the cluster solution and utilise the available network resources as the DVM might.

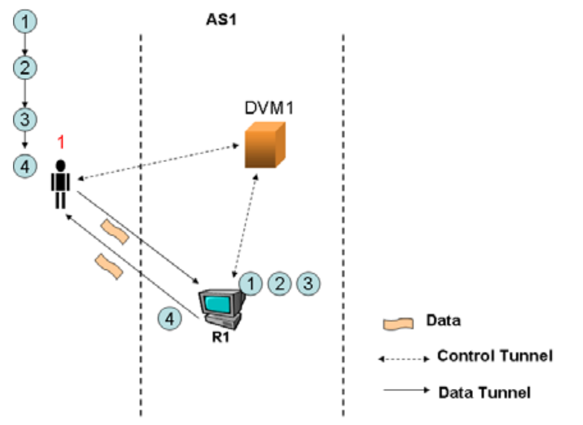
In this thesis, the author designed a novel Dynamic VPN architecture through the introduction of a new entity-DVM. The DVPN is an open system where various functions can be further developed. The test-bed and simulation model developed during this research aimed to assess the basic functionalities and performance, and to investigate the potential concerns of this new architecture. A number of issues and points of further interest have been outlined and addressed through the discussion above.

8.2 Scalability Analysis

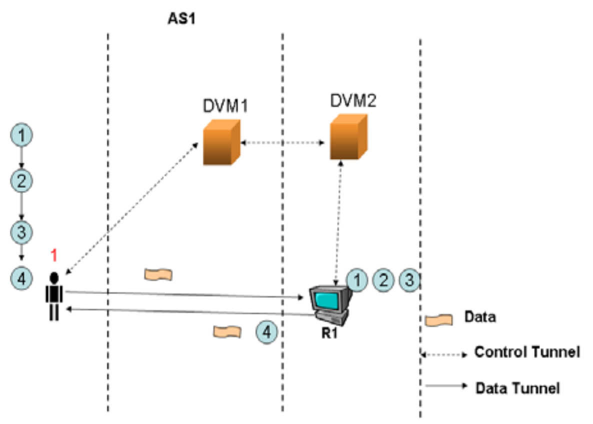
The control messages constitute an overhead in the DVPN system. In this section the scalability characteristics of the DVPN control message interactions are considered.

8.2.1 DVPN Control Messages

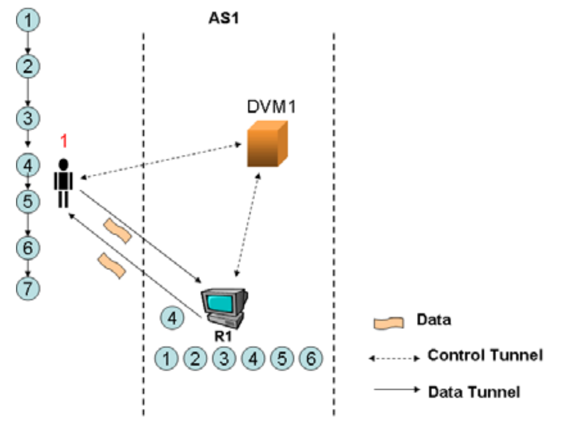
The number of the DVPN control messages is dependent on the size and pattern of the scenario including, for example, the number of tasks, dependency relationships and the resources employed. The number of control messages is also dependent upon whether inter-domain services are required. In the following description, six typical scenarios shown in Figure 8.1 are examined in terms of the number of control messages (overhead) they give rise to. The results are listed in Table 8.1



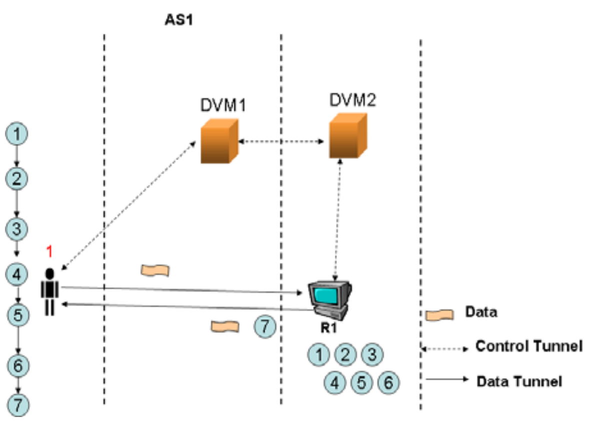
(a) Intra-Domain; Single Resource Simple Tandem Task Sequence



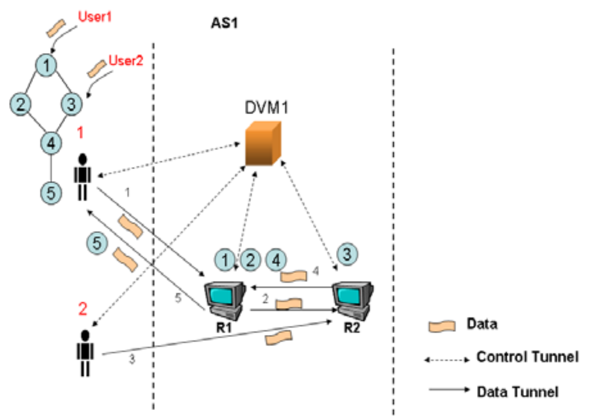
(b) Inter-Domain; Single Resource Simple Tandem Task Sequence



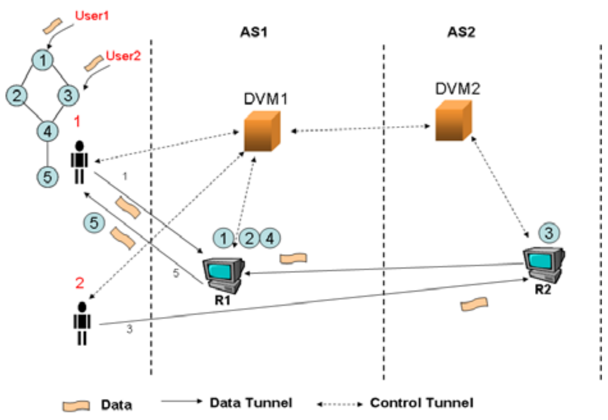
(c) Intra-Domain; Single Resource Long Tandem Task Sequence



