Contributions to Solving Backbone VLAN Identifier (BVID) Problem and Optimizing Control-Plane Traffic in Service Provider Transport Networks

Submitted in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

of the

Indian Institute of Technology Bombay, Mumbai, India
and
Monash University, Australia
by

Deval Arunkumar Bhamare

(IITB ID: 10405402

[redacted]

Supervisors:

Prof. Ashwin Gumaste (IIT Bombay)

Prof. Mohan Krishnamoorthy (Monash University)

The course of study for this award was developed jointly by
the Indian Institute of Technology, Bombay and Monash University, Australia
and given academic recognition by each of them.
The program was administered by the IITB-Monash Research Academy

(Year 2014)

Copyright Notices

Notice 1

Notice 2

Approval Sheet

This thesis entitled **Optimization for High-Speed Networks and Control Plane Architecture** by **Deval Bhamare** is approved for the degree of Doctor of Philosophy.

Examiner(s)

_____

_____

_____

Supervisor(s)

_____

_____

Chairman

_____

Date: _____

Place: _____

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. Please note that the core chapters that are chapter number 2, 3 and 4 of the thesis (text and diagrams) have appeared in [8, 105], [106] and [107] respectively. The work presented in chapter number 5 is under preparation for submission. Please refer to the Section 'List of Publications' for more details. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

_____

(Signature)

Deval Arunkumar Bhamare

(Name of the student)

10405402

(IITB Roll No.)

[redacted]

[redacted]

Date: _____

Abstract

Internet traffic has grown at the rate of 40-100% per year and is only expected to grow at the same rate or more aggressively in the future. Due to the enormous growth in IP traffic volume, network operators are confronted by the unprecedented challenge of accommodating the IP traffic volume in already deployed networks in a short time-span. In addition to this, network operators are observing stringent latency and reliability requirements. The overall effect of these problems has been sub-optimal utilization of network resources resulting in higher network-wide capital and operational expenditures, along with lesser revenues. Hence, there is a need to optimize network cost while accommodating maximum traffic volume in the network and simplify the existing Carrier Ethernet Networks.

In this thesis, we analyze high-speed telecommunication networks, with a focus on cost and performance optimization of high-speed Carrier Ethernet Networks. The first problem we focus on is that of optimization, in which, we try to minimize the number of network identifiers in the routers used in high-speed networks. Since available identifiers in the hardware are limited (due to the limited number of bits allocated to the identifiers), and requests for connections in the networks are increasing, it is desirable that such identifiers are re-used so as to maximize the number of connection requests satisfied in a network. In addition, we try to minimize the cost of the contemporary high-speed networks so that the internet service providers can benefit from maximum profits by reducing *Capital Expenditures* (CAPEX) and *Operational Expenditures* (OPEX). We focus, specifically, on reducing the total cost of the network interfaces at different network layers, satisfying all the traffic demands. In this pursuit, we study multilayer optimization from the perspective of deploying services so as to reduce the CAPEX in provider networks. To this end, we propose a 3-layer network hierarchical model based on IP, OTN and DWDM technologies. The goal of this work is to ascertain the type of traffic and how much of the traffic would require to be routed through a particular layer of the network to reduce the total cost of ownership.

In the latter part of the research, we seek to answer an interesting question—how does Carrier Ethernet perform for *Virtual Machines* (VM) migration in data-center and cloud environments? The question arises since VMs form the central processing entity in data-centers and are crucial to facilitating cloud computing environments as well as the recent enhancements in Carrier Ethernet, which can be used as an efficient transport mechanism in DC/Cloud environments. To this end, we perform an extensive simulation study measuring the performance of VM migration over both flavors of Carrier Ethernet—namely PBB-TE and MPLS-TP. Our study concludes with an examination of the feasibility of Carrier Ethernet as a transport technology in data-centers and clouds.

To extend the optimization work in high-speed carrier networks, we also focus on the implementation of the *Centralized Control Plane* for managed networks and try to optimize the Control traffic in the network. The overlapping of the control plane and the data plane in contemporary Carrier Ethernet (CE) networks leads to fragile and unmanageable networks. CE networks using packet technologies ought to be more manageable, scalable and robust. We propose an approach of a *Centralized Control Plane* to overcome the problem mentioned earlier. However, the control traffic generated due to the interaction between the centralized control plane and network elements increases exponentially as the number of nodes in the network increase and adds to the latency in the data traffic flow. The problem of control traffic overhead in managed networks is analyzed using an appropriate simulation model. A scheme to divide the network into smaller sub-networks is proposed so that total control traffic is always below a certain threshold. An ILP model and a heuristic approach are presented to strengthen the claim. The work is divided into separate chapters, as outlined in the Introduction.

Contents

List of Tables

Table of Acronyms

Acronym	Full-Form
PBB-TE	*Provider Backbone Bridging–Traffic Engineering*
LAN	*Local Area Network*
VID	*VLAN ID*
BVID	*Backbone VLAN ID*
MAC	*Media Access Control*
SONET/SDH	*Synchronous Optical Networking/Synchronous Digital Hierarchy*
MPLS-TP	*Multi-Protocol Label Switching-Transport Profile*
Backbone MAC	*BMAC*
CSMA	*Carrier Sense Multiple Access*
STP	*Spanning Tree Protocol*
STAG	*service provider TAG*
CTAG	*Customer TAG*
tagged VLAN	*IEEE 802.1Q Standard*
Q-in-Q standard	*IEEE 802.1ad Standard*
ISID	*Backbone Service Instance ID*
ILP	*Integer Linear Program*
IEEE	*Institute of Electrical and Electronics Engineers*
IETF	*Internet Engineering Task Force*
ITU	*International Telecommunication Union*
TCAM	*Ternary Content-Addressable Memory*
OTN	*Optical Transport Network*
DWDM	*Dense Wavelength Division Multiplexing*
ODU	*Optical Data Unit*
ROADM	*Reconfigurable Optical Add-Drop Multiplexer*
IP	*Internet Protocol*
VM	*Virtual Machine*
CE	*Carrier Ethernet*
TDM	*Time Division Multiplexing*
CAPEX	*Capital Expenditures*
OPEX	*Operational Expenditures*
PDR	*Page Dirty Rate*
RAM	*Random Access Memory*
SAN	*Storage Area Network*
NAS	*Network Attached Storage*
NMS	*Network Management System*
CFM	*Connectivity Fault Management*
CESR	*Carrier Ethernet Switch-Router*
SNMP	*Simple Network Management Protocol*
OAM&P	*Operation, Administration, Maintenance and Provisioning*
GMPLS	*Generalized Multiprotocol Label Switching*
OSPF	*Open Shortest Path First*
EON	*European Optical Network*
NMS	*Network Management System*
CSAHLP	*Capacitated Single Allocation Hub Location Problem*

List of Figures

Chapter 1

Introduction

Carrier Ethernet has rapidly advanced to become an important technology in metro transport. The migration of Ethernet from an infrastructural solution to a transport solution has been brought about by the emergence of data-networking and rich multimedia applications. Despite the phenomenal growth of the telecom service industry, developing countries like India are the lowest users of broadband services. Also, the growth of Carrier Ethernet has been stagnant in the already deployed networks all over the world. This has been a result of the fact that networks have, traditionally, been designed for data traffic like file transfer, with no attention to end-to-end service delivery. However, the same networks are being used to deliver services comprising multimedia-rich applications like voice and video, which are growing exponentially. Due to the enormous growth in *IP* traffic volume, network operators and service providers are confronted by the unprecedented challenge of accommodating the *IP*

traffic volume in already deployed networks in a short time-span, while their revenues have been flat. It has been a major cause of the hindrance to broadband penetration in the developing countries and hindrance to the proliferation of Carrier Ethernet across the Internet. Hence, in this work, we try to optimize some of the parameters in high-speed networks, such as network identifiers in PBB-TE networks and network ownership cost in Multilayer Networks. We also propose a design approach for centralized control plane architecture with minimal control traffic overhead. The intended result is that, maximum service requests can be satisfied in the given network with minimum cost to the service providers with more simplified underlying networks.

In this pursuit, we analyze high-speed telecommunication networks, with a focus on cost and performance optimization. First, we attempt to maximize the number of services that can be provisioned in core networks with the limited number of Backbone VLAN Identifiers (BVIDs), so that Ternary Content-Addressable Memory (TCAM) utilization of network equipment is less. Since TCAM is an expensive component of network equipment, our work leads to significant cost savings with maximum number of service provisioning in the contemporary Carrier Ethernet networks. We specifically focus on the BVID problem in PBB-TE networks since this problem has not been addressed before and it may result in the failure of the PBB-TE standard. However, a problem in MPLS-TP networks which is similar in nature but different in principal has already been studied [15, 19]. Further we study multilayer optimization from the perspective of deploying services, to reduce the capital expenditure in provider-networks. We then try to optimize the cost of the contemporary high-speed networks so that internet-service providers can have the benefit of maximum profits by reducing the total network ownership cost. Previous research works either focus on optimization in IP/MPLS over WDM networks, IP over SONET/SDH or on the survivability of IP/MPLS over WDM networks. However, we focus on 3-layer network architecture from the perspective of network interfaces and their implementation costs. We also focus on performance optimization of high-speed Carrier Ethernet networks by proposing centralized control plane architecture. We try to minimize the Control Traffic overhead in such centrally managed networks and propose a scheme for the optimal placement of controllers. We present simulation results to

demonstrate the improvement in the centrally managed networks from the perspective of control traffic overhead. We believe that the combined work presented in this dissertation will enable more efficient traffic-engineering with end-to-end service provisioning in the contemporary networks and proliferation of Carrier Ethernet.

The organization of the thesis is as follows. Please note that the core chapters that are chapter number 2, 3 and 4 of the thesis (text and diagrams) have appeared in [8, 105], [106] and [107] respectively. Work presented in chapter number 5 is under preparation for submission. Please refer to the Section 'List of Publications' for more details.

1.1 Carrier Ethernet and BVID allocation Problem

Contemporary carrier services are IP-based and Carrier Ethernet has been a natural choice for the deployment of IP-based services [18, 19]. There have been two critical standardization processes that have brought about the advent of Carrier Ethernet—the work in the IEEE called the Provider Backbone Bridging-Traffic Engineering or PBB-TE [2] and the joint work in the IETF and ITU called the Multi-Protocol Label Switching-Traffic Profile or MPLS-TP [11]. The former approach of PBB-TE makes use of the existing Ethernet internetworking, switching and bridging architecture to create a new carrier-grade transport philosophy, while the latter approach of MPLS-TP, undercuts MPLS features to develop it into a transport technology. Both approaches are distinct protocol-wise, but similar conceptually. However, the PBB-TE approach has gone through a better evolution resulting in a wide number of 802.1 standards (such as 802.1ad, 802.1ah, 802.1Qay, 802.1as and 802.1ag), as explained briefly in the next paragraph [2-7]. In comparison, the MPLS-TP work has gained better acceptance due to the massive installed base of MPLS in metro and core regions.

Conventional CSMA Ethernet relies on the principle of MAC learning and Spanning Tree Protocol (STP) to enable end-to-end communication. Given the absence of any priority information in Ethernet frames and the likelihood of huge spanning trees, the IEEE 802.1Q (tagged VLAN) standard was introduced. Now, an Ethernet domain

could be divided into sub-domains or VLANs, or VLANs could simply be tagged to produce 8 QoS levels [6]. A single VLAN had 12-bits to denote the VLAN ID (VID) resulting in 4096 (effectively 4094 with 2 reserved) tags, and 3-bits to denote 8 priority levels—meaning that, the service provider network could only support 4094 unique service instances. The problem is further aggravated when one considers that customers connected to a service provider 802.1Q network may also have their own VLANs (within customer premises)—which lead to the use of separate or stacked VLANs—one for the customer (CTAG) and one for the service provider (STAG).

This led to the Q-in-Q standard as defined by the IEEE 802.1ad. This solved the problem of demarcation between customers' and providers' VLANs. However, the problem of security, from the provider's perspective, persisted because the provider had to learn the customer's MAC address, thus exposing the providers' edge and core bridges to potential MAC security violations. To facilitate proper demarcation between the customer and the provider, the 802.1ah or provider-backbone-bridging standard was introduced—allowing the customer frame with the CTAG and provider-added STAG to be fully encapsulated in a provider backbone bridge frame with a Backbone Service Instance ID (I-SID), provider network specific VLAN-ID (BVID) and backbone MAC (BMAC). The ISID is a 24-bit VID (as opposed to the CTAG, STAG or BVID which are all 12-bit values). The ISID maps a service to a network-specific VID, but for forwarding purposes, the ISID needs to be mapped to a generic switch (one that is backward compatible with 802.1Q)—for which a 12-bit BVID is introduced. The BVID along with the BMAC address is used by switches for forwarding purposes, leading to a network-wide unique 60-bit address. Multiple ISIDs may be mapped to a BVID (in a sense that, services which are graphically overlapping and subtending to the same destination may be assigned the same BVID), since the number of BVIDs is restricted to 4094 [5]. Due to the dependence on STP and MAC learning, PBB results in probabilistic latency, and hence, is not carrier-class. To resolve this issue, PBB-TE was introduced. In PBB-TE, MAC learning and STP are both turned OFF and BVIDs are assigned manually.

As a part of this Ph.D. thesis, we have analyzed an interesting problem in the PBB-TE domain, which, if not solved, has the potential to make PBB-TE fail from a provisioning and implementation perspective. This problem is important due to the already accomplished standardization of PBB-TE and the direct relevance it has to high-speed networks that are currently being built. PBB-TE relies on the assignment of the BVIDs, which is further dependent on customer and service provider VLAN tags, a service provider MAC address and an intermediate instantiation service tag. Given the limited availability of BVIDs in a network on account of the basic tag format, it is desirable to re-use tags to facilitate larger service instance provisioning. This problem is further reduced to a constrained optimization problem. We have proposed four different heuristic algorithms to optimally allocate and re-use the BVIDs in the PBB-TE networks. The results obtained by implementing the heuristic algorithm and comparing it with the ILP are showcased.

1.2 Multilayer Optimization in High-Speed networks

Contemporary Service provider networks are multilayered, where two or more layers together perform the task of routing and packet forwarding. For example, IP over SONET/SDH or IP/MPLS over DWDM are practical deployments in contemporary networks (refer to Fig 1.1 below). Service provider networks are becoming increasingly complex with multi-domain and multilayer capabilities. Multilayer optimization has been proposed as a mechanism for planning provider-networks focused on optimizing revenue and reducing capital expenditure [37-40]. To this end we propose a 3-layer network hierarchical model based on *IP*, *OTN* and *DWDM* technologies. A significant amount of work has been conducted on the design of optimal multilayer networks. However, the focus has primarily been on design and optimization in two-layered networks. Previous works either focus on optimization in IP/MPLS over WDM networks, IP over SONET/SDH or on the survivability of IP/MPLS over WDM networks. Furthermore, additional work considers the OTN layer in the optimization model, but models this layer as an embedded one in the WDM layer and not as a separate layer. The work presented in [108] has considered OTN layer from a capacity

planning perspective in conjunction with the 3-layer network model, which we have aptly modeled from the perspective of a service-provider networks.

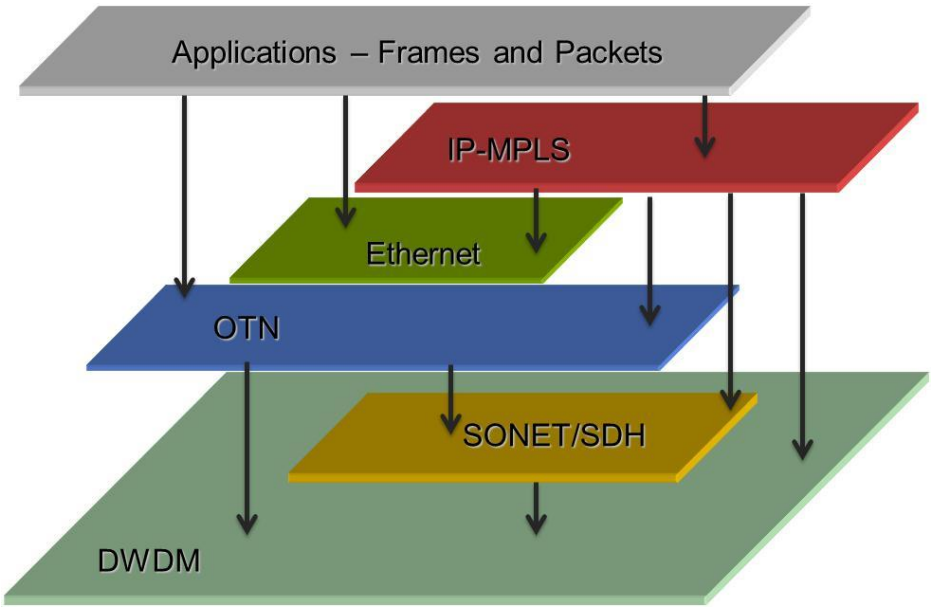


Fig. 1.1. Contemporary Multilayer Network Architecture.

In this work, we consider an optimization problem in a 3-layered network comprising IP as the topmost layer followed by the OTN layer that acts as opto-electro-opto (OEO) cross-connect and then, the DWDM layer, which is a physical layer. A network capacity planning problem, specifically, for a 3-layer network hierarchy comprising an IP layer, the OTN layer and the optical DWDM layer is considered [38, 39]. We demonstrated that network cost reduces significantly in 3-layer networks where OTN acts as an intermediate cross-connect layer between the IP and DWDM layers. An *Integer Linear Program* (ILP) formulation is presented to solve the problem for static traffic (connection) demands in the network. We aim to minimize the total interface deployment cost in a network where maximum traffic demands are satisfied. In addition to the ILP, we have proposed a heuristic approach to minimize the total interface deployment cost for dynamic traffic demands in large networks (up to 1000-nodes). The heuristic runs within reasonable computational time. The results obtained by implementing the heuristic algorithm and comparing it with the ILP are showcased.

1.3 Transport Technology Choices for Virtual Machines (VMs) in Data-Center and Clouds

Cloud computing and data-centers are expected to dominate much of metro transport and strongly affect IT-applications and businesses. A paradigm shift has been happening from local computation to cloud computing. One technology that has empowered this paradigm shift is virtualization. Virtualization brings with it capabilities of isolating, consolidating and migrating workload. A sample VMware vCenter architecture is shown in Fig 1.2 below [67]. With the ability to move virtual machines between hosts, live migration has been a core feature of virtualization. Virtual Machine (VM) migration has been a key in making data-centers and clouds a reality from the point of view of the scalability and the availability of applications [61, 64].

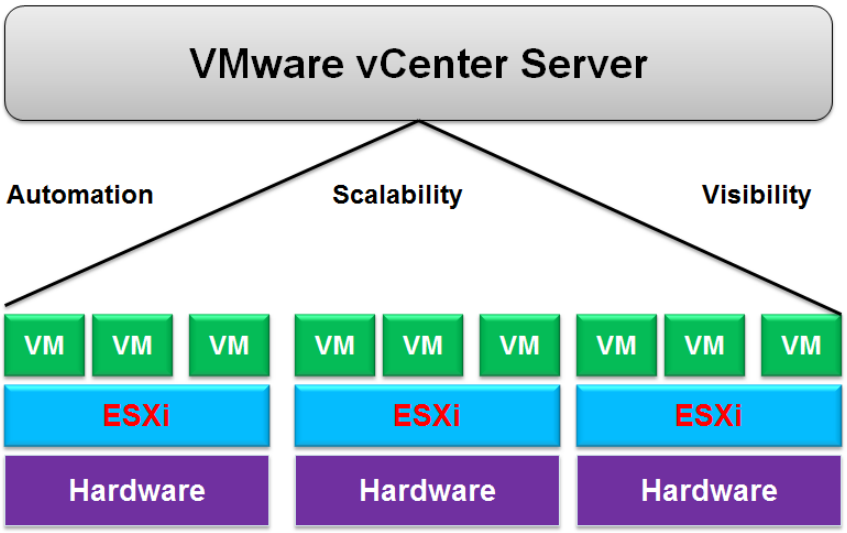


Fig. 1.2. VMware vCenter Architecture.

The underlying network that connects the disparate servers and storage devices to the rest of the network plays an important role in facilitating migration of content (of the VMs) across the servers; it also enables ease of connectivity for the end-user. The underlying network fabric that binds disparate servers, storage entities and other IT appliances has a stronger role to play when we move from a single data-center (DC) to a collection of DCs—such as in a cloud environment. Traditionally, SONET/SDH systems, and more recently, IP/MPLS have served as an effective interconnection

fabric. SONET/SDH is primarily plagued with cost and TDM-related rigidity in bandwidth provisioning, while IP/MPLS does not manifest the carrier-class features that are necessary for transport and is also seen as an expensive technology for cloud/DC domains. Deploying the recently standardized Carrier Ethernet technology seems an attractive option from both, cost and performance perspective [87]. There are two flavors of Carrier Ethernet that are available today—the MPLS-TP (as in RFC 5317) and the PBB-TE (as in IEEE 802.1Qay) [2, 11]. In this work, we propose the use of Carrier Ethernet as a transport technology specific to DCs and cloud environments. We have proposed the use of Carrier Ethernet in the data-center/cloud environments through an extensive simulation model.

1.4 Models, Algorithms and Solution Methods for Centralized Control Planes to Optimize Control Traffic Overhead

The overlapping of the control plane and the data plane in contemporary Carrier Ethernet (CE) networks leads to fragile and unmanageable networks. CE networks using packet technologies need to be more manageable, scalable and robust. It is recommended that the control and management planes in CE networks be decoupled from the forwarding and routing planes to overcome this problem [81]. As a result, we investigate the possibility of a centralized control plane to manage CE networks. The centralized control plane is a networking paradigm that abstracts and centralizes the control information of the network from the underlying distributed data forwarding infrastructure [77-80]. A logical view of centralized controlling entity is show in the Fig. 1.3 below [75]. In this work, we discuss the need for a centralized control plane in CE networks and focus on the engineering and architectural issues related to the design of a centralized control plane; we present an approach for the implementation of the Network Management System (NMS) that is central to the working of CE networks.

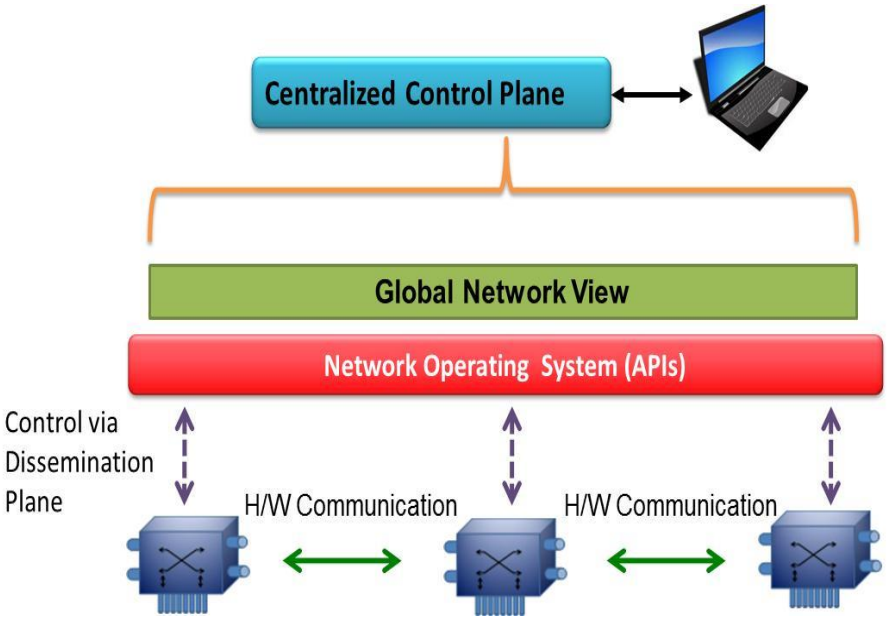


Fig. 1.3. Logical view of Centralized Controlling Entity.

Furthermore, the problem of control traffic overhead in managed networks is analyzed using an appropriate simulation model. A scheme to divide the network into smaller sub-networks is proposed so that total control traffic is always below a certain threshold. An ILP for the controllers' placement in the partitioned network is presented to minimize the total control traffic, total controllers' implementation cost and overall response time in the network. Since the ILP can solve the problem for networks with a limited number of nodes, two heuristic approaches are presented for larger and real-time provider-networks, while preserving the correctness of the ILP solution. The results have been showcased.

Chapter 2

Carrier Ethernet and BVID allocation Problem

2.1. Journey of Carrier Ethernet

Carrier Ethernet is being heralded as the next major innovation in transport technology and is being sought after in both, the enterprise and the service provider worlds. The migration of Ethernet from an infrastructural solution to a transport solution has been brought about by the emergence of data-networking and rich multimedia applications that are not suited to legacy SONET/SDH [1] transport. There have been two critical standardization processes that brought about the advent of Carrier Ethernet—the work in the IEEE called the Provider Backbone Bridging - Traffic Engineering [2] and the joint work in the IETF and ITU called the Multi-Protocol Label Switching-Traffic

Profile or MPLS-TP [3]. The former approach of PBB-TE makes use of the existing Ethernet internetworking, switching and bridging architecture to create a new carrier-grade transport philosophy; the latter approach of MPLS-TP undercuts MPLS features to develop it into a transport technology [4]. Both approaches are distinct protocol-wise but similar conceptually, though the PBB-TE approach has gone through better evolution, resulting in a wide number of 802.1 standards (such as 802.1ad, 802.1ah, 802.1Qay, P802.1as and 802.1ag). In comparison, the MPLS-TP work has gained better acceptance due to the massive installed base of MPLS within metro and core regions.

The focus in this report is an acute problem in the PBB-TE domain that has the potential to make PBB-TE fail completely from a provisioning and implementation perspective. This problem is important due to the already accomplished standardization of PBB-TE and the direct relevance it has to high-speed networks that are currently being built. To understand this problem, first, we briefly discuss the evolution of the Ethernet technology and the working of PBB-TE. Please note that the pictures for the frame formats have been re-created but essentially the same as the pictures which appeared in IEEE standards such as [2, 5-7].

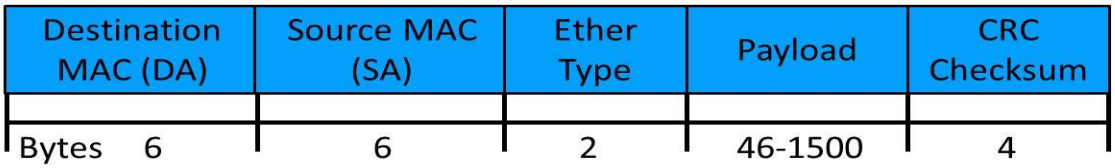


Fig. 2.1 Carrier Ethernet general frame format.

2.1.1 CSMA Ethernet

- Conventional CSMA Ethernet relies on the principle of MAC learning and Spanning Tree Protocol (STP) to enable end-to-end communication.
- Given the absence of any priority information in Ethernet frames and the likelihood of huge spanning trees, the IEEE 802.1Q (tagged VLAN) standard was introduced.

2.1.2 IEEE 802.1Q

- At this stage, an Ethernet domain could be divided into sub-domains or VLANs, or VLANs could simply be tagged to produce 8QoS levels.
- A single VLAN had 12-bits to denote the VLAN ID (VID) resulting in 4096 (effectively 4094 with 2 reserved) tags, and 3-bits to denote 8 priority levels—meaning that the service provider network could only support 4094 unique service instances [6].
- The problem was further aggravated considering that customers connected to a service provider 802.1Q network may also have had their own VLANs (within customer premises)—which led to the use of separate or stacked VLANs—one for the customer (CTAG) and one for the service provider (STAG). This led to the Q-in-Q standard as defined by the IEEE 802.1ad [7].

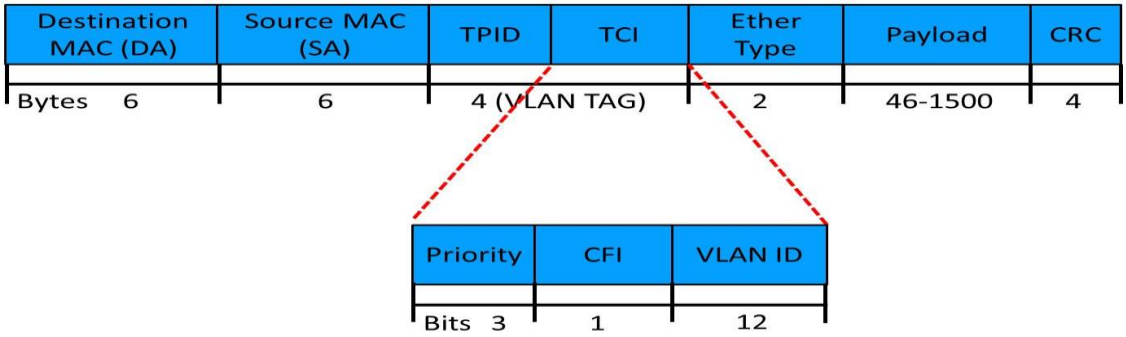


Fig. 2.2 802.1Q frame format.

2.1.3 IEEE 802.1ad

- 802.1Q solved the problem of demarcation between customers and providers VLANs.
- However, the problem of security, from the provider's perspective persisted since the provider had to learn the customer's MAC address, thus, exposing the providers' edge and core bridges to potential MAC security violations.

To facilitate appropriate demarcation between the customer and the provider, the 802.1ah or provider-backbone-bridging [5] standard was introduced.

2.1.4 IEEE 802.1ah (PBB)

- 8201.1ah allows the customer frame with the CTAG and provider-added STAG to be fully encapsulated in a provider backbone bridge frame with a Backbone Service Instance ID (I-SID), provider network-specific VLAN-ID (BVID) and backbone MAC (BMAC).
- The generation of BVID depends on an intermediate instantiation service ID called ISID that is service-specific and network-wide unique. The ISID is a 24-bit VID (as opposed to the CTAG, STAG or BVID, which are all 12-bit values).
- The ISID maps a service to a network-specific VID, but for forwarding purposes, the ISID needs to be mapped to a generic switch (one that is backward compatible with 802.1Q)—for which a 12-bit BVID is introduced.
- The BVID along with the BMAC address is used by switches for forwarding purposes leading to a network-wide unique 60-bit address.
- Multiple ISIDs may be mapped to a BVID—as the number of BVIDs is restricted to 4094. The BVID mapping in a PBB network is dependent on a Multiple Spanning Tree Protocol (MSTP) [4-7].
- Due to the dependence on STP and MAC learning, PBB results in probabilistic latency, and hence, is not carrier-class. To absolve this, PBB-TE was introduced.

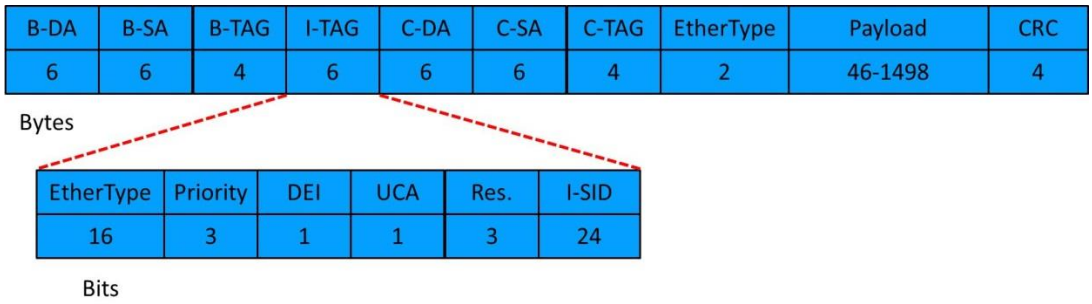


Fig. 2.3 IEEE 802.1ah frame formats.

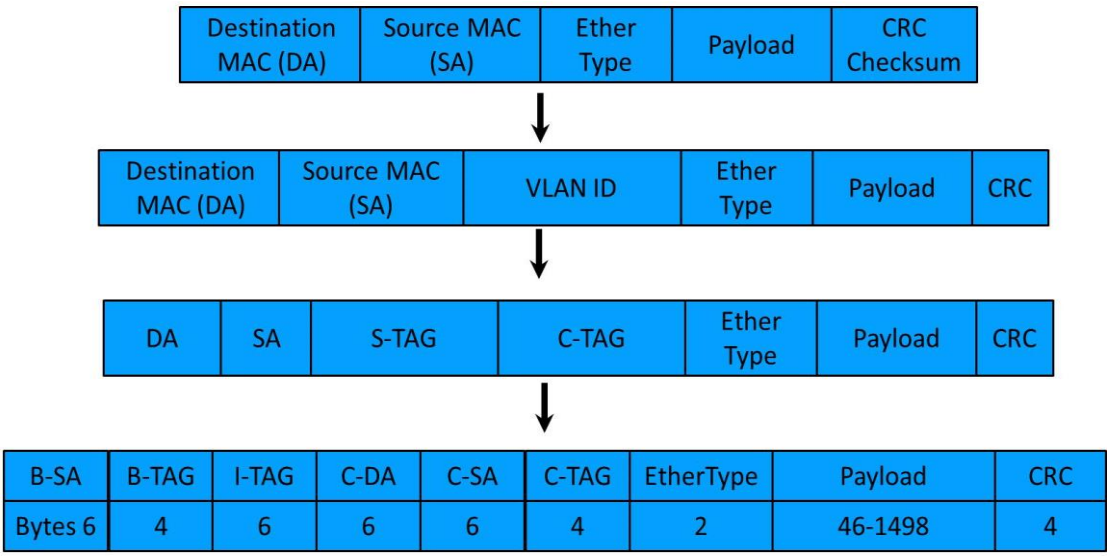


Fig. 2.4 Carrier Ethernet frame format evolution.

2.1.5 IEEE 802.1QaY (PBB-TE)

- In PBB-TE, MAC learning and STP are both turned OFF and BVIDs are assigned manually [2].
- To conserve BVIDs (restricted to 4094), they are often re-used, thus mapping several ISIDs to a single BVID.
- The standard does not consider the specific algorithm by which a BVID is assigned to a particular service and maximizing of the number of services in a network-graph, given the fixed number of BVIDs.

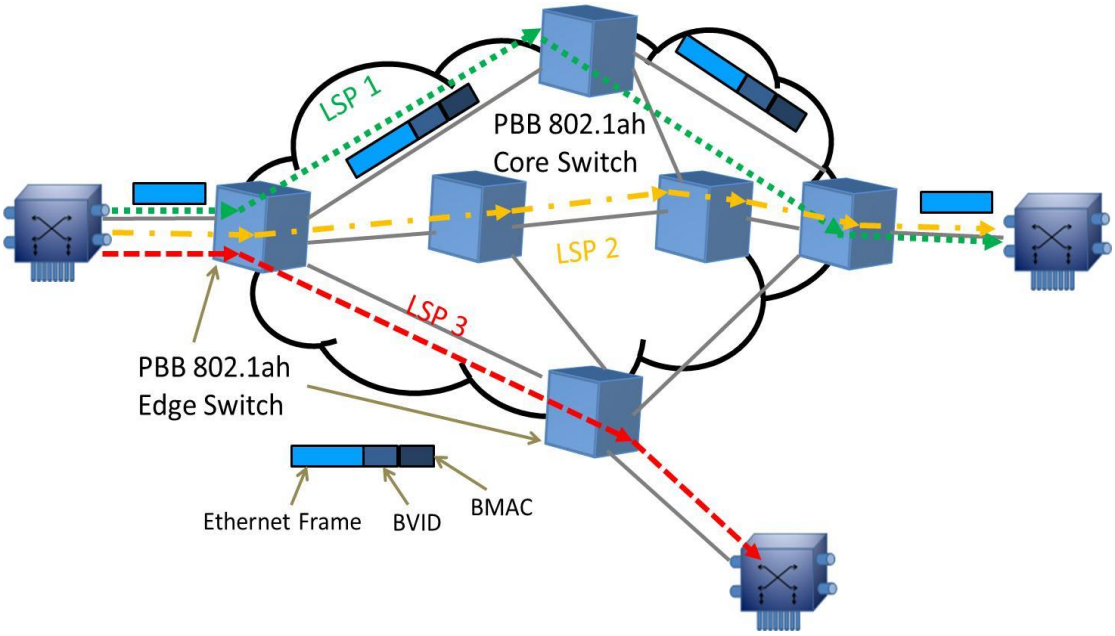


Fig. 2.5. Configuring Ethernet trunks (LSPs) with PBB (802.1ah).