(d) Inter-Domain; Single Resource Long Tandem Task Sequence



(e) Intra-Domain; Single Resource Concurrent Task Sequence



(f) Inter-Domain; Single Resource Concurrent Task Sequence

Figure 8.1 Scenario Examples

Table 8.1 Number of Control Messages Produced

	Scenario					
Message type	a	b	c	d	e	f
1. Service Request	1	1	1	1	1	1
2. Service Accepted	1	1	1	1	1	1
3. VPN Link Setup	2	4	2	4	4	6
4. VPN Link Setup Confirm	2	4	2	4	4	6
5. VPN Link Remove	2	4	2	4	4	6
6. VPN Link Remove Confirm	2	4	2	4	4	6
7. Inter-domain VPN Link Request	0	2	0	2	0	2
8. Inter-domain VPN Link Confirm	0	2	0	2	0	2
9. Inter-domain label Request	0	2	0	2	0	2
10. Inter-domain label Reply	0	2	0	2	0	2
11. Inter-domain LSP Setup	0	2	0	2	0	2
12. Inter-domain LSP Setup Confirm	0	2	0	2	0	2
13. Inter-domain VPN Link Remove	0	2	0	2	0	2
14. Inter-domain VPN Link Remove Confirm	0	2	0	2	0	2
15. Inter-domain LSP Remove	0	2	0	2	0	2
16. Inter-domain LSP Remove Confirm	0	2	0	2	0	2
17. Inter-domain Label Release	0	2	0	2	0	2
18. Resource Book	3	3	6	6	4	4
19. Resource Book Confirm	3	3	6	6	4	4
20. Inter-domain Resource Booking	0	3	0	6	0	1
21. Inter-domain Resource Booking Confirm	0	3	0	6	0	1
22. Data Transfer	2	2	2	2	4	4
23. Data Transfer Complete Confirm	2	2	2	2	4	4
24. Inter-domain Data Transfer	0	1	0	1	0	1
25. Inter-domain Data Transfer Complete	0	1	0	1	0	1
26. Task Complete	3	3	6	6	4	4
27. Inter-domain Task Complete	0	3	0	6	0	1
Total Messages	23	64	32	82	38	73

According to the above data, the conclusion can be drawn that the scenario size will affect the DVPN overhead (the number of control messages). Generally, a bigger size scenario containing more tasks and resources requires more control messages than a small one. The number of necessary control messages increases approximately lineally according to the increasing of the scenario size. The scenario pattern (i.e. the mean upper dependent tasks) also affects the number of control messages. For the same size scenario, the one whose tasks have more upper dependent tasks will require more control messages. This is because more VPN links are necessary for the data transactions from the upper dependent tasks to the downstream tasks. On the other hand, the inter-domain service requires more control messages than the intra-domain case. This is because that the inter-domain service requires the messages for inter-domain negotiations and information exchange which is not necessary for an intra-domain service. Meanwhile, an inter-domain VPN link also requires more processes to be setup/removed than an intra-domain one (this is because an inter-domain VPN link is formed by splicing several intra-domain VPN segments), which also increases the number of control messages.

8.2.2 VPN Action Performance for Various Scenarios

The number of the control messages is related to how many VPN actions (create/remove VPN links) are required. The number of VPN actions necessary is in turn dependent upon the scenario patterns (i.e. the DAG structure and how the DVM decides to map the tasks onto the available resources). How the scenarios affect the number of VPN actions necessary is studied in this section.

A VPN link is created before a data transfer and this VPN link will be removed after the data transfer completes. For this reason, the VPN create and remove actions are considered in the following simulations. The VPN actions defined in this section refer to both the intra/inter domain VPN actions.

The relationship between the number of VPN actions and the mean number of dependent uppers is studied through simulations; the results are shown in Figure 8.2. In the simulations, the total number of tasks is 100; there are 3 available resources. 20 job scenarios are generated each with 1 to 25 mean dependent uppers, thus there are 500 scenarios, though each experiment is only conducted once in this case. The job scenario is generated in the same way as described in Section 6.3.6. The number of VPN actions (Create/Remove VPN) is recorded for each simulation and the average results of 20 simulations are calculated. The results imply that the number of VPN actions increases linearly with respect to the mean number of dependent uppers.

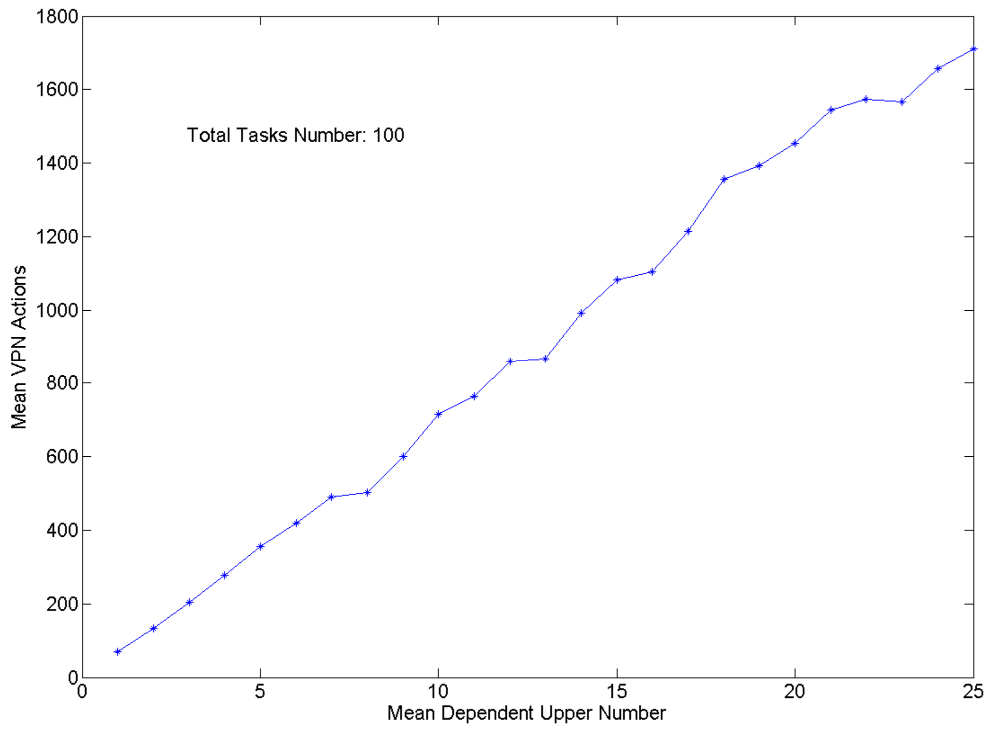


Figure 8.2 Mean VPN Actions versus Mean Number of Dependent Uppers

The number of VPN actions is related to how many tasks are included in the job scenario. This relationship is also studied through simulations. 20 simulations are carried with the same number of tasks (100) in a 3-

resource environment and mean number of dependent uppers tasks is set to 5. The resource capability is set to 1, 2 and 3 for the three resources. The average number of the VPN actions is recorded. The results suggest that the number of VPN actions also increases in a linear manner relative to the total task number as shown in Figure 8.3.