2.1.6 MPLS/MPLS-TP

Parallel to the IEEE standardization process, ITU too was developing a carrier-centric mechanism to transport Ethernet or equivalent Layer-2 transport services. The underlying assumption guiding ITU's development centered on the abundance of IP/MPLS routers spanning the core of most metropolitan domains with Ethernet supportive interfaces. To transport Ethernet services, the ITU initiative is called transport-MPLS—derived from MPLS, making it connection-oriented and carrier centric. MPLS-TP is somewhat similar to PBB-TE, in the sense that both are connection-oriented and the control planes are assumed to be independent of the data plane, much like the optical supervisory channel in metro WDM networks. MPLS-TP is similar to MPLS in certain aspects, such as the use of labels and the method of forwarding; however, there are also differences. Among these, the primary difference is in the manner in which labels are used: while MPLS does label switching along multiple paths for a single flow and assumes bidirectional paths, MPLS-TP neither supports bidirectional paths nor the spreading of the flow into several disjoint paths. MPLS-TP sets up one tunnel from source to destination (no stopping at the penultimate node as in case of MPLS). This tunnel, due to its deterministic nature of bandwidth and delay,

provides a carrier class solution for the transport of any payload (not just Ethernet) [3, 4].

2.2 BVID Allocation in PBB-TE Networks

2.2.1 Introduction

The approach of PBB-TE makes use of the existing Ethernet switching and bridging framework to create a new carrier-grade transport philosophy. The approach of MPLS-TP, relaxes MPLS features to develop a transport technology [4]. Both approaches are, protocol-wise, distinct, but conceptually similar. The PBB-TE approach has gone through a longer evolution resulting in a wide number of 802.1 standards (such as 802.1ad or MAC bridges, 802.1ah or Provider Bridges, 802.1Qay or PBB-TE, P802.1as or shortest path bridging and 802.1ag or connectivity fault management). In comparison, the MPLS-TP work is better positioned due to the massive installed base of MPLS within metro and core regions. The ready standardization of PBB-TE has implied that equipment is available today for the deployment of PBB-TE-based networks.

In PBB-TE networks, spanning tree and MAC learning are switched-off. Ethernet frames are forwarded using network-wide VLAN identifiers called Backbone VLAN Identifiers or BVIDs that are used in conjunction with a manufacturer-assigned MAC address, called Backbone MAC (BMAC) or Backbone Destination address (B-DA) (Fig. 2.3, pg. 14). Though a combined 60-bit identifier is sufficient to achieve scale, in practice, a small number of BVIDs are actually allocated to the PBB-TE switches/routers to reduce the processing complexity and cost of the device. Given the limited availability of BVIDs in a network on account of the basic tag format (4-byte with 12-bits allocated as an identifier), it is desirable to reuse tags to facilitate larger service instance provisioning. Unplanned reuse of the BVIDs (as is the current case) results in conflicts in connection requests (ringing problem, as explained in Section 2.2.3). This problem has the potential to fail PBB-TE from the perspective of service provisioning and scalability, especially, when we consider that most Carrier Ethernet

deployments would be upgrades to SONET/SDH interconnected ring networks. The interconnected ring network presents a unique case, whereby, for purposes of load balancing two connections between a source in one ring and destination in another ring can possibly go through a common intersection point. The problem of BVID allocation typically arises when connection requests graphically (partially) overlap and subside at a common destination node but have different paths beyond the first overlapping segment/node. In such a situation, if two requests are assigned the same BVID, then the 60-bit BMAC+BVID identifier would be common for both paths. This implies that the switch that is responsible for forwarding the requests at the egress of the overlap would not be able to differentiate between the two requests. This would lead to a common entry, implying two different output ports in the PBB-TE table at the switch, resulting in the failure of the switch. This is the BVID allocation problem (described in detail in Section III). To consider an illustrative example of the BVID allocation problem, consider Table 2.1.

PBB-TE Traffic case	Destination BMAC	BVID	Scale	Use Cases
Non-overlapping	xx:xx:xx:xx:xx:xx Unique/Same	Don't Care	YES	Low Granularity-ELINE
Overlapping – Different Destination Different Source	00:01:23:45:67:89 00:01:23:45:67:9A Unique	Don't Care	YES	Low Granularity-ELAN
Overlapping – Different Destination Same Source	00:01:23:45:67:89 00:01:23:45:67:9A Unique	Don't Care	YES	E-TREE
Overlapping – Same Destination Different Source	00:01:23:45:67:89 00:01:23:45:67:89 Unique	Should be Unique	No	Large Enterprises and Mesh Networks
Overlapping – Same Destination Same Source	00:01:23:45:67:89 00:01:23:45:67:89 Unique	Should be Unique	No	Interconnected Rings and SDH/SONET

TABLE 2.1: Cases for BVID Allocation Problem.

As can be seen in Table-1, in the top 3 cases, when the source and destinations are unique, the BVID can be reused, resulting in no problems with scalability. However,

the last two cases are where the problem occurs. In these two cases, there is a graphical overlap for two connections between a common source-and-destination. This leads to the limitations in scalability when the BVID is reused, implying that the forwarding table has a common 60-bit entry for two output ports.

While the premise of this problem seems, at first, to be a corner case in networks, it turns out that this corner case is actually quite prevalent. Given that CE is being considered as a replacement for SONET/SDH networks, which in most cases manifests as interconnected rings, the deployments for CE are also interconnected rings. If the source is in one-ring and the destination node in another ring, then the CE ELINE that connects source to destination must pass through the common interconnection point. Now, if this source-destination pair has heavy traffic granularity requirement, there may be several ELINEs between the source and destination. Since both versions of CE (PBB-TE and MPLS-TP) do not support equal cost multiple paths as an automatic option, the *Network Management System* (NMS) will route some demands using a particular path through the interconnected rings, while other demands will be routed using the wrap-around paths in the same interconnected rings or mesh networks. These set of paths meet at the interconnection point and again diverge only to meet at the destination. This is the premise of the BVID allocation problem and it does happen that this is quite prevalent in interconnected ring and mesh networks—the bulk of metro topologies. We note that the work is about solving a practical problem that would immediately impact the deployment of Carrier Ethernet networks. The goal is to find an engineering solution that is efficient (in terms of implementation ease).

This part of the work discusses how to assign and reuse BVIDs to CE streams in a network. In particular, an optimal allocation strategy of BVIDs is discussed here. A related problem of service merging that results in an intrinsic deterrent to the reuse of BVIDs is also considered and solved using both an optimal solution (for the static/offline case) and a heuristic solution (the static as well as the dynamic/online case).

2.2.2 Related Work

In this work we propose the general BVID allocation problem by extending our preliminary work on this problem first proposed in [8]. Specifically, we have modified the Integer Linear Program (ILP) proposed in [8] to make it exhaustive by adding constraints that maintain the linearity of the formulation. In this extended version, we have discussed the complexity of the BVID Allocation problem and proved it to be NP-complete. The modified ILP has led to detailed results (up-to 40-node topology) that are useful for comparison with the heuristics as well as to serve as a benchmark for PBB-TE implementation. In [8], we have presented a preliminary heuristic to solve the BVID problem, while, in the extended version, we have presented four different approaches for BVID assignment. We have presented exhaustive results (up-to 200 nodes and 5000 connection requests) to gain better insights into the performance of the proposed heuristics and for a comparison between the different approaches. Results demonstrate the scalability of the algorithms and exhaustiveness of the results in practical scenarios. In addition, we have analyzed the complexity of the proposed heuristic algorithms.

To the best of our knowledge, this problem has not yet been considered. The adjacent problem of label assignment in MPLS networks [15, 19] has been considered. The problem in MPLS networks is different from PBB-TE networks in the sense that label swapping and merging is allowed in MPLS [20, 21, 23]. This means that it is easy to obtain a string of labels, value of which is unique for different overlapping paths. Hence, the BVID allocation problem and MPLS label assignment problems [31, 34], though similar in structure, are quite different in principle. A scheme for MPLS label space reduction is shown in [22, 32]. This scheme is interesting to note, as it tackles a similar problem. Since the work in [22] was prior to the standardization of PBB-TE, it does not consider the PBB-TE case which is different on account of the failure which occurs due to certain graphical properties. However, [22] concludes that with a large number of services, the performance of a network with labels (and hence also with tags) is adversely impacted unless there is an effective label space reduction technique deployed. This work, hence, gives us the necessary motivation for adopting BVID minimization to PBB-TE networks. Note that in MPLS, there is no limit to the addition

of labels. This is not the case in PBB-TE networks—where we are restricted to a single BVID tag.

2.2.3 BVID Allocation: A Primer and Example

To understand the BVID allocation problem, we briefly discuss the working of PBB-TE [2, 8]. Conventional *Carrier Sense Multiple Access* (CSMA) Ethernet relies on the principle of MAC learning and Spanning Tree Protocol (STP). Given the absence of any priority information in Ethernet frames and the likelihood of huge spanning trees, the IEEE 802.1Q (VLAN) standard was introduced. With 802.1Q, an Ethernet domain can be divided into sub-domains (VLANs), or VLANs could be tagged to produce 8 QoS levels [5-7].

A single 4-byte VLAN had 12-identifier bits to denote the VLAN ID (VID) resulting in 4096 (effectively 4094 with 2 reserved) tags, and 3-bits to denote 8 priority levels—implying that the service provider network could only support 4094 unique service instances. The problem was further aggravated considering that customers connected to a service provider 802.1Q network could also have had their own VLANs (within customer premises) —which led to the use of separate or stacked VLANs—one for the customer (CTAG) and one for the service provider (STAG). This led to the Q-in-Q standard (IEEE 802.1ad). Q-in-Q solved the problem of demarcation between customer and provider VLANs [7]. However, the problem of security from the provider perspective persisted as the provider had to learn the customer's MAC address, thus, exposing the providers' edge and core bridges. To facilitate proper demarcation between the customer and the provider, the 802.1ah or *provider-backbone-bridging* [5] was introduced—allowing the customer frame with the CTAG and provider-added STAG to be encapsulated in a provider backbone bridged frame with a Backbone Service Instance ID (called ISID), provider network-specific VLAN-ID (BVID) and backbone MAC (BMAC).

The generation of BVIDs depends on an intermediate instantiation service ID – ISID which is service-specific and network-wide unique. The ISID is a 24-bit VID (as

opposed to the CTAG, STAG or BVID which are all 12-bit values). The ISID maps a service to a network-specific VID. However, for forwarding purposes, the ISID needs to be mapped to a generic forwarding address (one that is backward compatible with 802.1Q)—for which a 12-bit BVID is introduced. The BVID along with the BMAC address is used by switches for forwarding in PBB-TE networks—leading to a network-wide unique 60-bit address. Multiple ISIDs need to be mapped to a single BVID (in a sense that, services which are graphically overlapping and subtending to the same destination may be assigned the same BVID) since the number of BVIDs is restricted to 4094 [30, 33]. To conserve BVIDs (restricted to 4094 or less), they are often reused. The standard does not consider the specific algorithm of how a BVID is assigned to a particular service and how to maximize the number of services in a network-graph, given the fixed number of BVIDs. Impromptu reuse of the BVIDs might result in a conflict of connection requests in a network. A much larger problem is when the same BVID is allocated by a network management system to two or more partially overlapping paths, as elaborated in section 2.2.3. This chapter discusses mechanisms to allocate BVIDs in a PBB-TE network [6-12] so as to satisfy a maximum number of connection requests. In a PBB-TE network, the communication between the source and the destination occurs by provisioning of managed paths [2, 23-24]. The mechanism of provisioning managed paths enables PBB-TE to facilitate deterministic delay and 50ms protection switching to a backup path. These paths are identified at a switching node—either an *edge* or a *core* backbone bridge—using the 60-bit BVID+BMAC identifier.

Apart from providing managed services, the manual set up process allows for the creation of multiple paths between a pair of source and destination nodes. This is necessary when we consider the issue of provisioning more than one service request between a source and destination pair [25, 26]. The paths provisioned for service requests may or may not follow the same route from the source to the destination, as explained in Section I. In this situation, each service is provisioned on a different path. It may happen that these multiple provisioned paths pass through a common intermediate bridge (switch). This is where the BVID assignment problem arises. Consider the scenario shown in Fig. 2.6 where two connection requests arrive between source V_1 and destination V_6. Both the connection requests pass through common

switches V_2 and V_5 and then diverge to follow different paths towards the destination node V_6. Another situation that will lead to the BVID assignment problem is when two services are requested from different source nodes to the same destination node, and the paths provisioned to these services pass through a common node. This is shown in Fig. 2.7 where two connection requests are between source V_1 and destination V_7 and source V_2 and destination $V_{7,}$ respectively. Both the connection requests pass through common switches V_3 and V_6 and then diverge to follow different paths towards the destination node V_7 [8, 105].

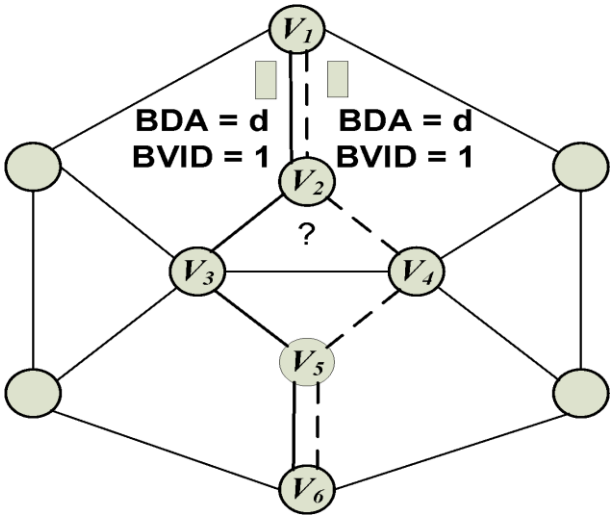


Fig. 2.6. Two service requests between a source-destination pair and passing through the common nodes.

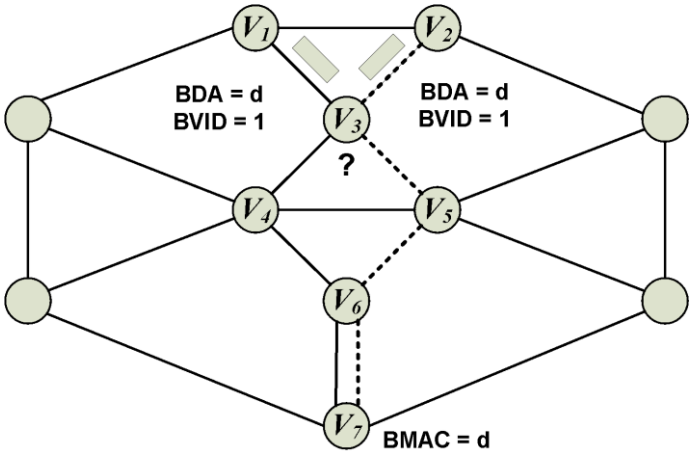


Fig. 2.7. Two services from different sources to the same destination and passing through common nodes.

In both the scenarios explained above, the problem is that the same BVID is assigned to two or more connections that have certain (well-defined) graphical properties causing discrepancy in the switch forwarding plane. However, given the limited number of BVIDs in a network (restricted to 4094 instances), there is a need to reuse the same BVID across multiple paths [6]; this means that this problem of BVID replication need to be addressed. Another common practice in traffic-engineered networks is to allocate a small number of dedicated BVIDs to each switch. This reduces the size of the forwarding tables that ultimately reduces the cost of the switch [18]. If we use the same BVID to provision multiple paths between the same source-destination pair, and further, if the paths cross or share the same set of links, then there will be a switching problem that would result in the failure of the PBB-TE system. This means, the switching fabric at a common node should have—for the same BMAC (corresponding to the destination) and the same BVID—a different output port mapping; it is not possible to have different mappings for the same 60-bit address. This is explained in Fig. 2.8.

Consider Fig. 2.8, where two services R_1 and R_2 are provisioned over paths P_1 and P_2. Paths P_1 and P_2 intersect at node V_3. Service R_1 enters node V_3 at port interface O_3 while service R_2 enters node V_3 at port interface O_6. Both R_1 and R_2 desire to exit node V_3 at interfaces O_4 and O_7, respectively. It is not possible to have different mappings for the same 60-bit address. Both the paths use the same 60-bit address to identify different ports, which causes a discrepancy in the forwarding table of the switch. Hence, the switch will not be able to differentiate traffic on the two outgoing tributaries. This can lead to ringing between the ports (random traffic passed to random ports) and lead to failure in provisioning the services R_1 and R_2, ultimately leading to the failure of the PBB-TE network.

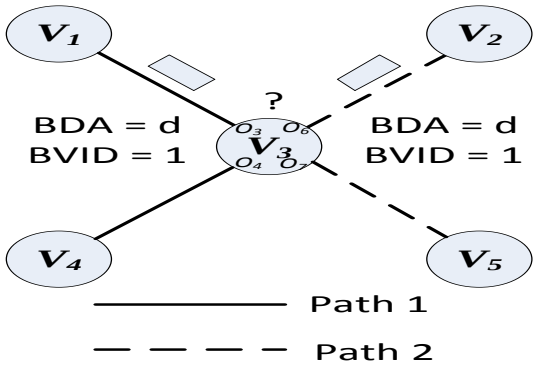


Fig. 2.8. Node V_3 is not able to forward the packets to the appropriate interface.

To avoid this issue of assigning the same BVID to multiple paths that are heading to a common destination, we propose four BVID allocation algorithms. These algorithms are optimized to use the least number of BVIDs to provision the maximum amount of traffic in the network [8]. To understand the severity of this problem, a general case is also considered, whereby, many source-nodes desiring to communicate to a common destination node experience the BVID assignment problem. In this general case, if the same BVID is used across each of the requests and if these requests have any common intermediate node, then this node would not be able to process and differentiate between individual requests [8]. To enable multiple overlapping paths to the same destination node, it is imperative that a unique BVID be allocated to every path (as shown in Fig. 2.9). As every network can support a finite number of unique BVIDs (4094 or less), we need a mechanism to allocate and reuse these BVIDs such that maximum connections can be provisioned in the network [8].

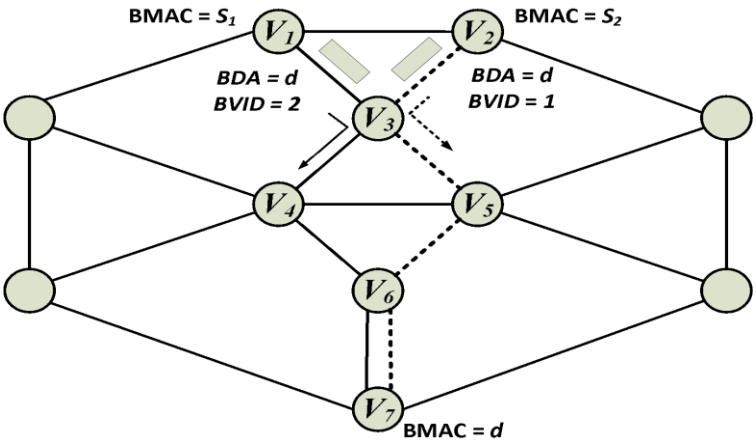


Fig. 2.9. The path followed by a frame with distinct BVIDs.

2.2.3.1 Formal definition of the BVID allocation problem*:* For a more intuitive understanding of the BVID allocation problem, we define the term "*Conflicting Connections*" for two connection requests c_i and c_j in a network, which satisfy the following properties:

1. c_i and c_j have the same destination node;
2. c_i and c_j have one or more common nodes(excluding the destination) on the paths allocated to them, and;
3. The part of the paths allocated to c_i and c_j, after the 1st common node does not contain the same set of nodes.

The idea is that while allocating BVIDs to the connection requests, no two *Conflicting Connections* should be assigned the same BVID. The BVID allocation problem is then stated formally as: Given a set of connection requests, C, choose a path for each request and allocate to each request a BVID:

1. The paths are the best-fit paths available (bandwidth capacity) from the source to the destination, for each c_i, satisfying the service granularity, and;
2. Maximum number of connection requests can be satisfied in the network [8].

Let $G = \{V, E, W\}$ be the graph representing the network topology, where, V = set of vertices indicating the switches in the network. E = set of edges indicating the active links between switches (vertices). W = capacity of each link.

Let $c_i \in C$ be i^{th} connection request between vertices s and d with granularity w: $s, d \in V$ and $w \leq \max(W)$.

Let B = set of BVIDs that can be allocated in the network.

Let L_d^s be a potential path for c_i.

Let B_d = set of all the BVIDs associated with vertex d.

$\forall\, b_i \in B_d$, let L_{Bd} = set of paths such that b_i is allocated on that path.

To provision a new path, a BVID b should be allocated such that it does not result in conflicting connections. The necessary conditions to ensure the above constraint are:

$$\forall\ L_{B_d}^{r'} \in L_{B_d}, \text{ let } b' \text{ be a BVID associated with } L_{B_d}^{r'}.$$

$$\text{If } L_d^s \cap L_{B_d}^{r'} = V' = \{v_1, v_2 \ldots v_n\} : \sum_k v_k \geq 2$$

$$\forall\ \mathrm{v_i} \in V'.$$

Let $L_{v_i}^d$ be a sub-path of L_d^s between v_i and d and $L_{v_i}^{d'}$ be a sub-path of $L_{Bd}^{r'}$ between v_i and d'. If both the sub-paths traverse the same set of nodes, then the same BVID can be allocated to a connection request c_i. If the two sub-paths are different, then a different BVID has to be allocated.

$$\text{If } L_{vi}^d = L_{vi}^{d\prime} \text{ then } b = b',$$

$$\text{else } b \neq b'.$$

2.3 Optimal BVID Allocation and Complexity

2.3.1 Integer Linear Program (ILP) Formulation

In this section, we formulate the static BVID allocation problem as a constrained optimization problem. We assume that 'N' connections are requested over a network and total 'B' BVIDs are available. Our goal is to maximize (up to N) connections using the pool of B BVIDs.

Indices:

1. $p, q \in P$, indices for paths.
2. $i, j, k \in N$, indices for connection requests.
3. $b \in B$, indices for BVIDs used.

The input parameters to the ILP formulation are as shown below:

1. P: Number of relevant paths in the network through which these connection requests can be satisfied.
2. B, N as defined above.
3. $D \in Z^{PxP}$ is a PxP matrix such that:

$$D_{pq} = \begin{cases} 1, \text{if path } p \text{ and } q \text{ do not conflict,} \\ 0, \text{otherwise.} \end{cases}$$

4. $E \in Z^{NxP}$ is a NxP matrix such that:

$$E_{ip} = \begin{cases} 1, \text{if network topology permits conn req } i \\ \quad \text{along path } p \\ 0, \text{otherwise.} \end{cases}$$

The decision variables are:

5. $X_{ip} = \begin{cases} 1, \text{if path } p \text{ is alloted to conn req } i, \\ 0, \text{otherwise.} \end{cases}$
6. $Y_i^b = \begin{cases} 1, \text{if conn req } i \text{ uses BVID } b, \\ 0, \text{ otherwise.} \end{cases}$
7. $Z_{ij} = \begin{cases} 1, \text{if conn req } i \text{ and } j \text{ use same BVID,} \\ 0, \text{ otherwise.} \end{cases}$

The model that includes the above decision variables will suffice to yield a feasible (and implementable) solution to the BVID allocation problem.

Objective: Our objective is to maximize the number of connection requests satisfied. Hence:

$$\textbf{Maximize } \sum_k \sum_j X_{kj}$$

Subject to the following set of constraints:

Each connection request should be allocated a feasible path.

$$X_{kj} \leq E_{kj} \quad \forall k, j \qquad (1)$$

Each connection request must get one path at the most (no splitting of requests among paths is permitted).

$$\sum_j X_{kj} \leq 1 \quad \forall k; \text{ and } \sum_k X_{kj} \leq 1 \quad \forall j \qquad (2)$$

Every provisioned connection request must have an associated BVID.

$$\sum_b Y_k^b = \sum_j X_{kj} \quad \forall k. \qquad (3)$$

Transitivity between *i, j* and *k*:

$$z_{ij} \geq z_{ik} + z_{kj} - 1 \qquad (4)$$

$$y_i^b \geq y_j^b + z_{ij} - 1 \qquad (5)$$

$$z_{ij} \geq y_i^b + y_j^b - 1 \ \forall i, j, k, b \qquad (6)$$

If connection request *i* and *j* are assigned the same BVID and are assigned path *p* and *q*, respectively, then path *p* and *q* should be non-conflicting paths.

$$x_{ip} + x_{jq} + z_{ij} \leq 2 + D_{pq} \ \forall i, j, p, q \qquad (7)$$

Reflexivity on Z: if connection request *i* and *j* are assigned the same BVID, then the reverse should also hold true. Thus,

$$z_{ij} = z_{ji} \ \forall i, j \qquad (8)$$

The decision variables *X*, *Y* and *Z* can have only binary values.

$$x_{ip}, \ y_i^b, z_{ij} \in \{0,1\} \quad \forall i, j, p, b \qquad (9)$$

The ILP attempts to maximize the connection requests provisioned in a given network with a given set of BVIDs satisfying all the aforementioned constraints. The ILP is computationally intensive and shown to be NP-complete (as shown in the next

section). The NP-completeness implies that there is a need for fast heuristic algorithms that compute the BVID matching in tractable time. In the subsequent sections, we propose four different heuristics for efficient BVID allocation in tractable time.

2.3.2 Complexity of the BVID Allocation Problem

To prove NP-completeness of the BVID allocation problem, we demonstrate that solving the *n*-graph-coloring problem will solve the Static BVID Allocation problem [13, 14]. Initially we convert an instance of a graph coloring problem to an instance of BVID allocation problem. Graph 1 shows a 4-node graph for the graph coloring problem:

$G'' = \{V'', E''\}$ be an undirected graph.

V'' = Set of vertices.

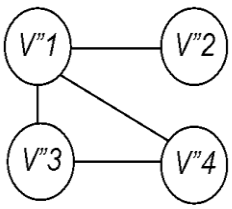
E'' = Set of edges between the nodes in V''.

An algorithm is now presented to convert the instance of a graph coloring problem into a network with a set of connections; the output is shown in Graph 2. The number of connections in graph 2 (called *G*) is equal to $|V''|$.

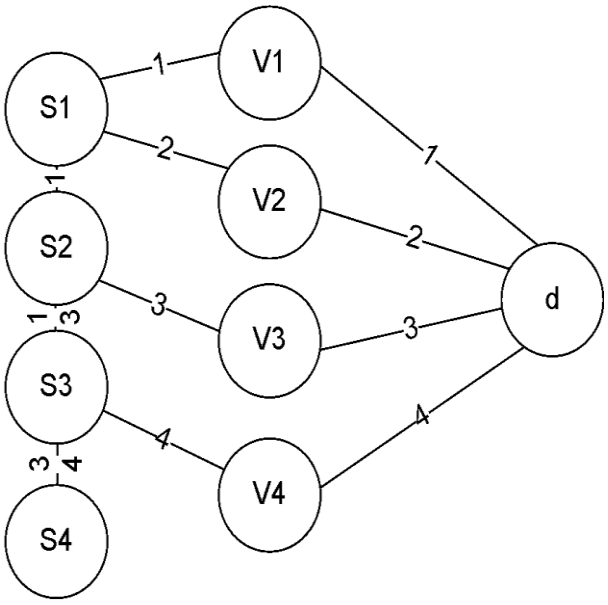
1. Create a separate node $d \in V$ in *G* for each disjoint graph in the given graph G''.
2. Create a node $v_i \in V$ in *G* for each node $v''_j \in V''$.
3. For every edge $e''_k \in E''$ between nodes v''_i and $v''_j \in V''$, create a node $s_k \in V$ in an auxiliary graph *G* such that *k* is set to 1 and is incremented for every edge in the original graph G''.
4. Each node in V'' represents a connection request, and each edge represents a conflict between the two requests. Accordingly, we build paths for connection requests by adding edges $e_r \in E$ in the new graph *G*.
5. For example, in Graph 1, nodes v''_1 and v''_2 are connected by an edge. It corresponds to 2 conflicting connection requests in *G* (Graph 2). To build these two

connection requests, we connect d to corresponding $v_i \in V$. E. g. in Graph 2, we connect d to v_1 and v_2.

6. Next, we extend these connection requests to a corresponding destination node $s_k \in V$. For example, in Graph 2, we extend connection 1 from d-v_1 to d-v_1-s_1. Similarly we extend connection 2 from d-v_2 to d-v_2-s_1.
7. Now consider nodes v''_1 and v''_3 in Graph 1. We create connection 3, as explained in steps 5 and 6. However, we have already created a connection 1. Hence, we extend the connection 1 to the next node $s_{k+1} \in V$ such that connection 1 and 3 conflict (since there is an edge between in v''_1 and v''_3 Graph 1). Now, connection 1 has a path d-v_1-s_1- s_2 in Graph 2.
8. We repeat the steps 5-7 for all the nodes in G''. The numbers on the edges in G indicate which connection/s that particular edge is a part of. The resultant graph of the algorithm above is shown in Graph 2 below.



Graph 1. Graph for coloring problem.

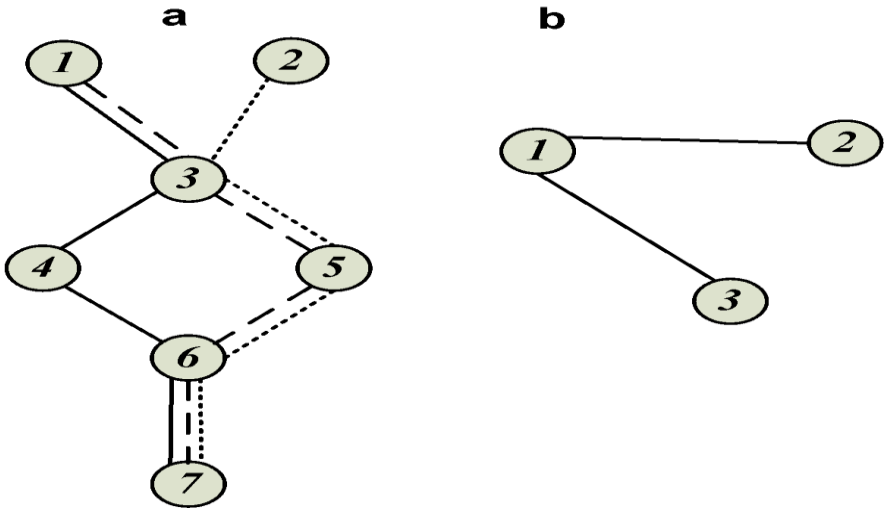


Graph 2. Equivalent network and connections for the graph 1.

From the construction of the network graph explained in the algorithm above, it is evident that if no two adjacent nodes have the same color in Graph 1, the connections in the Graph 2 can be assigned the BVIDs satisfying the condition mentioned above (no two *Conflicting Connections* are assigned the same BVID) [28, 29].

We now convert a BVID problem to a graph-coloring problem. Consider a Graph 3-a shown below:

Consider that we have two BVIDs available (b_1 & b_2). Now consider a set C' of connection requests such that $C' = \{c_1, c_2, c_3\}$. Connection requests c_1, c_2 and c_3 are such that path followed by them are via nodes 1-3-4-6-7, nodes 2-3-5-6-7 and nodes 1-3-5-6-7 respectively. We observe that c_1 (*solid line*) conflicts with c_2 *(dashed line)* and c_3 *(dotted lines).*



Graph 3. Graph for the BVID assignment problem and the graph coloring problem.

Let c_1, c_2 and c_3 represent separate nodes in the undirected Graph 3-b such that:

$G' = \{V', E'\}$ and V' = Set of vertices with each vertex representing a connection request c_i in C' in Graph 3-a. E' = Set of edges between the nodes in V'. There exists an edge between the nodes i and $j \in V'$ if the corresponding connection requests c_i and $c_j \in C'$ are *Conflicting Connections*. In this case, the graph will resemble Graph 3-b given above. The graph coloring problem in the Graph 3-b can be solved with two colors. For example, node 1 is assigned Red (R) and nodes 2 and 3 are assigned Blue (B); no two adjacent nodes have the same color. Let us map these colors to the BVIDs, say, $R = b_1$ and $B = b_2$. We observe that if connection request c_1 is assigned BVID b_1 and connection requests c_2 and c_3 are assigned BVID b_2, the BVID assignment problem is also solved.

Coloring of *V'* with *n* or less colors so that no two adjacent vertices have the same color will solve the problem of BVID allocation. Thus, finding a feasible coloring scheme will result in finding a feasible BVID allocation scheme. Essentially, we have converted the graph-coloring problem into a BVID allocation problem, thereby completing the proof for NP-completeness [16, 17].

2.4 Heuristic Approached for BVID allocation

In this section, we propose four heuristic algorithms to assign BVIDs in a network with dynamically arriving connection requests and a limited pool of BVIDs. The common approach in these algorithms is to reuse the BVIDs from a set of already allocated BVIDs. The algorithms we have proposed in this section differ in the method/steps by which the BVIDs are selected from a set of already used BVIDs. Note that these algorithms have been chosen from among multiple others, particularly because the performance of these algorithms compares well with the optimal, while they continue to be tractable in terms of complexity. The choice of the heuristics was determined by our goal of being able to meet the dynamic traffic provisioning scenario in tractable time [27]. In addition, we assume that each connection request is associated with some capacity (*service granularity*). Each link that is in the path allocated to a connection request should satisfy the service granularity of that connection request.

2.4.1 Sequential BVID Allocation

To assign a BVID, we create an auxiliary graph *G'* from the physical network topology *G*, by removing all the edges/links that do not satisfy the service granularity requested. The potential path for the service request is the best-fit path between source and destination in *G'*. The set of BVIDs for that potential path are checked for conflicting connections. We begin by proposing an algorithm in which the BVIDs are selected sequentially from a set of used BVIDs. The algorithm is stated below.

Let $R(s_r, d_r, w_r)$ be a service request,

Where, s_r = source node, d_r = destination node and w_r = service granularity.

Let $G\{V,E,W\}$ be the actual physical network topology. Let $\{e\} : w_e < w_r$,

G' = G – {e},

L_{sd} = shortest path between (s, d),

Let B = {All BVIDs allocated on L_{sd}},

for each b_i in B

begin

Let G_{b_i} be graph of all the links associated with BVID b_i

$$G'_{b_i} = G_{b_i} \cup L_{sd}.$$

If a cycle exists in G'_{b_i}, then discard b_i and select the next BVID.

else return b_i.

end

If no valid BVID $\in$ B satisfies the required constraints

begin

Calculate next shortest path (that excludes node(s) in L_{sd} with >32 BVIDs) which is edge disjoint with the previous path between s_r and d_r and perform the steps mentioned above.

If no path is available, then select the select the most optimal path and assign any unallocated BVID.

End

2.4.2 Random BVID Selection

In this algorithm, we select the BVIDs from set *B* randomly, instead of sequentially. The details of this algorithm are given below. We also note that the Random BVID selection technique is a variant of the Sequential algorithm.

Let $R(s_r, d_r, w_r)$ be a service request,

Where, s_r = source node, d_r = destination node and w_r = service granularity.