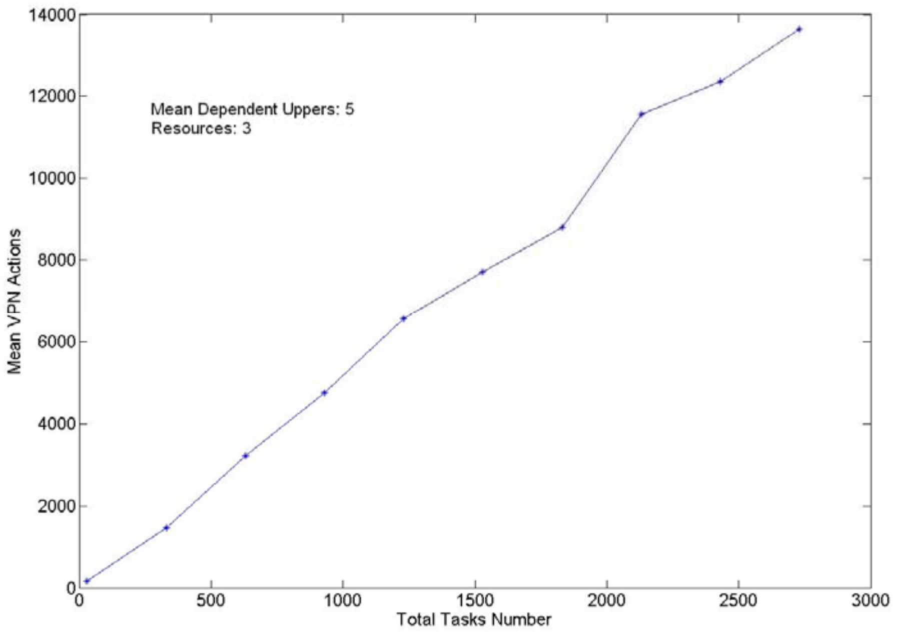


Figure 8.3 VPN Actions Vs Total Tasks Number

The relationship between the number of VPN actions and the number of resources is studied. In the simulations, 20 independent trials are carried out with the same number of resources, again with the processing capability being set to 1, 2 and 3. The job scenario consists of 100 tasks; each task has average 5 dependent uppers. The results imply that the number of VPN actions increase with the number of resources; however, the speed of increase reduces quickly, shown in Figure 8.4. The reason why the number of VPN actions increase with the number of resources is that, as the resources number increases, tasks will be allocated to more resources. As a result more VPN links are required for data transactions among dependent tasks (from one resource to another where the downstream task is located). However, when the number of resources is bigger than the total task number, an upper bound is reached, allocating each task to a different resource leads to the maximum number of VPN actions. Increases in the resource number beyond this point will not require additional VPN actions.

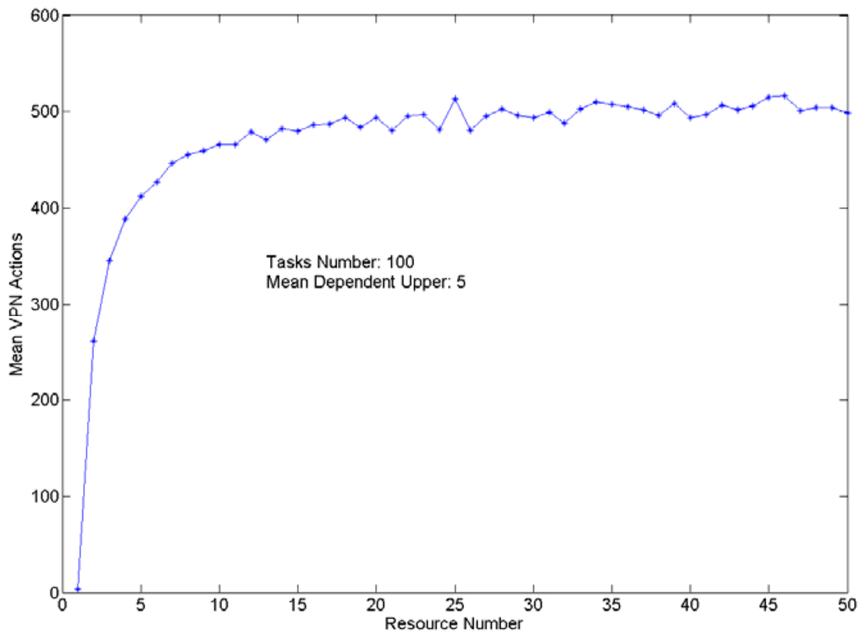


Figure 8.4 Mean VPN Actions versus Number of Resources

8.2.3 Number of Tasks versus Computation Time

The computation time required for calculating the resource scheduling is also considered as a potential scalability issue for the DVPN system. The relationship between the computation time and the total number of tasks is studied with simulations. 20 trials are carried out for the same scenario[6], and the average results are recorded. For all the simulations, the generation number is set to 10 and population ratio is set to 1. The results suggest that the computation time increases with the total tasks number in a linear manner as shown in Figure 8.5.

[6] In this case it is only the heuristic output from the scheduler that will vary as the seed for the RNG is altered for each trial.

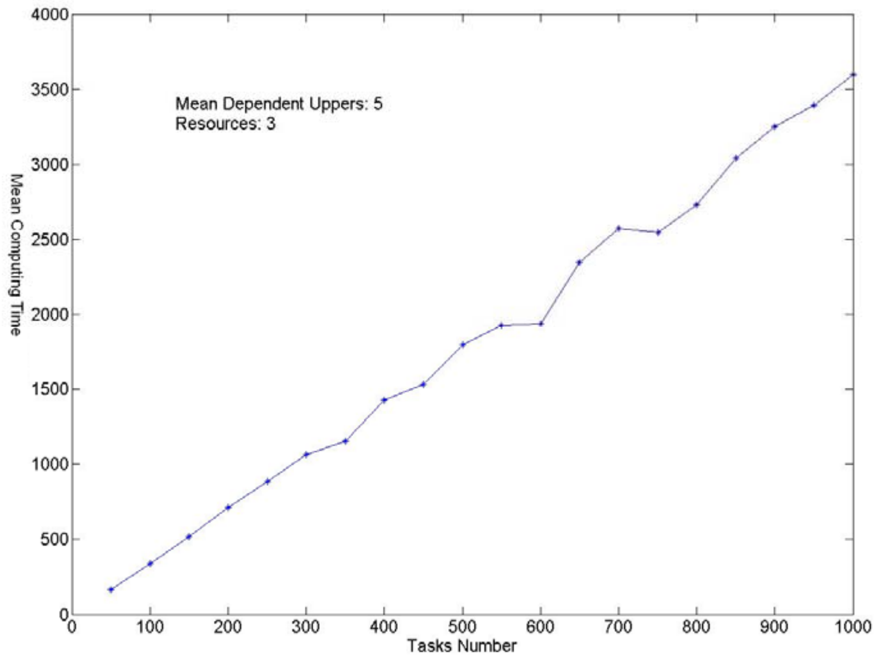


Figure 8.5 Mean Computation Time for Various Numbers of Tasks

8.2.4 Summary

According to the above appraisal, the DVPN architecture is not an NP-hard system. The overhead typically increases in a linear manner with increasing scale of the scenario. Considering this point, if the performance were to become an issue, the DVM could be configured into a cluster arrangement so that the VPN processing could be partitioned across multiple DVMs within an AS domain. Alternatively, as considered in Section 6.3, the resource scheduling processing burden can be reduced by regulating the number of iterations permitted in the optimisation mechanism. A further refinement is for the DVM to exploit some of the processing resources itself and use them to assist it in calculating the scheduling arrangements.

8.3 Conclusions

New applications (such as grid computing and global e-business) can employ orchestrated services and computing to carry out complex and computation-intensive jobs. They require that QoS guaranteed communications be provided over the Internet dynamically and automatically, which challenges existing network technologies. In this thesis, the author carried out an in-depth study on the technologies of BGP MPLS/IP VPNs and related technologies. The shortcomings of the existing technologies and the limitations of current proposed solutions are also analysed.

In this thesis, the author proposes a dynamic VPN architecture. The most important- component of this architecture is the DVM, which is the manager of the VPN. With the help of the DVM entity, inter-AS MPLS/BGP VPNs can be built and managed automatically. This architecture has more merits than traditional BGP/MPLS VPNs. This architecture is more flexible than current inter-AS VPN schemes in that it provides a suitable framework for automated VPN management, together with the ability for operators to integrate VPN management with value-added service resources such as processor “farms”, archival systems and so forth. A DVPN test-bed is built up successfully to demonstrate how the DVPN works, while showing its potential commercial usage.

The resource-scheduling problem for the orchestrated distributed computing is also studied in depth. A dynamic resource scheduling mechanism is proposed. GA and PSO are employed as the heuristic optimisation engines. The algorithm is examined using simulations. Both the GA chromosome population and PSO particle number are adapted to the current tasks number in RTS, which improves the optimisation efficiency. The results show that this algorithm is feasible and should operate efficiently within a typical DVPN scenario. The performance of GA and PSO variants are compared though simulations. For a given number of iterations, although GA tends to give a better solution, it takes considerably more computation time than PSO.

The selfish behaviour of the individual AS degrades the whole network performance. The DVPN system provides the negotiation mechanism among the ASes through the communication between DVMs. A Credit-based CIM mechanism is proposed to build up on the DVM based DVPN system. The simulation results imply that the CIM can improve the overall network efficiency through cooperation incentives among the ASes.

Nevertheless, the satisfactory operation of the DVPN architecture has been demonstrated. In embryonic form it offers a framework that facilitates management of dynamic VPN operations, suitable to support orchestrated computing and services across the domains over the Internet.

8.4 Further Work

There are several directions where the work presented in this thesis could be developed further.

One of the plans is to extend the current test-bed to explore the interaction with the Resource Management entities and to evaluate the inter-provider discovery, negotiation and LSP splicing mechanisms. This DVPN architecture can be further developed to be installed within a Service Provider (i.e. BT) experimental network. All kinds of tests can be launched and the performance can be examined in the Internet environment. Meanwhile, the simulation models could also be extended to explore its performance in large-scale scenarios.

The resource-scheduling algorithm could be further studied. In the current version, the optimisation engines only consider the tasks in RTS. The simulation results show that the algorithm performance degrades in the case that the job scenarios are very dependent. A potential solution is to make the optimisation tools predict several steps into the future and the trade off between efficiency and dynamic ability is also a potential issue to consider. According to the predications of the job scenarios and resources requirements, the resource-scheduling scheme can be adjusted in advance.

In this thesis, the CIM system is tested in a simple simulation scenario. In a future, this system could be tested in a larger and more complex scenario through simulations. More characteristics could be evaluated such as the overhead and failure restoration performance. Currently, the credit is only used to cost the bandwidths provided by the transit AS. However, credit can be applied to more resources and services, for example the CPU and databases. Another opportunity would be to realise the CIM over the DVPN test-bed.

References

[1] W. Augustyn *et al*, “Service Requirements for Layer-2 Provider Provisioned Virtual Private Networks”, Internet Engineering Task Force Document: draft-ietf-l2vpn-requirements-03.txt, October 2004.

[2] M. Carugi *et al*, “Service requirements for Layer 3 Virtual Private Networks”, Internet Engineering Task Force Document: draft-ietf-l3vpn-requirements-02.txt, July 2004.

[3] E. Rosen *et al*., “Multiprotocol Label Switching Architecture”, RFC 3031, January 2001.

[4] Luyuan Fang *et al*, “Interprovider IP-MPLS Services:Requirements, Implementations, and Challenges”, Communications Magazine, IEEE, Volume: 43, Issue: 6, On page(s): 119- 128, Date: June 2005.

[5] E. Rosen *et al*, “RFC 2547 - BGP/MPLS VPNs”, RFC 2547, October 2004.

[6] Chuck Semeria, “White Paper-RFC 2547bis: BGP/MPLS VPN Fundamentals”, Marketing Engineer, Juniper Networks, Inc., http://connect.to.jonathan.googlepages.com/juniper_mpls_vpn_200012.pdf, accessed at 15/09/2008.

[7] E. Rosen *et al*, “BGP/MPLS IP VPN5”, Internet Engineering Task Force Document: draft-ietf-l3vpn-rfc2547bis-03.txt, Oct. 2004.

[8] Murray E. *et al*, “e-Business and Technology Issues for Developing Economies:A Ukraine Case Study”, Electronic Commerce Research, Volume 4 , Issue 3, Pages: 263 – 286, 2004.

[9] Murray E. *et al*, “UNCTAD and E-Commerce Success”, The Electronic Journal on Information Systems in Developing Countries (EJISDC) 2003 Volume 11, Pages: 1-5, 2000.