Let G{V,E,W} be the actual physical network topology. Let {e} : $w_e < w_r$

G' = G – {e}

L_{sd} = shortest path between (s, d),

Let B= {All BVIDs allocated on L_{sd}}

while all the BVIDs from B are not processed, repeat

begin

Select BVID b_i randomly from B

begin

Let G_{b_i} be graph of all the links associated with BVID b_i

$$G'_{b_i} = G_{b_i} \cup L_{sd}$$

If a cycle exists in G'_{b_i} then discard b_i and select the next BVID.

Else return b_i.

end

end

If no valid BVID $\in$ B satisfies the required constraints,

begin

Calculate next shortest path (that excludes node(s) in L_{sd} with >32 BVIDs) which is edge disjoint with the previous path between s_r and d_r and perform the steps mentioned above.

If no path is available, then select the select the most optimal path and assign any unallocated BVID.

End

2.4.3 Weighted BVID Selection

In this heuristic, we will select the BVIDs depending on the weights associated with the BVIDs. The weight associated with a BVID is the number of distinct links associated with that particular BVID. Our approach is to select the least weight BVID from a set *B*. This approach reduces the possibility of loops to be formed. The heuristic algorithm is explained below.

Let $R(s_r, d_r, w_r)$ be a service request

Where, s_r = source node, d_r = destination node and w_r = service granularity.

Let G{V,E,W} be the actual physical network topology. Let {e} : $w_e < w_r$

G' = G – {e}

L_{sd} = shortest path between (s, d)

Let B= {All BVIDs allocated on L_{sd} with weight x_i associated with each BVID b_i }

sort B in the ascending order of the x_i

For each b_i in B

begin

Let G_{b_i} be graph of all the links associated with BVID b_i

$$G'_{b_i} = G_{b_i} \cup L_{sd}.$$

If a cycle exists in G'_{b_i} then discard b_i and select the next BVID.

else return b_i.

end

If no valid BVID $\in$ B satisfies the required constraints

begin

Calculate next shortest path (that excludes node(s) in L_{sd} with >32 BVIDs) which is edge disjoint with the previous path between s_r and d_r and perform the steps mentioned above.

If no path is available, then select the select the most optimal path and assign any unallocated BVID.

end

The algorithms mentioned above differ in the manner by which BVIDs are reassigned to the non-conflicting connection requests. In the heuristics proposed we prepare a set of already used BVIDs and attempt to reuse these to minimize the total number of BVIDs. The difference in the proposed heuristics is how the BVID is selected from the set of already used BVIDs. In the *Sequential BVID selection* heuristic, we start from the beginning of the set until we get a valid BVID to reuse. In *Random BVID selection* heuristic, a BVID is chosen randomly to be reused, while in the *Weighted BVID selection* heuristic, in addition to the set of already used BVIDs, we maintain weights associated with each BVID. These weights indicate the number of links, the BVID that has been used. We select the BVID from the set which has the least weight associated with it.

2.4.4 Weighted Links and Weighted BVID Allocation

This heuristic is a variant of a *Weighted BVID* allocation algorithm and differs from the already proposed algorithm in the steps by which the weights are assigned to the links in the network as well as to the BVIDs. The weight associated with a link indicates the total number of paths in which that particular link has been utilized. It is obvious that the more the weight of a link, the higher the probability that the path comprising that link will conflict with the new incoming path. Initially all links have *ZERO* weight assigned. In addition, we impose a penalty (constant positive integer *P*) on a link if the link is a part of a sub-path which causes conflict with an already allocated path. The penalty is *P.N*, where *N* is the number of paths with which the sub-path comprising that link is causing conflicts. In this variant of the Weighted BVID allocation algorithm, we have relaxed the constraint of choosing the shortest path. Instead, for an incoming

connection request, we attempt to find a path which comprises links that have as less a weight as possible.

Let $R(s_r, d_r, w_r)$ be a service request

Where, s_r = source node, d_r = destination node and w_r = service granularity.

Let $G\{V,E,W\}$ be the actual physical network topology. Let $\{e\} : w_e < w_r$

$G' = G - \{e\}$

L_{sd} = WeightedLinkPath (s, d)

Let B= {All BVIDs allocated on L_{sd} with weight x_i associated with each BVID b_i }

sort B in the ascending order of the x_i

For each b_i in B

begin

Let G_{b_i} be graph of all the links associated with BVID b_i

$$G'_{b_i} = G_{b_i} \cup L_{sd}.$$

If a cycle exists in G'_{b_i} then discard b_i and select the next BVID.

else return { b_i}.

end

If no valid BVID $\in$ B satisfies the required constraints

begin

Calculate next WeightedLinkPath(s, d) (that excludes node(s) in L_{sd} with >32 BVIDs) which is edge disjoint with the previous path between s_r and d_r and perform the steps mentioned above.

If no path is available, then select the select the most optimal path and assign any unallocated BVID.

end

The sub-routine '*WeightedLinkPath'* calculates a path for an incoming connection request that comprises the links that have as minimum of weights as possible while satisfying the link-capacity constraint. If no BVID could be allocated from the already used set of BVIDs to the chosen path between the source and the destination, the routine then finds next sub-optimal path. This process continues till all the possible paths are tested. If no path exists for which a BVID re-use is possible, a new BVID is allocated to the next best path. The newly proposed heuristic "Weighted Links and Weighted BVID Allocation" outperforms all the three already proposed simple heuristics; however the execution time for this algorithm is much more than that of its variants.

2.4.5 Complexity Analysis

In case of the first three algorithms, the complexity of the best-fit path finding module is $O(N^2)$, where N is the number of nodes in the network (Dijkstra's algorithm). Assume that the average number of BVIDs associated with each path that we calculate is B. It means that, the first loop in the heuristics will get executed for B number of times. In addition, assume that the average number of links associated with each BVID is $L`$. Hence, the complexity of the module to find the loop in the network is $O(N+L`)$. We find the next best-fit path if no BVID can be reused on the current path. This process is repeated B times. We find k-next best-fit paths before we assign an unused BVID to the new request. Hence the complexity of the complete algorithm is: $O(N^2 + k*B*N + k*B*L`+ k*N^2)$. This term is reduced to $O(N^2)$ by discarding the constants.

The complexity of "*Weighted Links and Weighted BVID*" algorithm is bounded by the complexity of a sub-routine—'*WeightedLinkPath*'. In this case, the complexity of the sub-routine is $O(N^2*L)$, since we are comparing each path with each link for a possible conflict. Keeping the rest of the assumptions same, the overall complexity of the heuristic is: $O(N^2*L + k*B*N + k*B*L + k*N^2*L)$. The whole term can be approximated to $O(N^2*L)$ by dropping the constants and keeping higher degree terms. It shows that the complexity of the heuristic is dependent on N and L, where N is the number of nodes and L is the number of links. We observe in Table 2.2 that the

execution time for the "*Weighted Links and Weighted BVI"* algorithm increases rapidly compared to the previous three algorithms as the size of the network grows. The diagram below compares the four proposed heuristic algorithms relatively on the basis of the following parameters:

1. Complexity of the algorithm,
2. Performance of the algorithm: The total BVIDs required for a particular input instance of connection requests (100 node-dense-mesh with 2000 connection requests),
3. Optimality of the solution,
4. Actual time required by the algorithm to execute a particular input instance (the same as in 2).

As far as parameters 1, 2 and 4 are concerned, a smaller value indicates better performance of the heuristic algorithm, since it is desired that the heuristic is less complex and satisfies the connection requests with a minimum number of BVIDs. However, for parameter 3, a larger value indicates better performance of the heuristic algorithm, since it is desirable that the heuristic produces an optimal solution. For better understanding of the comparison we have normalized all the values to the scale of 100 as shown in Fig. 6 below.

We observe from the Fig. 2.10 that the heuristic "*Weighted Links and Weighted BVIDs*" produces best results as it requires the least number of BVIDs to satisfy a set of connection requests. However, it has the highest complexity (of the order O($N^2.L$)) and takes the longest duration to execute. On the contrary "*Sequential BVID allocation*" and "*Random BVID allocation*" heuristics need the highest number of BVIDs, however, complexity is the lowest. The "*Weighted BVID allocation algorithm*" performs close to the solution given by the "*Weighted Links and Weighted BVIDs*" heuristic. It requires significantly less execution time as compared to the "*Weighted Links and Weighted BVIDs*" heuristic. A detailed comparison between the performances of heuristics is shown in Table-2.2.

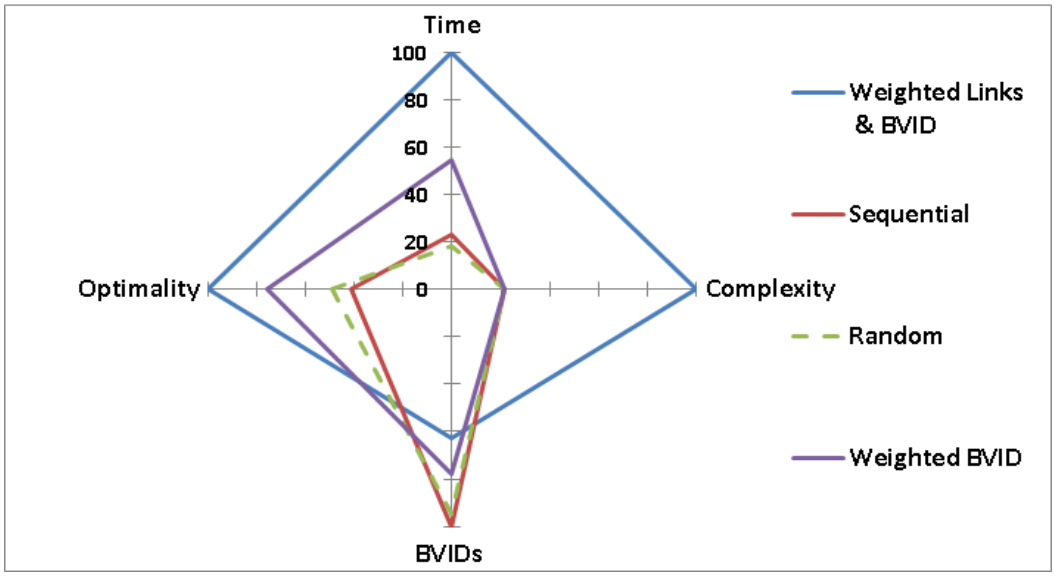


Fig. 2.10. Performance comparison of the proposed heuristic algorithms.

2.5 Evaluation of Heuristics and Simulation Results

In this section we provide a numerical evaluation of the proposed BVID allocation heuristics. Initially we have optimally solved the problem for up to a 40-node topology. A comparison between the optimal solution and the heuristics is given in Table 2.2. In addition, a comparison between the heuristics for larger data-sets is given in Table 2.3. We have measured the performance of the heuristics and ILP in terms of the numbers of BVIDs required to satisfy a given set of connection requests and the maximum number of connection requests satisfied for a specific number of BVIDs.

2.5.1 Illustrative Example

Consider a simple scenario that focuses on the need for improvements in the naïve BVID allocation heuristic. Consider the network topology of Fig. 2.7. Assume that we have to provision four connection requests, c_1, c_2, c_3 and c_4. Let p_1, p_2, p_3 and p_4 be the paths allocated to the given instances of connection requests respectively:

$p_1 = \{e_1, e_2, e_3, e_4\}$, $p_2 = \{e_5, e_2, e_3, e_4\}$, $p_3 = \{e_1, e_6, e_7, e_4\}$,

$p_4 = \{e_5, e_6, e_7, e_4\}$ *and*

$e_1 = \{v_1, v_3\}$, $e_2 = \{v_3, v_4\}$, $e_3 = \{v_4, v_6\}$, $e_4 = \{v_6, v_7\}$,

$e_5 = \{v_2, v_3\}$, $e_6 = \{v_3, v_5\}$, $e_7 = \{v_5, v_6\}$

To understand the problem, a simple naïve approach is considered which is described as follows:

1. For a given request, compute the available path.
2. Assign an available BVID to the request.
3. If we exceed the limit of 4094 BVIDs per network or the number of dedicated BVIDs per device is exhausted (in the event that for purpose of implementation each device has only a few allocated BVIDs), then reuse an already assigned BVID. Note that such unplanned reallocation will result in a conflicting problem in switches at overlapping locations.

Let us further assume that the number of the available BVIDs is bounded by a set $B = \{b_1, b_2\}$ where b_1 and b_2 are the available BVIDs in the network. The BVID assignment can be accomplished in several ways. Let us use a naïve (original or non-optimal) algorithm that picks up an available BVID and assigns it to the path associated with the request. Now we examine the performance of this naïve algorithm. The following sequence of actions is performed as the connection requests arrive:

- Connection request c_1 arrives.

- c_1 is provisioned with BVID b_1.

- Connection request c_2 arrives.

- c_2 is provisioned with BVID b_2 as b_1 is already assigned and b_2 is the next unassigned BVID (see Fig. 2.11).

- Connection request c_3 arrives.

Since both of the available BVIDs are already used, the algorithm will attempt to reuse the BVIDs. Reusing the BVID b_1 or b_2 will result in a ringing problem at Switch V_3 which will lead to a loop at Switch V_3 since c_3 conflicts with already provisioned

connection requests (as shown in Fig. 2.7). Similarly, the reuse of BVIDs for connection request c_4 will also result in a ringing problem at Switch V_3 (see Fig. 2.12). As a result, c_3 and c_4 cannot be provisioned and will be considered as dropped. Hence, with a naïve algorithm, eventually, we can fulfill only the first two connection requests resulting in no support for any additional requests.

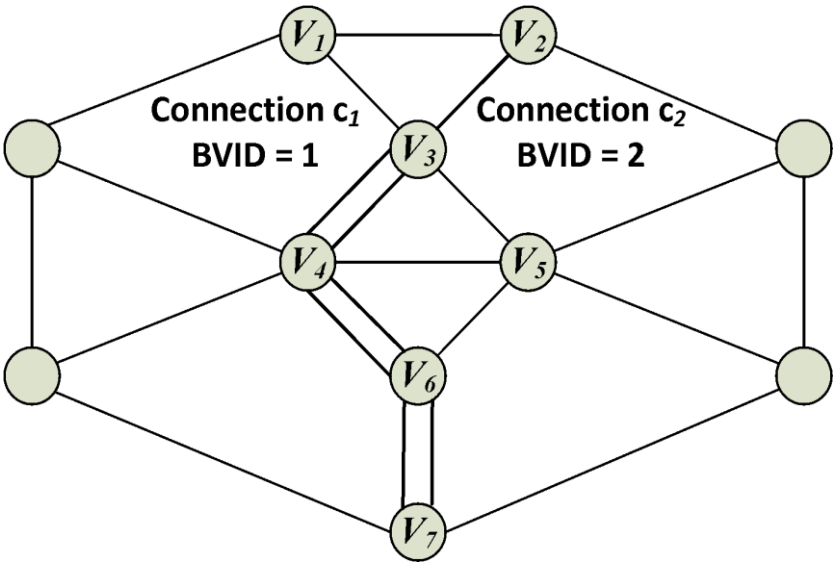


Fig. 2.11. Requests c_1 and c_2 are provided with BVIDs b_1 and b_2.

Now let us examine the proposed BVID allocation algorithms described in Section VII.

The following sequence of actions occurs as a connection request arrives:

- Connection request c_1 arrives.
- c_1 is provisioned with BVID b_1.
- Connection request c_2 arrives.
- c_2 is provisioned with BVID b_1. The proposed BVID allocation algorithms assign BVID b_1 to the connection request c_2 as the requests c_1 and c_2 do not conflict.

Compared with the naïve algorithm described earlier, after provisioning the request for c_2, we still have the BVID b_2 free for use in future connection requests (See Fig. 2.13). So, with the new proposed BVID allocation algorithms, we can fulfill all the four

connection requests, improving the service provisioning capability of PBB-TE Networks [8, 105].

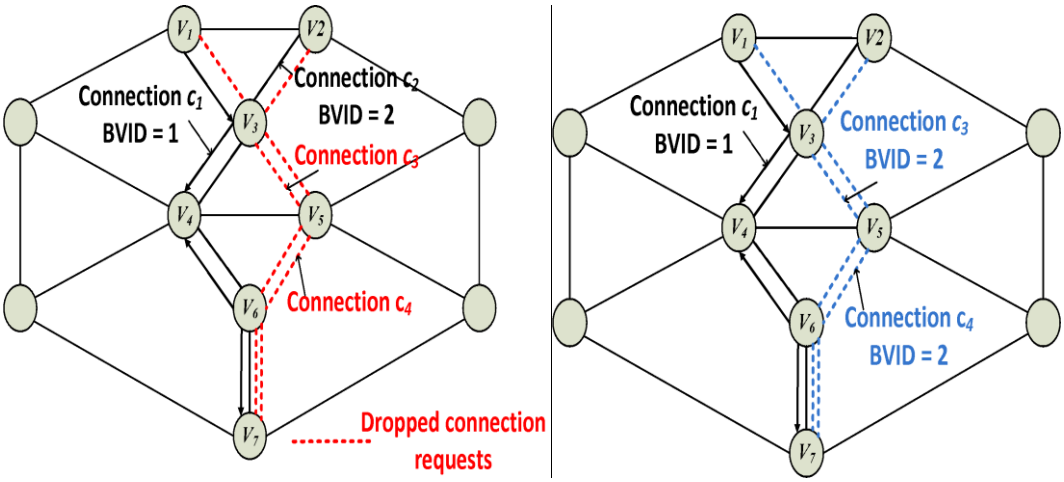


Fig. 2.12. Requests c_3 and c_4 cannot be provisioned due to the ringing problem.
Fig. 2.13. All requests have been provisioned with the proposed algorithm.

To gain more insight into the performance of the heuristics proposed above, we present a simulation model and evaluate our proposed heuristics for BVID allocation. In service provider networks, the two most common topologies used are the 'Ring Topology' and the 'Mesh Topology'. Rings are prevalent especially in legacy networks where SONET/SDH traffic existed and is now replaced with packet-mode Ethernet traffic. Mesh networks are a migration from rings, creating multi-degree hub nodes that direct traffic in a more efficient manner. It is commonly believed (and is well supported by documentation from the Metro Ethernet Forum [1, 8, 34-36]) that Carrier Ethernet implementation cases are the most common for rings and mesh networks. Linear topologies and trees are considered to be sub-cases of rings or mesh networks. So, we consider two of the most common network topologies—a network arranged in an interconnected ring topology (Fig. 2.14.a) and a network arranged as a mesh topology (Fig. 2.14.b) to demonstrate that the proposed heuristics are apt for real time networks, thereby, exhaustively covering network deployments [35, 36].

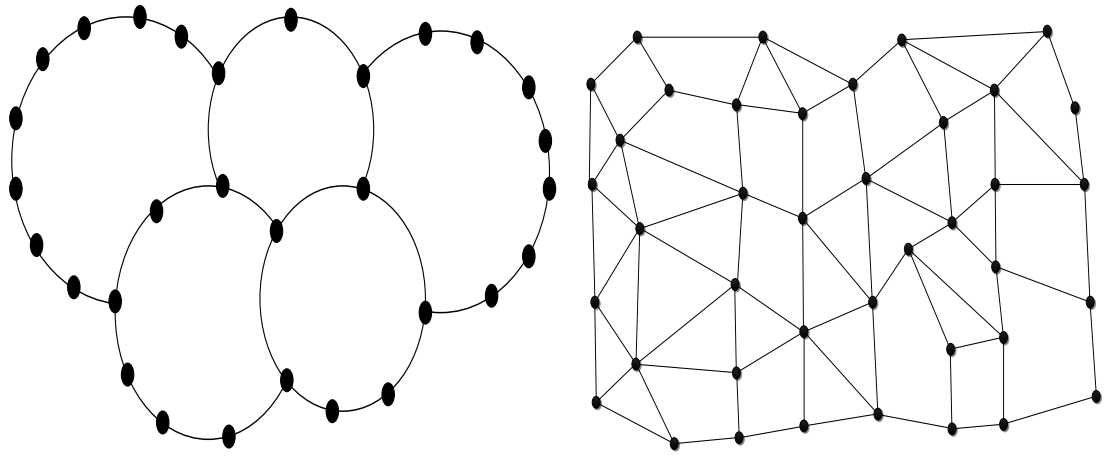


Fig. 2.14. (a) Ring topology. (b) Mesh topology.

We compare the proposed BVID allocation algorithms against a naïve approach for BVID allocation. Then we compare the performance of the heuristic against that of ILP.

2.5.2 Comparison between the Heuristics and the ILP

We now compare the performance of the *Weighted BVID* allocation heuristic (since it performs close to the optimal and executes in tractable time as shown in *VII.E*) with the optimal approach using the ILP proposed in Section V. The longest time taken by ILP is 23830.711 seconds (more than 6 hours approximately) for a dense mesh topology with 40 connection requests. However in some cases, we could not obtain the results due to the limitations on the server resources. The results are tabulated in Table 2.2. Results are provided for up to 50 requests, beyond which, we need more capacity simulation servers. An entry of '–' indicates that the ILP was unable to generate results at that traffic due to the involved intractability. From the results, we observe that the heuristic algorithms result in a performance which is close to the ILP. It can be observed from the 'Gap' column (heuristic-solution less ILP-solution expressed as a percentage) in the table, that the worst-case difference between ILP and the heuristic is 6.67%. In some cases, including the optimal case, the number of the connections satisfied is below the number of connection requests since there are no more BVIDs available to allocate to the new incoming request. Further, we also observe that the computational time required by the heuristic is well within the range for the practically deployed networks (the heuristics generate result in real-time), when compared to the large ILP execution times.

It must be noted that the ILP and heuristics perform similar only at a low number of requests. It can be intuitively understood from Table 2.2 that the gap continues to increase with load (number of requests), and hence, the ILP will perform significantly better at a larger number of requests.

No of Nodes	Topology Type	No_Connection Requests	CPU_TIME ILP (ms)	ILP Performance	CPU_TIME Heuristic (ms)	Weighted BVID Heurisitc Performance	Gap(%)	No of Nodes	Topology Type	No_Connection Requests	CPU_TIME ILP (ms)	ILP Performance	CPU_TIME Heuristic (ms)	Weighted BVID Heurisitc Performance	Gap(%)
10	Ring	20	267	20	48	19	5.00	30	Ring	20	270	20	63	20	0.00
		30	173108	29	59	27	6.67			30	582	30	95	30	0.00
		40	23807904	38	80	36	5.00			40	1764	40	115	38	5.00
		50	–	–	91	46	–			50	1211	50	156	49	2.00
	Mesh-Sparse	20	260	20	52	20	0.00		Mesh-Sparse	20	272	20	74	20	0.00
		30	172785	29	69	28	3.33			30	564	30	103	30	0.00
		40	–	–	84	36	–			40	1774	40	140	38	5.00
		50	–	–	108	47	–			50	1217	50	184	48	4.00
	Mesh-Dense	20	265	20	52	20	0.00		Mesh-Dense	20	254	20	69	20	0.00
		30	173525	29	68	27	6.67			30	1436	30	121	30	0.00
		40	23830711	38	79	36	5.00			40	14680	40	163	39	2.50
		50	–	–	119	46	–			50	312007	50	205	48	4.00
20	Ring	20	207	20	43	20	0.00	40	Ring	20	199	20	84	20	0.00
		30	681	30	70	29	3.33			30	475	30	111	30	0.00
		40	1390888	39	84	38	2.50			40	4025	40	133	39	2.50
		50	190476	50	96	48	4.00			50	33788	50	180	49	2.00
	Mesh-Sparse	20	211	20	64	20	0.00		Mesh-Sparse	20	198	20	94	20	0.00
		30	694	30	75	28	6.67			30	471	30	133	30	0.00
		40	1385229	39	89	38	2.50			40	4013	40	176	40	0.00
		50	193915	50	112	48	4.00			50	33525	50	199	48	4.00
	Mesh-Dense	20	882	20	71	20	0.00		Mesh-Dense	20	275	20	127	20	0.00
		30	1088	30	91	30	0.00			30	3467	30	164	30	0.00
		40	3497126	39	103	37	5.00			40	12579	40	203	40	0.00
		50	697088	50	190	48	4.00			50	11012	50	241	50	0.00

TABLE 2.2: Comparison between ILP and Weighted BVID Allocation heuristic

2.5.3 Comparison between the Heuristics and the Naïve BVID Allocation Algorithm

A simulations study of the heuristics is now presented. We have implemented the heuristics explained in Section VII and a naïve BVID allocation algorithm as managed applications developed with a C# code. To generate traffic, an application that generates non-Gaussian distribution traffic was developed. Readings for the number of BVIDs

allocated in two different networks—30-node interconnected-ring network and 40-node mesh network were obtained. The average degree of each vertex in the ring topology is 2.5 while in the mesh network average degree of each vertex is 3.5. Three sets of readings for service requests with different granularities 1Mbps, 2Mbps and 3Mbps were obtained. The connection request arrival rate is characterized by a truncated Poisson distribution and granularities were modeled as exponentially distributed. We have used CPLEX to solve the formulated ILP on an Intel quad-core 2.4GHz server with 32GB of RAM.

Fig. 2.15 shows the number of BVIDs allocated to the service requests in an interconnected ring network using the Sequential BVID Selection algorithm. Readings were obtained for three different sets of two hundred service requests with each set had a request granularity of 1Mbps, 2Mbps and 3Mbps respectively. One of the key findings is that the number of BVIDs required is very low compared to the number of service requests.

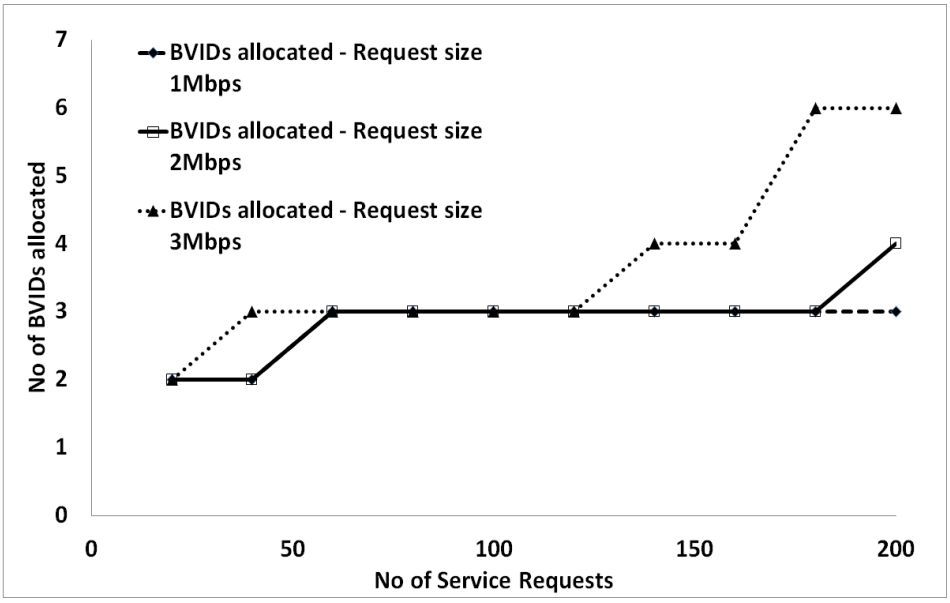


Fig. 2.15. BVIDs allocated for a Ring Network using the Sequential BVID Selection algorithm.

In Fig. 2.16, we show the number of BVIDs used in a network using the Random BVID Selection heuristic. We showcase the number of BVIDs used with the Weighted BVID Selection heuristic in Fig. 2.17 using the same traffic profile as used

for the generation of data in Fig. 2.16. It is again crucial to note that a limited number of BVIDs are required to provision the connection requests.

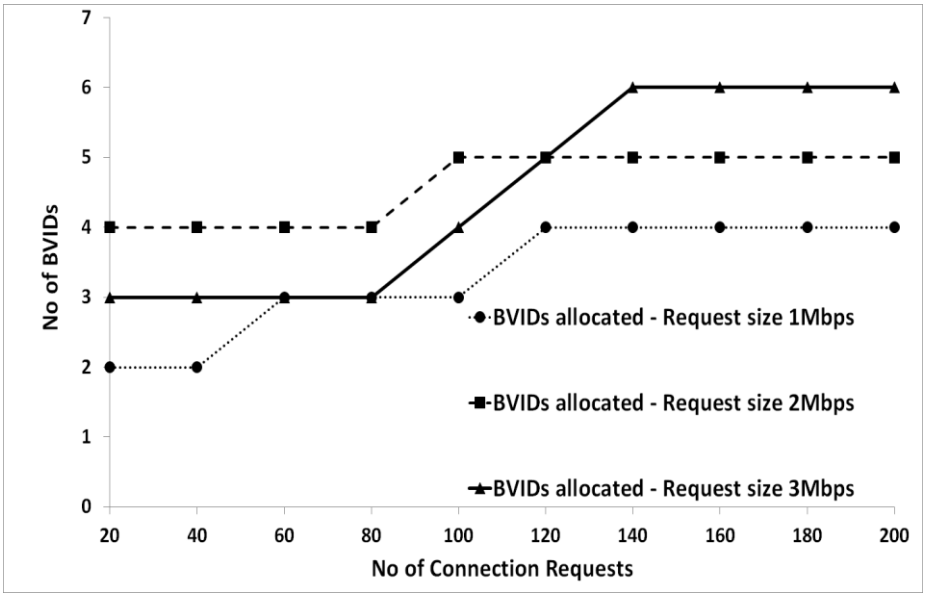


Fig. 2.16. BVIDs allocated for a Ring Network using the Random BVID Selection algorithm.

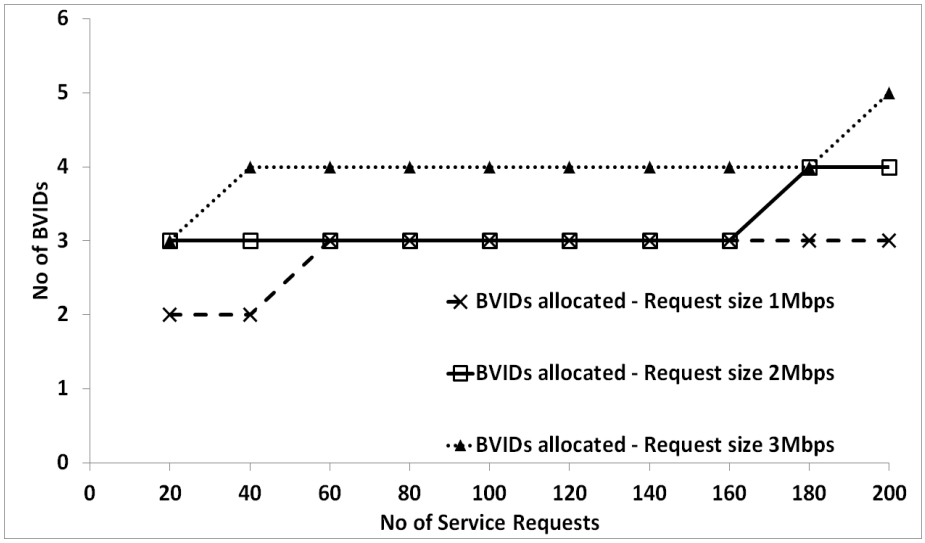


Fig. 2.17. BVIDs allocated for a Ring Network using the Weighted BVID Selection algorithm.

Fig. 2.18 shows the number of BVIDs allocated using the naïve BVID allocation algorithm applied to the traffic profile used to generate plots in Fig 2.15-2.19. It is evident from Fig. 2.15-2.19 that our proposed heuristics require a lesser number of BVIDs to configure the traffic as compared to the naive BVID allocation algorithm (approximately around 94% less BVIDs are required in the interconnected ring with the

proposed heuristics for BVID allocation). This is a substantial saving of BVIDs, thereby leading to better engineering of the CE network.

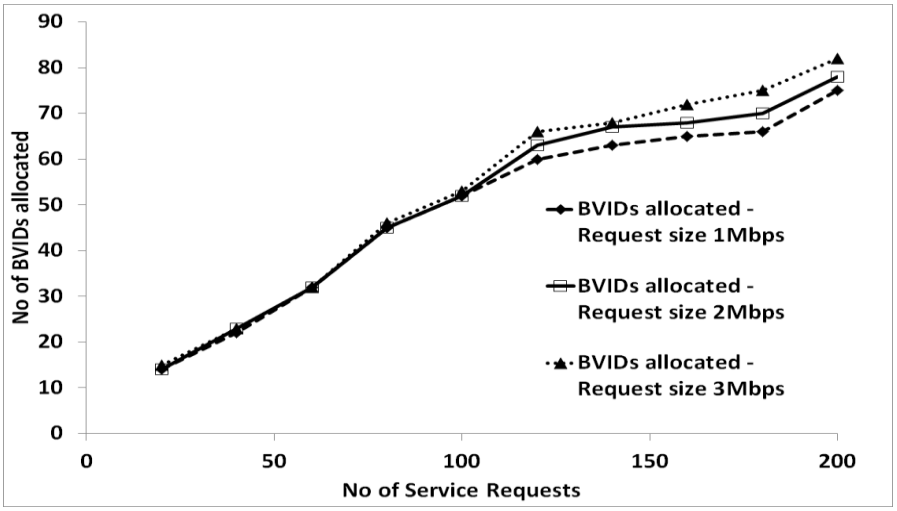


Fig. 2.18. BVIDs allocated for a Ring Network using naïve BVID allocation algorithm.

In the Fig. 2.19, we have displayed the number of BVIDs allocated to a set of two hundred service requests, with each set has a request granularity of 3Mbps in a network that has a ring topology, using the naive BVID allocation algorithm. We then compared it with our proposed heuristics. The graph verifies the statistics mentioned above. As can be seen in the figure, each of our proposed schemes outperforms the naïve approach. The takeaway of our schemes is that they are load invariant. The proposed heuristics consume the same number of BVIDs even for 200 traffic requests.

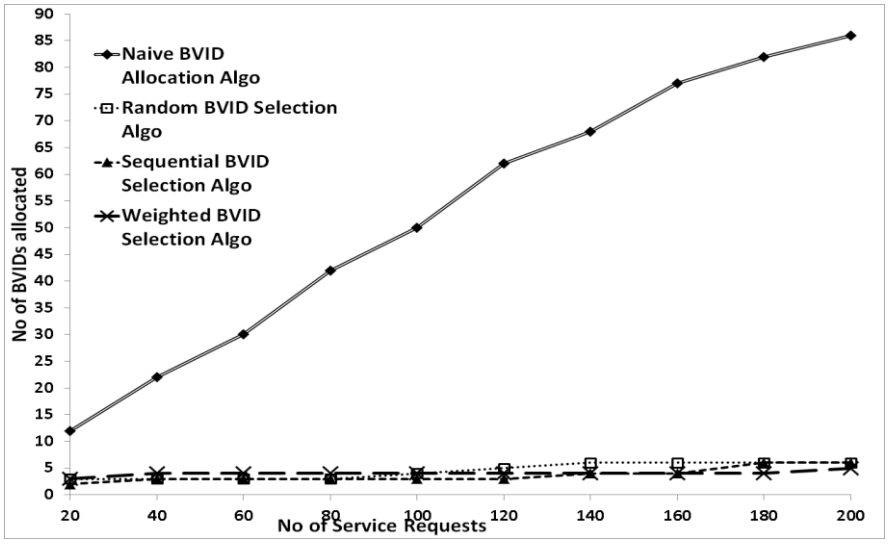


Fig. 2.19. A comparison of the BVIDs allocated in a Ring Network with the proposed BVID allocation algorithms and the naïve BVID algorithm.

Fig. 2.20 shows the number of BVIDs allocated for a mesh network using the Sequential BVID Selection heuristic that we have proposed using the same set of connection requests as in the interconnected rings.