[10] Peter Kacsuk *et al*, “GridDemo: International Workshop on Live Demonstrations of Grid Technologies and Applications”, CCGRID 2002, Page(s):298 – 298, 21-24 May 2002.

[11] D.D. Kouvatsos *et al*, “Multicast communication in grid computing networks with background traffic”. Software, IEE Proceedings , Volume 150, Issue 4, Page(s):257 – 264, Aug. 2003.

[12] Pu Juhua *et al*, “THE RESEARCH ON QOS FOR GRID COMPUTING”, ICCT2003, Volume 2, Page(s):1711 – 1714, April 2003.

[13] Raymond Zhang, “QoS-enabled Inter-Provider MPLS VPN Services”, Inter-provider QoS Workshop, MIT, Cambridge, MA, October, 2004

[14] Yiran Gao *et al*, “Inter-Provider Dynamic VPNs”, Pgnet 2005, Liverpool, UK, June, 2005.

[15] “VPN Technologies: Definitions and Requirements”, VPN Consortium, July 2008, http://www.vpnc.org/vpn-technologies.html, accessed at 18/09/2008.

[16] “IP/MPLS-Based VPNs-Layer-3 vs. Layer-2”, http://www.foundrynet.com/solutions/appNotes/PDFs/L3vsL2.pdf, accessed at 28/10/2006.

[17] Muthucumaru Maheswaran *et al*, “Dynamic Matching and Scheduling of a Class of Independent Tasks onto Heterogeneous Computing Systems”, 8th Heterogeneous Computing Workshop (HCW ’99), pagers: 19, 1999.

[18] Ajith Abraham *et al*, “Heuristics for Scheduling Jobs on Computational Grids”, The 8th, IEEE International Conference on Advanced Computing and Communications (ADCOM 2000), 2000. http://www-mugc.cc.monash.edu.au/~abrahamp/adcom2000.pdf, accessed at 21/09/2008.

[19] J. Moy, “OSPF Version 2”, RFC 2328, April 1998.

[20] Y. Rekhter *et al*, “A Border Gateway Protocol 4 (BGP-4)”, RFC 1771, March 1995.

[21] G. Malkin, “RIP Version 2”, RFC 2453, November 1998.

[22] “SISCO Internetworking Technologies Handbook Chapter 39, Border Gateway Protocol”, http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/bgp.pdf, accessed at 18/09/2008.

[23] Y. Rekhter *et al*, “Application of the Border Gateway Protocol in the Internet”, RFC1772, March 1995.

[24] P. Traina, “Experience with the BGP-4 protocol”, RFC1773, March 1995.

[25] P. Traina, “BGP-4 Protocol Analysis”, RFC 1774, March 1995.

[26] S. Willis, *et al*, “Definitions of Managed Objects for the Fourth Version of the Border Gateway Protocol (BGP-4) using SMIv2”, RFC 1657, July 1994.

[27] “BGP-4 Protocol Overview”, http://freesoft.org/CIE/Topics/88.htm, access at 18/09/2008.

[28] Y. Rekhter *et al*, “Carrying Label Information in BGP-4”, RFC 3107, May 2001.

[29] T. Bates *et al*, “RFC 2283, Multiprotocol Extensions for BGP-4”, February 1998.

[30] T. Bates *et al*, “BGP Route Reflection -An Alternative to Full Mesh IBGP”, RFC 2796 April 2000.

[31] P. Ferguson *et al*, “What is a VPN?”, Revision 1, April 1998, http://www.potaroo.net/papers/1998-3-vpn/vpn.pdf, accessed at 18/09/2008.

[32] “Cisco MPLS based VPNs: Equivalent to the security of Frame Relay and ATM”, http://www.cedarcom.net/Miercom%20paper%20on%20Cisco's%20MPLS-VPNs.pdf, accessed at 09/08/2008.

[33] Y. Rekhter *et al*, “Address Allocation for Private Internets”, RFC 1918, February 1996.

[34] CISCO, “MPLS VPN—Inter-AS—IPv4 BGP Label Distribution”, http://www.cisco.com/en/US/docs/ios/12_0st/12_0st21/feature/guide/fsbgp.html, accessed at 19/09/2008.

[35] J.T. Park, “management of BGP/MPLS VPN with resilient path”, NOMS 2004, Volume 1, Pages:177 – 190, 23-23 April 2004.

[36] Anotai Srikitja *et al*, “Topological Design of Multiple VPNs over MPLS Network”, Global Telecommunications Conference, 2002, Volume 3, Pages: 2195 – 2199, Nov. 2002.

[37] “Spirent White Paper: MPLS/BGP Virtual Private Networks”, http://www.spirent.com/documents/427.pdf, accessed at 09/08/2008.

[38] P. Marques *et al*, “Constrained Route Distribution for Border Gateway Protocol/MultiProtocol Label Switching (BGP/MPLS) Internet Protocol (IP) Virtual Private Networks (VPNs)”, RFC 4684, November 2006.

[39] “ANALYSIS OF MPLS-BASED IP VPN SECURITY: COMPARISON TO TRADITIONAL L2VPNS SUCH AS ATM AND FRAME RELAY, AND DEPLOYMENT GUIDELINES”, http://www.cisco.com/warp/public/cc/so/neso/vpn/prodlit/mpvpn_wp.pdf, accessed at 09/08/2008.

[40] M. Behringer, “Analysis of the Security Of BGP/MPLS IP Virtual Private Nextwork”, RFC 4381, February 2006.

[41] Angefertigt nach, *et al*, “BGP convergence analysis”, Diplomarbeit, am Fachbereich Informatik, der

Universit at des Saarlandes, 16. June 2003.

[42] C. Labovitz *et al*, "The impact of internet policy and topology on delayed routing convergence", Infocom 2001, Volume: 1, pages: 537-546, April 2001.

[43] C. Labovitz *et al*, "Delayed internet routing convergence", Networking, IEEE/ACM Transactions on, Volume: 9, pages: 293-306, Jun 2001.

[44] Johan Nykvist *et al*, "Simulating Convergence Properties of BGP", Computer Communications and Networks, 2002, Page(s): 124 – 129, Oct. 2002.

[45] T. Griffin *et al*, "An Experimental Analysis of BGP Cvergence Time", Network Protocols, 2001, pages 53-61, Nov. 2001.

[46] Dan Pei1 *et al*, "Improving BGP Convergence Through Consistency Assertions", INFOCOM 2002, Volume: 2, pages: 902- 911, 2002.

[47] Gao, L., "On inferring automonous system relationships in the internet", Networking, IEEE/ACM Transactions, Volume: 9, pagers 733-745, Dec 2001.