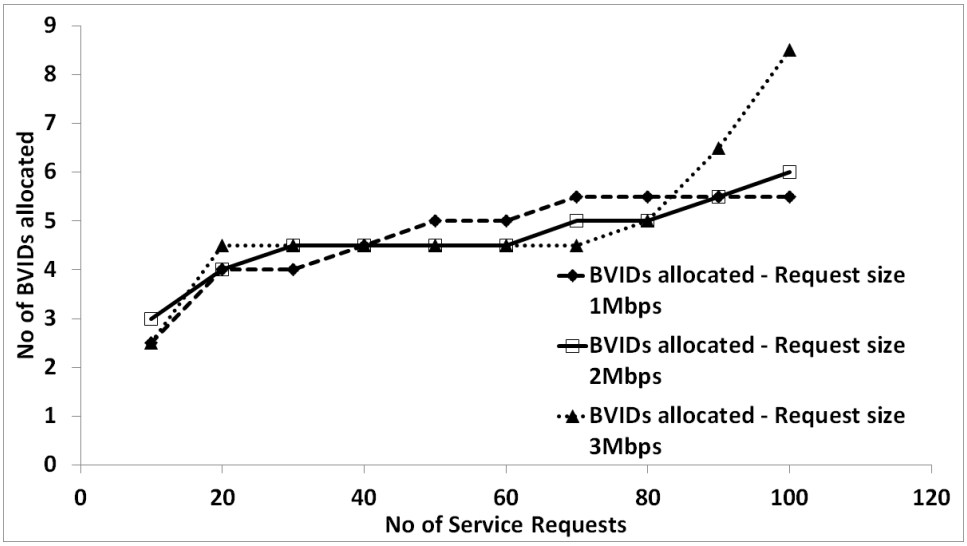


Fig. 2.20. BVIDs allocated for a Mesh topology with the proposed Sequential BVID Selection algorithm.

Fig. 2.21 and Fig. 2.22 displays the number of BVIDs allocated to 200 requests with a network that has a mesh topology using the proposed Random BVID Selection heuristic and the Weighted BVID Selection heuristic, respectively.

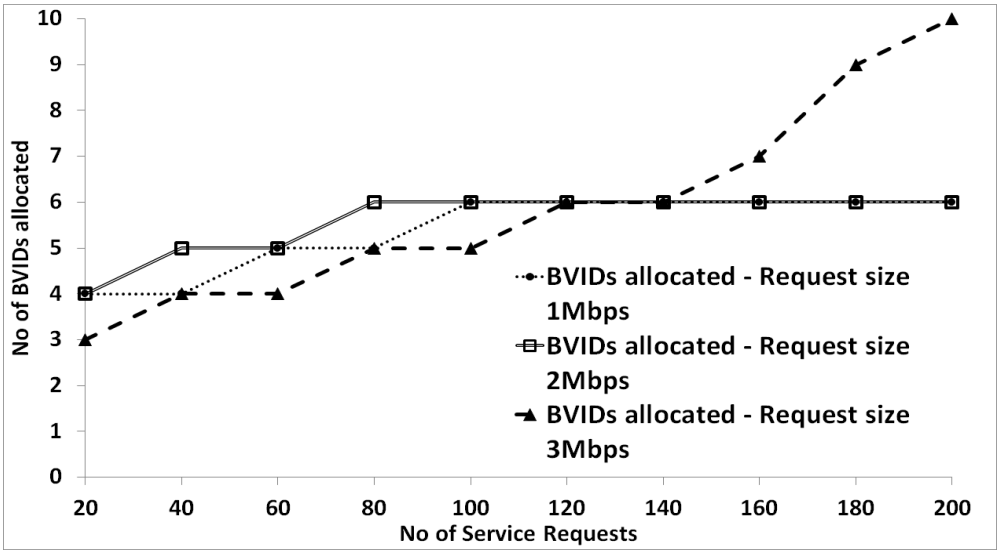


Fig. 2.21. BVIDs allocated for a Mesh Topology with the proposed Random BVID Selection algorithm.

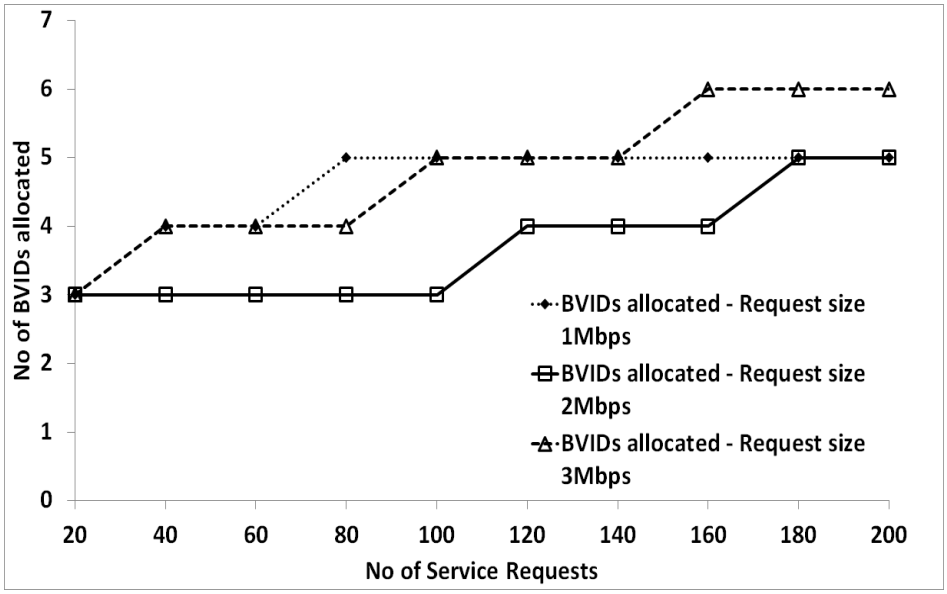


Fig. 2.22. BVIDs allocated for a Mesh Topology with the proposed Weighted BVID Selection algorithm.

Fig. 2.23 shows the number of BVIDs allocated to 200 requests by a naïve algorithm. In the mesh network, our proposed heuristic requires around 92% lesser BVIDs as compared to naïve BVID algorithm.

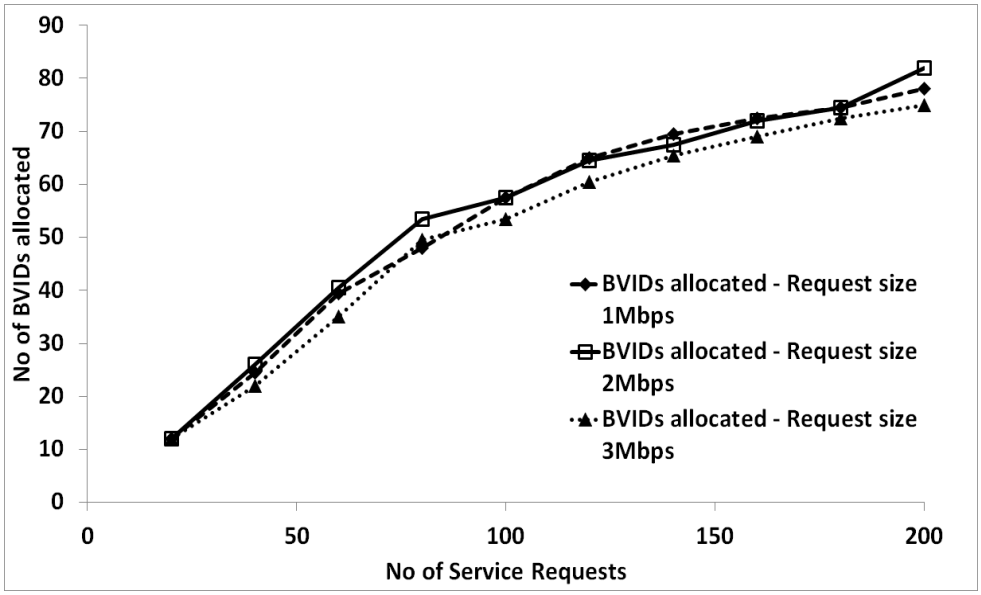


Fig. 2.23. BVIDs allocated for a Mesh Topology with a naïve BVID allocation algorithm.

We further compare the performance of the proposed heuristics against a naïve BVID allocation for the mesh network topology, shown in Fig. 2.22. In this figure the number of BVIDs allocated in a mesh topology to a set of two hundred service requests

with each request of granularity 3Mbps, is displayed. The graph verifies that the performance of our proposed heuristics continues to be better than the naïve approach.

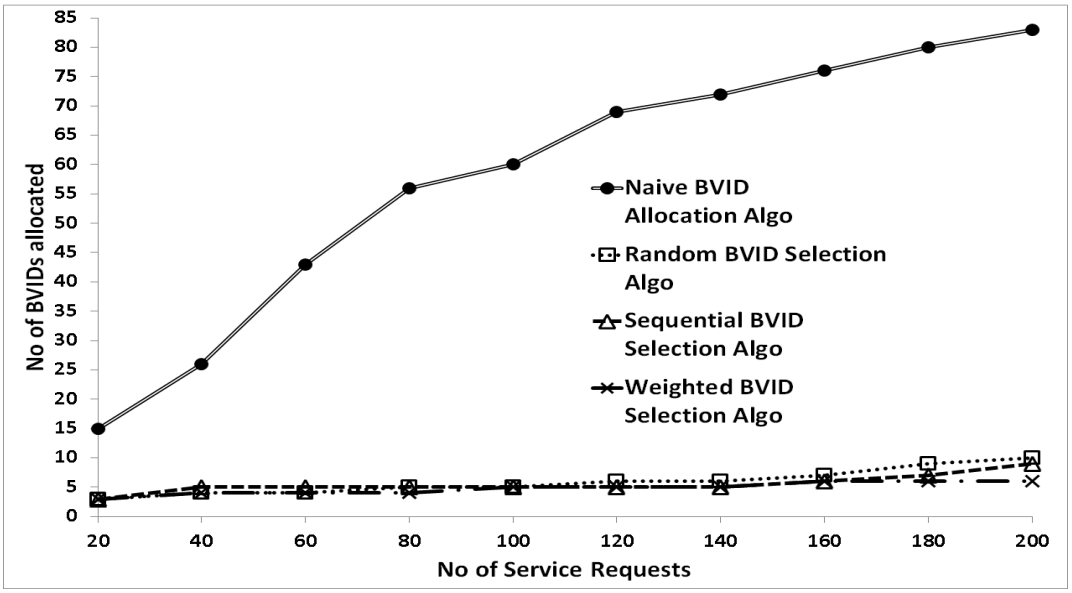


Fig. 2.24. A comparison - Mesh Network with the proposed BVID allocation algorithms and a naive BVID allocation algorithm.

While the collective performance of the proposed heuristics against the naïve algorithm is important, it is of interest to consider also, the performance of the individual heuristic algorithm. So, we compare the performance of individual heuristics against a naïve BVID allocation heuristic for both the ring and the mesh topologies. Fig. 2.25 shows a comparison of the number of BVIDs allocated to a set of connection requests in a ring and a mesh topology using a naïve BVID allocation algorithm and the Sequential BVID Selection heuristic algorithm. On an average, with a naïve BVID allocation algorithm, 78 BVIDs are required to configure all 200 services in the two network topologies—interconnected ring and mesh networks. On the contrary, when the Sequential BVID Selection approach is used, only 4 and 6 BVIDs are required.

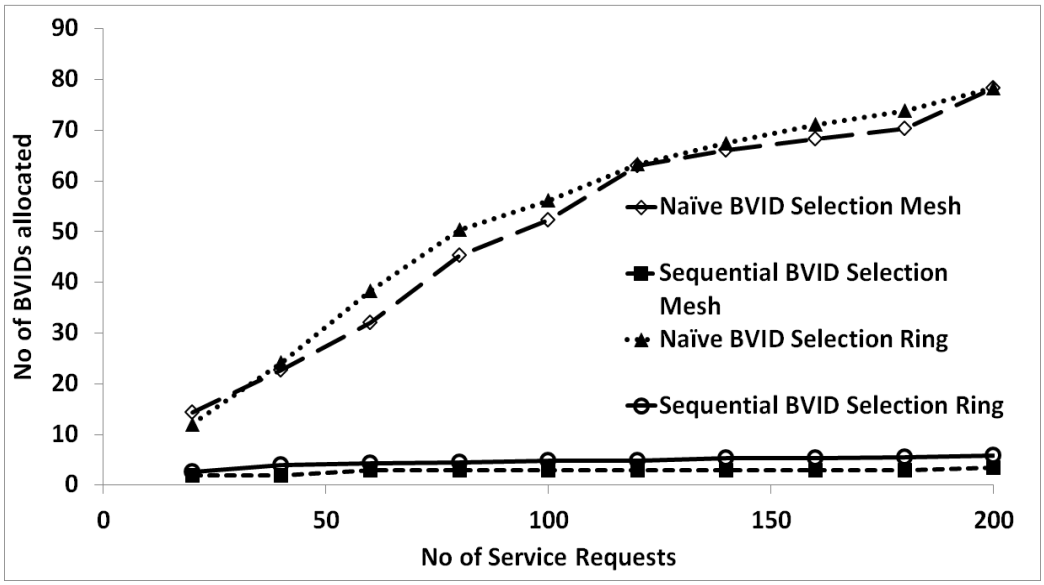


Fig. 2.25. BVIDs allocated for Ring and Mesh network topologies using a naïve and the proposed Sequential BVID Selection algorithm.

Fig. 2.26 and Fig. 2.27 show a comparison of the average number of BVIDs allocated to a set of connection requests in a ring topology and a mesh topology using a naïve BVID allocation algorithm and the Random BVID Selection heuristic and Weighted BVID selection heuristic algorithm, respectively. The Random BVID Selection heuristic requires 5 and 7 BVIDs for interconnected rings and mesh networks, while only 3 and 5 BVIDs are required for interconnected rings and mesh networks using the Weighted BVID Selection algorithm. This is a significant improvement in the naïve BVID allocation algorithm, by which, a saving of 92% to 94% BVIDs is observed.

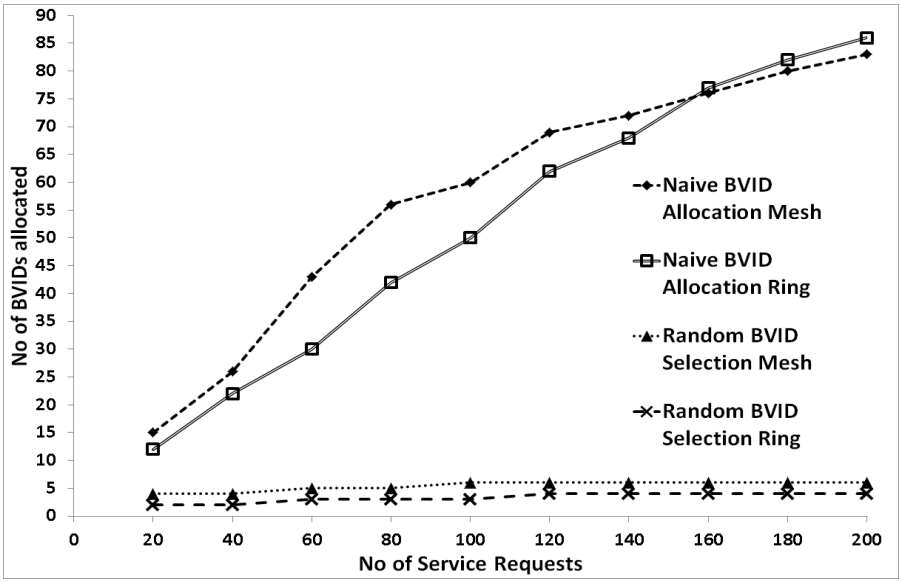


Fig. 2.26. BVIDs allocated for Ring and Mesh network topologies using a naïve and the proposed Random BVID Selection algorithm.

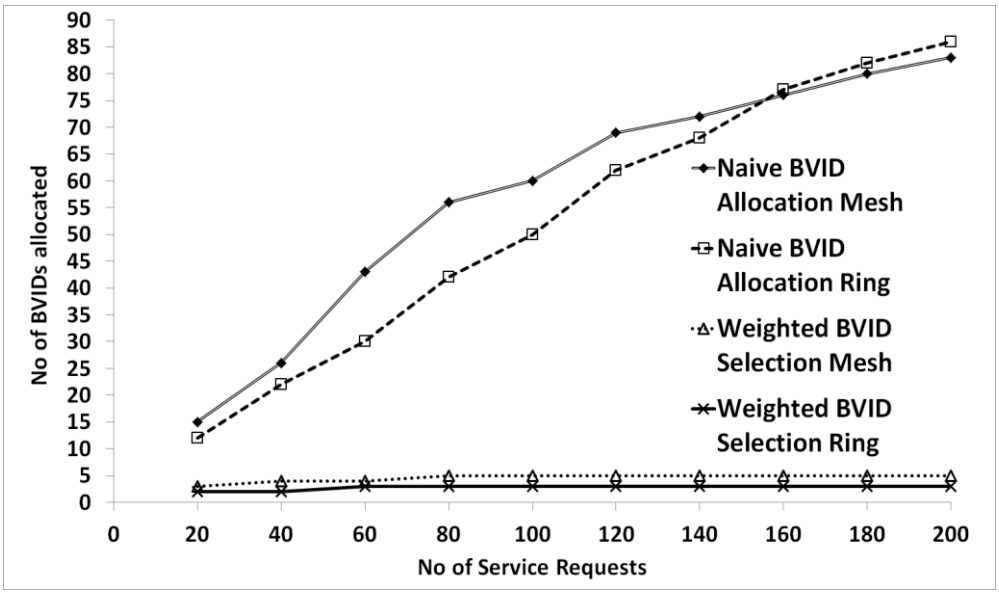


Fig. 2.27. BVIDs allocated for Ring and Mesh network topologies using an naïve and the proposed Weighted BVID Selection algorithm.