[48] RFC 4272, http://www.ietf.org/rfc/rfc4272.txt?number=4272, accessed at 19/07/2007.

[49] R. A. Guerin *et al*. "QoS routing mechanisms and OSPF extensions", IEEE Globecom 97', pages 1903–1908, Nov. 1997.

[50] Li Xiao *et al*, "QoS Extension to BGP", Networking and Services, 2006. ICNS '06, On page(s): 80-80, 2006.

[51] Anat Bremler-Barr *et al*, "Improved BGP Convergence via Ghost Flushing", INFOCOM 2003, Volume: 2, pages: 927- 937 vol.2, March. 2003.

[52] Meiyuan *et al*, "The Performance Impact of BGP Security", Network, IEEE, Volume: 19, Issue: 6, On page(s): 42- 48, Nov.-Dec. 2005.

[53] Ke Zhang "An Analysis on Selective Dropping Attack in BGP", Performance, Computing, and Communications, 2004 IEEE International Conference on, On page(s): 593- 599, 2004.

[54] Stephen Kent *et al*, "Secure Border Gateway Protocol (S-BGP)", IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, Vol. 18, pages 582-592, April. 2000.

[55] http://en.wikipedia.org/wiki/X.509, accessed at 10/09/2008.

[56] Lakshminarayanan Subramanian *et al*, "HLP: A Next Generation Interdomain Routing Protocol", SIGCOMM'05, August, 2005.

[57] O. Bonaventure, "Using BGP to distribute flexible QoS information", Internet Draft: draft-bonaventure-bgp-qos-00.txt, work in progress, February 2001.

[58] G. Cristallo *et al*, "Providing Quality of Service Indication by the BGP-4 Protocol: the QOS NLRI attribute", Internet Draft draft-jacquenet-qos-nlri-03.txt. Work in progress, March 2002.

[59] B. Abarbanel *et al*, "BGP-4 Support for Traffic Engineering", Internet Draft draft-abarbanel-idr-bgp4-te-00.txt. Work in progress, September 2000.

[60] A. Fei *et al*, "Extending BGMP for shared-tree interdomain QoS multicast", IWQoS, June, 2001.

[61] Liwen He, "A Novel Scheme on Building a Trusted IP Routing Infrastructure", Networking and Services, 2006. ICNS '06, pages 13-13, 2006.

[62] D.W. Manchala, “Trust metrics, models and protocols for electronic commerce transactions”, The 18th International Conference on Distributed Computing Systems, May, 1998.

[63] S. Ba, “Establishing online trust through a community responsibility system”, Decision Support Systems, Vol. 31,pp. 323-336, 2001.

[64] R. Au *et al*, “Automated Cross Organisational Trust Establishment on Extranets”, Workshop on Information Technology for Virtual Enterprises, On page(s): 3-11, January 2001.

[65] L. Mui *et al*, “A Computational Model of Trust and Reputation”, Proceedings of the 35th Hawaii International Conference on System Science, Volume, Issue, Page(s): 2431 – 2439, Jan. 2002.

[66] M. Y. Siyal *et al*, “A novel trust service provider for Internet based commerce applications”, Internet Research: Electronic Networking Applications and Policy, vol.12, no.1, pp 55-65 2002.

[67] X. Li *et al*, “A reputation-based trust model for Peer-to-Peer eCommerce Communities”, Proceedings of the IEEE International Conference on E-commerce, On page(s): 275- 284, June 2003.

[68] A. Kapadia *et al*, “Routing with Confidence: Supporting Discretionary Routing Requirements in Policy Based Networks”, Fifth IEEE International Workshop on Policies for Distributed Systems and Networks, On page(s): 45- 54 June 2004.

[69] Zhong, S *et al*, “Sprite: a simple, cheat-proof, credit-based system for mobile ad-hoc networks”, INFOCOM 2003, Volume 3, Page(s):1987 – 1997, March-April 2003.

[70] Younghwan Yoo *et al*, “A Credit-Payment Scheme for Packet Forwarding Fairness in Mobile Ad Hoc Networks Communications”, ICC2005, Page(s):3005 – 3009, May 2005.

[71] Teruyuki KOMIYA *et al*, “A Proposal on Flexible CUG Service Architecture”, Parallel and Distributed Systems: Workshops, Seventh International Conference on, 2000, Pages:315 – 320, July 2000.

[72] Sebastian Staamann *et al*., “Closed User Groups In Internet Service Centres”, 2nd IFIP International Working Conference on Distributed Applications and Interoperable Systems (DAIS'99), June, 1999.

[73] Kindred *et al*. “Dynamic VPN communities: implementation and experience”, DARPA Information Survivability Conference & Exposition II, 2001. DISCEX '01, Volume 1, Page(s):254 - 26312-14, June 2001.

[74] Y. Jia *et al*, “Dynamic resource allocation in QoS-enabled/MPLS supported virtual private networks and its Linux based implementation”, Electrical and Computer Engineering, 2002. IEEE CCECE 2002, Volume 3, Page(s):1448 - 1454 vol.3, May 2002.

[75] Y. Jia *et al*, “Design and testbed implementation of adaptive MPLS-DiffServ enabled virtual private networks”, Electrical and Computer Engineering, 2003. IEEE CCECE 2003, Volume 2, Page(s):965 - 968, May 2003.

[76] P. Lago *et al*, “A TINA-based solution for dynamic VPN Provisioning on heterogeneous Networks”, Proc. IEEE Telecommunications Information Networking Architecture Conference (TINA'2000), http://www.tinac.com/conference/proceedings/day2/scientific2/Lago.pdf, accessed at 21/09/2008.

[77] Ivan Đorđević, “Architecture for Dynamic and Secure Group Working”, The PhD Thesis, Department of Electronic Engineering Queen Mary, University of London ,United Kingdom, June 2004.

[78] Fujita, N. *et al*, “Scalable overlay network deployment for dynamic collaborative groups”, Proc. 2005

Symposium on Applications and the Internet, Page(s):102 – 109, 2005.

[79] http://www.6net.org/, accessed at 22/08/2008.

[80] Dr. S. Zeber *et al*, “The Dynamic VPN Controller”, Defence R&D Canada-Ottawa, TECHNICAL MEMORANDUM, DRDC Ottawa TM 2005-025, March 2005, http://www.ottawa.drdc-rddc.gc.ca/docs/e/TM2005-025.pdf, accessed at 22/08/2008.

[81] http://www.6net.org/publications/deliverables/D5.8.pdf, accessed at 22/08/2008.

[82] S. Mary Saira Bhanu *et al*, “A Hyper-Heuristic Approach for Efficient Resource Scheduling in Grid”, Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844, Vol. III, No. 3, pp. 249-258, 2008.

[83] Juan Antonio G`onzalez, “A Hyper-heuristic for scheduling independent jobs in Computational Grids”, http://www-sop.inria.fr/mascotte/WorkshopScheduling/Slides/Gonzalez.pdf, accessed at 04/08/2008.

[84] R. Armstrong *et al*, “The relative performance of various mapping algorithm is independent of sizable variance in run-time predictions”, 7th Heterogeneous Computing Workshop (HCW’ 98), pp. 79-87, March 1998.

[85] R. Freund *et al*, “SmartNet: a scheduling framework for heterogeneous computing”, The International Symposium on Parallel Architectures, Algorithms, and Networks, On page(s): 514-521, June 1996.

[86] Ibarra *et al*, “Heuristic algorithms for scheduling independent tasks on non identical processors”, Journal of the ACM, 24(2):280-289, April 1977.

[87] Johnson P, Thomas *et al*, “A hierarchical Petri net framework for the representation and analysis assembly”, Systems, Man and Cybernetics, IEEE Transactions, Volume: 24, Issue: 4, On page(s): 564-573, April 1994.

[88] David Abramson, “High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid?”, Parallel and Distributed Processing Symposium, 2000. IPDPS 2000, Volume, Page(s):520 – 528, 2000.

[89] Saleem Bhatti *et al*, “Grid Resource Scheduling”, Computer Supported Cooperative Work in Design, 2005, Volume: 1, On page(s): 367- 372, May 2005.

[90] JianQiang Li *et al*, “Timing Constraint Workflow Nets for workflow analysis”, Systems, Man and Cybernetics, Part A, IEEE Transactions Volume 33, Issue 2, Page(s): 179 – 193, March 2003.

[91] Kanmal S *et al*, “Clock selection for performance optimisation of control-flow intensive behavior”, Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions, Volume 20, Issue 1, Page(s):158 – 165, Jan 2001.

[92] Jin Hyun *et al*, “Finding the critical path in a time-constrained workflow”, Real-Time Computing Systems and Applications, Volume, Issue, 2000 Page(s):102 – 107, 2000.

[93] Arvind Sudarsanam *et al*, “Resource estimation and task scheduling for multithreaded reconfigurable architectures”, Proceedings of the tenth international conference on parallel and distributed system (ICPADS’04), On page(s): 323- 330, July 2004.

[94] Kwang-Hoon Kim *et al*, “Performance estimations of clustered workflow architectures”, Proceedings of the Fourth Annual ACIS International Conference on Computer and information Science (ICIS’05), On

page(s): 288- 293, 2005.

[95] R. Armstrong, “Investigation of Effect of Different Run-Time Distributions on SmartNet Performance”, Master’s thesis, Department of Computer Science, Naval Postgraduate School, 1997 (D. Hensgen, advisor).

[96] Maheswaran *et al*, “A dynamic matching and scheduling algorithm for heterogeneous computing systems”, Heterogeneous Computing Workshop, 1998. (HCW 98) Proceedings, Page: 57, 1998.

[97] Alhusaini *et al*, “A unified resource scheduling framework for heterogeneous computing environments”, Heterogeneous Computing Workshop, 1999. (HCW '99), On page(s): 156-165, 1999.

[98] R. Jain *et al*, “Soft real-time scheduling on simultaneous multithreaded processors”, Proceedings of the 23rd Real- Time Systems Symposium, On page(s): 134- 145, 2002.

[99] Tracy D.Braun *et al*, “A Comparison of eleven static heuristics for mapping a class of tasks to heterogeneous Distributed Computing System”, Journal of Parallel and Distributed Computing Vol:61, pp: 810 - 837, 2000.

[100] “Genetic algorithm”, Wikipedia, http://en.wikipedia.org/wiki/Genetic_algorithm, accessed 21/09/2008.

[101] Christos A. FRANGOPOULOS, “Brief Review of Methods for the Design and Synthesis Optimisation of Energy Systems”, Int.J. Applied Thermodynamics, Vol.5, On pager(s): 151-160, December 2002.

[102] Kennedy *et al*, “Particle swarm optimisation”, IEEE int'l conf. on neural networks Vol. IV, On pager(s) 1942-1948, 1995.

[103] “Particle swarm optimisation”, http://en.wikipedia.org/wiki/Particle_swarm_optimisation, accessed at 13/08/2008.

[104] Kennedy, J, “The behavior of particles”, Evolutionary Programming VII: Proceedings of 7th Annual Conference on Evolutionary Programming, Springer-Verlag, 1998.

[105] Sol M. Shatz *et al*, “Task allocation for maximizing reliability of a distributed system using hybrid particle swarm optimisation”, Journal of Systems and Software, Pages 724-735, 1992.

[106] Peng Yeng, “A particle swarm optimisation approach to the nonlinear resource allocation problem”, Applied mathematics and computation, vol. 183, On pages: 232-242, 2006.

[107] Eghbal *et al*, “Application of metaheuristic methods to reactive power planning: a comparative study for GA, PSO and EPSO”, Systems, Man and Cybernetics, 2007, On page(s): 3755-3760, Oct, 2007.

[108] http://www.swarmintelligence.org/, accessed at 13/08/2008.

[109] “National Science Foundation (NSF). Office of Cyberinfrastructure (OCI). Retrieved Jan. 8, 2007”, http://www.nsf.gov/oci, accessed at 22/09/2008.

[110] “CERN. Grid Cafe - The place for everybody to learn about the Grid”, http://gridcafe.web.cern.ch/gridcafe/, accessed at 15/09/2008.

[111] F. Berman *et al*. “Grid Computing: Making the Global Infrastructure a Reality”, Pages: 1012, 2003.

[112] D. Simeonidou *et al* “An Optical Network Infrastructure Suitable for Global Grid Computing”, Selected Papers from the TERENA Networking Conference, June 2004.

[113] Ian Foster *et al*, “The Anatomy of the Grid”, Cluster Computing and the Grid, 2001. Proceedings. First IEEE/ACM International Symposium on , May 2001.

[114] N. Sakamoto *et al*, “Grid Computing Solutions for Artificial Neural Network-based Electricity Market Forecasts”, 2006 International Joint Conference on Neural Networks, Page(s):4382 - 4386 July 16-21, 2006.

[115] Ian Foster, “Computing Grids”, http://www.globus.org/alliance/publications/papers/chapter2.pdf, accessed at 12/08/2008.

[116] “Issue paper for working group “service”, The economy in Europe: Brusell, 1-2 March, 2001, http://ec.europa.eu/enterprise/events/e-economy/doc/services_paper.pdf, accessed at 22/09/2008.

[117] Liang Chen, “Web Service Orchestration with BPEL”, Proceedings of the 28th international conference on Software engineering, Pages: 1071 – 1072, 2006.

[118] R. Chinnici *et al*, “Web services description language (wsdl). Working draft, World Wide Web Consortium”, Aug. 2004, http://www.w3.org/TR/wsdl20, accessed at 13/10/2008.

[119] M. Gudgin *et al*, “Simple object access protocol (soap). Recommendation”, World Wide Web Consortium, http://www.w3.org/TR/soap12-part1, June 2003, accessed at 13/10/2008.

[120] “Orchestrating Web Services:The Case for a BPEL Server”, An Oracle White Paper, June 2004.

[121] “Web service”, http:/en.wikipedia.org/wiki/Web_service, accessed at 18/02/2009.

[122] T. Bray *et al*, “Extensible Markup Language. Recommendation”, http://www.w3.org/TR/1998/REC-xml-19980210, WorldWide Web Consortium, Mar. 1998, accessed at 13/10/2008.

[123] A. Nadalin *et al*. “Ws security. Technical report”, OASIS, Mar. 2004.

[124] “Virtual Private Network Architectures—Comparing Multiprotocol Label Switching, IPSec, and a Combined Approach”, http://www.cisco.com/warp/public/cc/so/neso/vpn/vpnsp/solmk_wp.pdf, accessed at 10/09/2008.

[125] http://linux.soft5000.com/dlf_5167.html, accessed 15/09/2008.

[126] E. Rosen *et al*, “BGP/MPLS VPNs”, IETF Request for Comments: 2547, March 1999.

[127] J. Case *et al*, “RFC 1157 - Simple Network Management Protocol”, RFC 1157.

[128] Phillips, C., “Flexible distributed testbed for high performance network evaluation”, Testbeds and Research Infrastructures for the Development of Networks and Communities, 2006. TRIDENTCOM 2006, Volume , Issue , 1-3, March 2006.

[129] “Web Services Description Language (WSDL) 1.1”, W3C Note 15 March 2001, http://www.w3.org/TR/wsdl accessed at 04/10/2008.

[130] http://www.w3.org/TR/soap/, accessed at 04/10/2008.

[131] “Axis C++ Server User's Guide”, http://ws.apache.org/axis/cpp/serveruser-guide.html, accessed at 11/10/2008.

[132] “Tomecat”, http://tomcat.apache.org/, accessed at 23/11/2008.

[133] http://ethereal.osmirror.nl/, accessed at 1/12/2008.

[134] Edwin S . H . Hou *et al*, “A Genetic Algorithm for Multiprocessor Scheduling”, IEEE Transactions On Parallel And Distributed Systems. Vol. 5, No. 2, February 1994.

[135] Kennedy J. *et al*,“Swarm Intelligence”, Morgan Kaufmann, 2001.

[136] http://en.wikipedia.org/wiki/PSO, accessed at 25/07/2008.

[137] http://jswarm-pso.sourceforge.net/, accessed at 2/12/2008.

[138] Lei Zhang1 *et al*, “A Task Scheduling Algorithm Based on PSO for Grid Computing”, International Journal of Computational Intelligence Research, ISSN 0973-1873 Vol.4, No.1 (2008), pp. 37–43.

[139] http://jgap.sourceforge.net/, accessed at 1/12/2008.

[140] http://en.wikipedia.org/wiki/Fitness_proportionate_selection, accessed at 2/12/2008.

[141] Grefenstette, J.J, “Optimisation of Control Parameters for Genetic Algorithms”, Systems, Man and Cybernetics, IEEE Transactions on, Volume 16, Issue 1, Jan. 1986 Page(s):122 – 128.

[142] Donald A. Sofge, “Using Genetic Algorithm Based Variable Selection to Improve Neural Network Models for Real-World Systems”, Proceedings of the 2002 International Conference on Machine Learning, 2000.

[143] A. Tuson *et al*, “Adapting operator settings in genetic algorithms”, Evolutionary Computation, vol. 6, no. 2, pp. 161–184, 1998.

[144] H. E. Aguirre *et al*, “Genetic algorithms on nk-landscapes: Effects of selection, drift, mutation, and recombination”, Lecture Notes in Computer Science, vol. 2611, pp. 131–142, January 2003.

[145] S. Uyar *et al*, “A gene based adaptive mutation strategy for genetic algorithms”, Lecture Notes in Computer Science, vol. 3103, pp. 271–281, January 2004.

[146] A. Acan, “Mutation multiplicity in a panmictic two-strategy genetic algorithm”, Lecture Notes in Computer Science, vol. 3004, pp. 1–10, January 2004.

[147] B. Djurisic *et al*, “Continuous optimisation using elite genetic algorithms with adaptive mutations”, Lecture Notes in Computer Science, vol. 1585, pp. 365–372, January 1999.

[148] Aggarwal, M *et al*, “Genetic algorithm based scheduler for computational grids”, High Performance Computing Systems and Applications, 2005. HPCS 2005. 19th International Symposium on Volume, 15-18 Page(s): 209 – 215, May 2005.

[149] http://java.sun.com/j2se/1.3/docs/api/java/lang/Math.html#random(), accessed 10/10/2008.

[150] C. E. Perkins *et al*,“Ad hoc On-demand Distance Vector (AODV) Routing”, IETF MANET Internet-draft, Feb. 2003.

[151] D. B. Johnson *et al*, “The Dynamic Source Routing (DSR) Protocol for Mobile Ad hoc Networks”, IETF MANET Internet-draft, Apr. 2003.

[152] H. Hassanein *et al*, “Routing with Load Balancing in Wireless Ad hoc Networks”, in Proc. ACM MSWiM, 2001, pp. 89–96.

[153] S.-J. Lee *et al*, “Dynamic Load-Aware Routing in Ad hoc Networks”, in Proc. IEEE ICC, 2001, pp. 3206–3210.

[154] Y. Yoo *et al*, “A Simple Load-Balancing Approach in Cheat-Proof Ad-Hoc Networks”, in Proc. IEEE GLOBECOM, 2004, pp. 3573–3577.

[155] L. Buttyan *et al*, “Enforcing Service Availability in Mobile Ad-Hoc Networks”, in Proc. ACM MobiHoc, 2000, pp. 87–96.

[156] http://www.mathworks.com/products/matlab/, accessed at 23/02/2009.

[157] http://en.wikipedia.org/wiki/Exponential_function, accessed at 14/05/2008.