In Fig. 2.28 we present the number of connection requests satisfied by a naïve heuristic and the proposed heuristics keeping the available BVIDs as constant. The graph demonstrates that a significantly large number of connection requests (around 10,000) can be satisfied with the proposed heuristics as compared to the naïve BVID

allocation algorithm (around 350) with a limited number of BVIDs available (<128). We also generated results for larger datasets to the point where all the available 4094 BVIDs in the network and these were exhausted by the naïve algorithm. Table 2.3 presents a comparison between the proposed heuristics and a naïve algorithm for larger data sets.

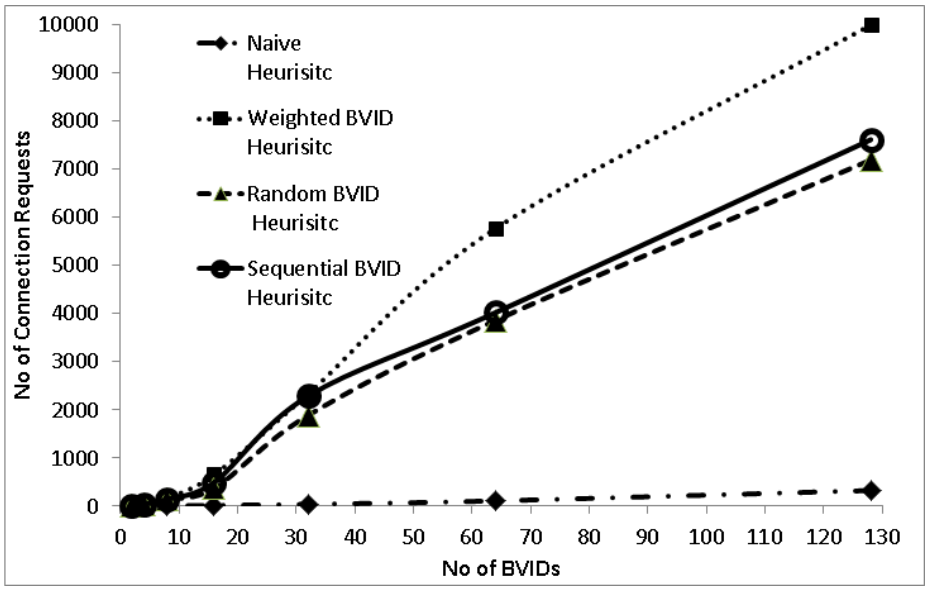


Fig. 2.28. Number of requests satisfied keeping the number of BVIDs constant.

Table 2.3 presents results with respect to the number of BVIDs required to satisfy all the connection requests in a given network. For this purpose, we considered different sized networks, with nodes varying from 10 to 50 typical to a metropolitan deployment. In addition, we considered different topologies such as: (1) Ring, (2) Sparse Mesh and (3) Dense Mesh. For simplicity, we considered that each ELINE service has a granularity of 3Mbps. We observe from Table 2.3 that the "*Weighted Links and Weighted BVID Allocation"* heuristic performs the best among the four proposed heuristics. In all the four proposed BVID allocation heuristics there is a significant improvement compared to the naïve implementation. The results demonstrate that without implementation of a BVID Allocation algorithm, the management system could run out of the available BVIDs and connection requests may remain non-provisioned (ringing problem) in the network (refer to rows 47 and 48 in Table 2.3 below). However, with the implementation of the proposed BVID Allocation algorithms, all these requests can be satisfied with a significantly lessser number of BVIDs.

Data Set	No of Nodes	No of Connection requests	Link Density	Non-Optimal BVID Allocation Algo		Random BVID Selection Algo		Sequential BVID Selection Algo		Weighted BVID Selection Algo		Weighted Links & Weighted BVID Selection Algo	
				BVIDs Required	CPU Time (ms)	BVIDs Required	CPU Time (ms)	BVIDs Required	CPU Time (ms)	BVIDs Required	CPU Time (ms)	BVIDs Required	CPU Time (ms)
1	10	20		15	9	3	14	3	15	2	36	2	116
2		40	Ring	25	18	4	27	4	29	3	69	2	166
3		60		43	54	4	81	4	89	3	174	3	222
4		80	Mesh-sparse	56	72	5	110	5	112	3	233	3	278
5		100		62	135	5	210	5	209	3	439	3	353
6		120	Mesh-Dense	73	162	5	243	5	252	4	495	3	450
7	15	50		35	54	6	84	7	90	3	182	3	564
8		60	Ring	46	63	7	99	7	96	4	216	3	700
9		90		63	189	8	279	8	284	5	575	5	866
10		100	Mesh-sparse	66	210	8	315	9	320	6	634	5	1069
11		140		68	441	9	657	9	669	6	1321	5	1310
12		160	Mesh-Dense	77	504	9	756	9	766	6	1526	5	1599
13	20	80		59	152	8	228	9	229	7	471	5	1939
14		100	Ring	65	190	9	287	10	292	8	591	5	2352
15		140		69	532	8	798	8	802	8	1605	6	2836
16		160	Mesh-sparse	83	608	9	924	9	917	8	1867	6	3419
17		180		82	926	9	1389	9	1391	9	2785	6	4131
18		200	Mesh-Dense	85	988	9	1482	10	1490	9	2984	6	4986
19	30	200		93	870	11	1305	12	1311	9	2614	6	6009
20		300	Ring	118	1305	14	1967	13	1966	9	3935	6	7222
21		400		145	1810	17	2715	16	2720	9	5443	8	8678
22		500	Mesh-sparse	187	2250	17	3375	17	3378	10	6764	8	10443
23		600		225	4050	17	6075	18	6085	12	12175	8	12543
24		800	Mesh-Dense	303	5440	19	8160	19	8170	14	16334	9	15068
25	40	400		149	2200	18	3300	19	4210	13	6629	10	18107
26		600	Ring	233	3330	21	4995	21	5001	15	10001	10	21744
27		800		314	8180	22	12270	23	12280	16	24552	11	26122
28		1000	Mesh-sparse	442	10450	23	15675	23	15682	17	31374	11	31358
29		1200		567	12310	26	18465	27	18466	18	36931	11	37651
30		1600	Mesh-Dense	788	14270	28	21405	29	21410	21	42821	13	45199

TABLE 2.3: Comparison between different heuristics for larger data-sets.

2.6 Conclusions

In this chapter, we have addressed the problem of optimal BVID allocation in PBB-TE networks. The performance of PBB-TE networks is tightly coupled with optimal BVID allocation making this an important problem for the proliferation of Carrier Ethernet. We propose an optimization scheme that is a generalization of our preliminary work in [8, 105]. We have also proposed a set of heuristics that solve the BVID allocation problem in tractable time and whose performance is comparable to the otherwise intractable optimization scheme.

Data Set	No of Nodes	No of Connection requests	Link Density	Non-Optimal BVID Allocation Algo		Random BVID Selection Algo		Sequential BVID Selection Algo		Weighted BVID Selection Algo		Weighted Links & Weighted BVID Selection Algo	
				BVIDs Required	CPU Time (ms)	BVIDs Required	CPU Time (ms)	BVIDs Required	CPU Time (ms)	BVIDs Required	CPU Time (ms)	BVIDs Required	CPU Time (ms)
31	50	800		322	7312	24	10968	26	10975	18	21938	12	54264
32		1000	Ring	462	8750	25	13125	28	13133	19	26269	12	65145
33		1200		578	10760	28	16140	31	16141	20	32294	13	78193
34		1400	Mesh-sparse	687	12255	27	18450	29	18389	21	36915	13	93846
35		1800		856	15640	29	23460	31	23466	22	46938	15	112643
36		2000	Mesh-Dense	930	17670	31	26505	33	26506	25	53026	19	135206
37	100	2000		1036	21356	43	31452	47	32556	35	57113	21	157113
38		2200	Ring	1566	26031	51	41412	57	42841	39	74452	23	236012
39		2400		1998	31526	63	54550	69	56053	45	97317	27	354893
40		2600	Mesh-sparse	2278	38255	75	71462	82	73345	53	126719	30	533261
41		2800		2564	46115	91	93499	98	95863	57	165083	32	800871
42		3000	Mesh-Dense	2821	55692	104	121988	113	125251	69	215155	33	1201852
43	200	3000		2798	67287	138	159209	139	163596	78	280357	38	1803424
44		3400	Ring	3267	81180	159	207349	171	212967	90	365202	44	2705899
45		3800		3521	97690	181	270137	199	277090	109	475358	49	4059277
46		4200	Mesh-sparse	3868	117749	206	351392	219	360520	121	618699	56	6089846
47		4600		4156	141887	219	457353	234	469421	135	804816	67	9135103
48		5000	Mesh-Dense	4478	170821	248	594902	278	610497	151	1046611	79	13703485

TABLE 2.3: Comparison between different heuristics for larger data-sets.

Specifically, we have proposed four heuristics to minimize the number of BVIDs required to satisfy dynamic service (connection) requests in a network. A general Integer Linear Program formulation is also presented to solve the problem with static service requests and a bounded number of BVIDs. We aim to maximize the service requests satisfied in the network with a limited number of BVIDs in this case.

The results obtained by implementing the heuristic algorithms and comparing them with a naïve BVID allocation algorithm are showcased. Results show that the heuristics compare favorably with the optimal algorithm in terms of computational times and size of problems that can be solved without compromising on the solution quality. We also conclude that it is impossible for the optimal approach to solve practical-sized (or real-world) problems. For such real-world problems, if we use the proposed heuristics, the solution quality is acceptable, and is reasonably close to the optimal as shown through the simulations.

Chapter 3

Multilayer Optimization for Service Provider Transport Networks

Service provider networks are becoming increasingly complex with multi-domain and multilayer capabilities. Multilayer optimization has been proposed as a mechanism for planning provider-networks focused on optimizing revenue and reducing capital expenditure. In this chapter, we study multilayer optimization from the perspective of deploying services, to reduce the capital expenditure in provider-networks. To this end we propose a 3-layer network hierarchical model based on IP, OTN and DWDM technologies. The goal of this work is to ascertain the type and amount of traffic that should be routed through a particular layer of the network, to reduce the total cost of ownership. In this chapter, we propose an optimization model, a heuristic algorithm and their solution methods. While the optimization model deals with the network planning cases that involve static traffic demands, the heuristic algorithm solves the dynamic

case. We also develop a simulation model to validate our optimization and heuristic approaches.

3.1 Introduction

Internet traffic has grown at the rate of 40-100% per year and is expected to grow with the same rate or more aggressively in the future [37, 38]. Due to the enormous growth in *IP* traffic volume, network operators are confronted by the unprecedented challenge of accommodating the *IP* traffic volume in already deployed networks in a short time-span. In addition, network operators observe stringent latency and reliability requirements [37-41]. The overall effect of these aforementioned problems has been sub-optimal utilization of network resources resulting in higher network-wide capital and operational expenditures; it has also resulted in lesser revenues. Hence, there is a need to optimize the network cost while accommodating maximum traffic volume in the network. In this work, we consider an optimization problem in a 3-layered network comprising *IP* as the topmost layer followed by the *OTN* layer that acts as opto-electro-opto (*OEO*) cross-connect and *DWDM* layer being physical layer.

Contemporary networks are multilayered, where two or more layers together perform the task of routing and packet forwarding. For example, *IP* over *SONET/SDH* or *IP/MPLS* over *DWDM* [39-42] are practical deployments in contemporary networks. A significant amount of work has been conducted on the design of optimal multilayer networks. However, the focus, primarily, has been on design and optimization in two-layer networks. The work presented in [43-45] focuses on optimization in *IP/MPLS* over *WDM* networks. The work presented in [46] focuses on *IP* over *SONET/SDH*. The solution discussed in [41, 47-49] focuses on the survivability of *IP/MPLS* over *WDM* networks. Furthermore, work presented in [50] considers the *OTN* layer in the optimization model, but models this layer as an embedded layer in the *WDM* layer and not as a separate layer. The work in [51] considers the *OTN* layer, but from a perspective of traffic uncertainty. Further, work done in [37] considers the *OTN* as a separate layer; however, the emphasis has been on traffic grooming, studying the basic advantages of *ODU*-switching. The work presented in [42, 52] has considered *OTN* as a

separate layer in 3-layer networks. The work presented in [108] has considered *OTN* layer from a capacity planning perspective with the 3-layer network model, which we have aptly modeled from the perspective of a service-provider networks.

In this chapter, we consider a network capacity planning problem, specifically, for a 3-layer network hierarchy comprising an IP layer, the OTN layer and the optical DWDM layer. The network architecture view of such an IP-OTN-DWDM multilayer model is shown in the Fig. 3.1. Interconnectivity between the different layers at a particular node and across nodes is displayed in Fig. 3.2 and Fig. 3.3, below. We differentiate between the *OTN* and the *DWDM* layer, in the sense that, the traffic between two nodes (inclusive of the pass-through traffic) is such that the total bandwidth can be supported by just a single *OTN* connection. We then map such a connection using a single fiber (as shown in Fig. 3.2). This means is that the fiber has no *DWDM* channels, but just a single *OTN* connection. This kind of communication is facilitated by *ODU*-switching at the ingress and egress nodes that also provide for *OTN* layer multiplexing and de-multiplexing [50, 53].

3.2 Multi-Layered Traffic Models

We now consider the following traffic models that would be used for our proposed algorithm. As we will see, the models are multilayered consisting of *IP* over *OTN* over *DWDM*.

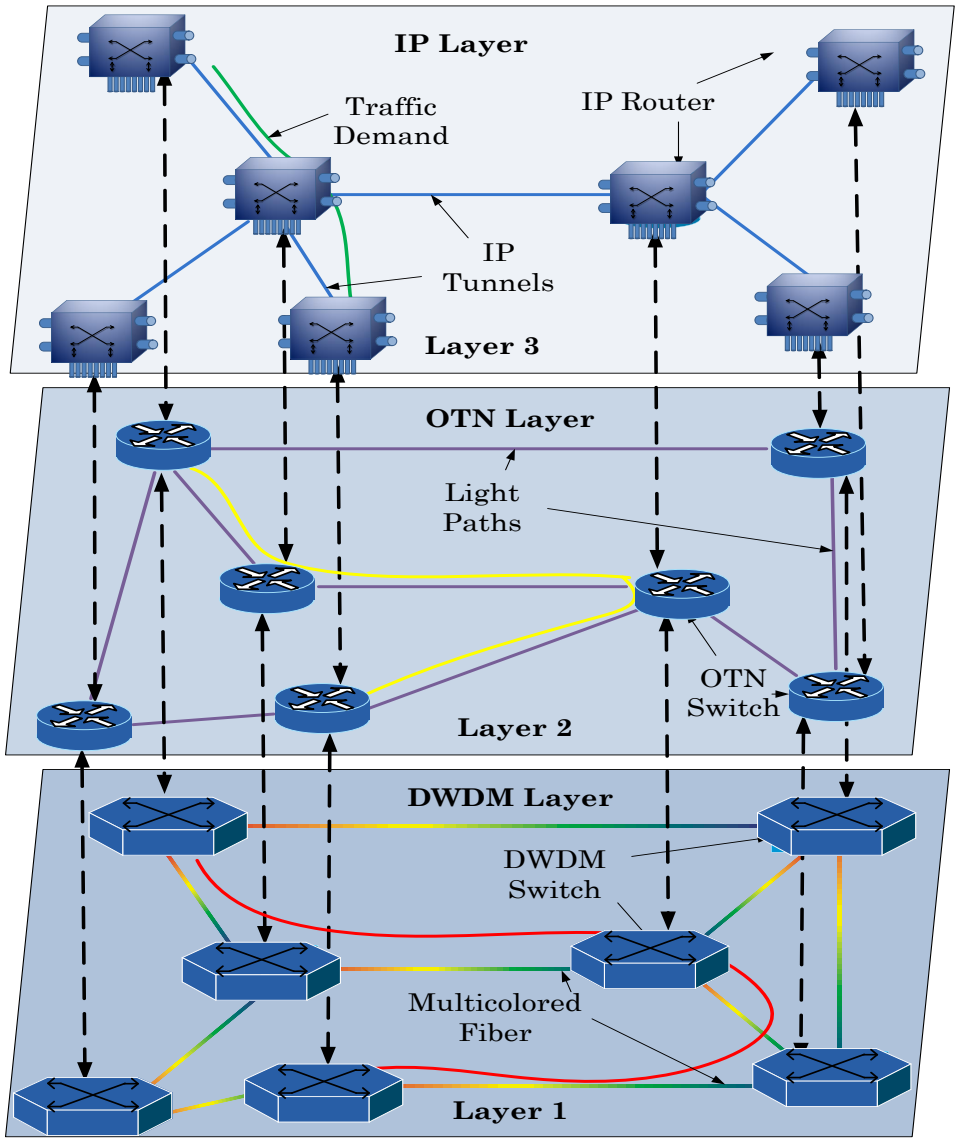


Fig. 3.1. IP-OTN-DWDN Multilayer Correlation.

3.2.1 IP-centric model

In this case, demands arrive at the *IP* layer and are routed through the network. Demands are either encapsulated in an *OTN* frame that is transported directly over the fiber or encapsulated in an *OTN* frame that is further mapped onto a wavelength and then sent through the *DWDM* layer. In this model, the signal is necessarily regenerated at each node and sent to an *IP* router for packet forwarding decisions. The *IP* router forwards the packets based on network-centric identifiers (longest prefix matching) and

then sends the packets to the corresponding egress interfaces (at the node). The *IP*-centric model is the most expensive model as it requires regeneration at each node in the network and in addition, IP-level forwarding of packets. However, it is also the most flexible model in terms of its ability to cater to packet-mode communications [45].

3.2.2 OTN-centric model

In this case, the traffic demands are routed as *ODU0*, *ODU1*, *ODU2* or *ODU3* signals (or even as *ODU2-flex*) at the *OTN* layer. The demands are mapped to the *OTN* layer in such a way that the *ODU-x* format is chosen, where *x* is the next highest line-rate that the demand can be fed to. In this model, due to the expected difference between traffic demand and the rigidity in the *OTN* layer granularities, there is some efficiency degradation [42, 52]. In this work we assume that multiple lower-tributary *ODU-x*'s with a common destination are already multiplexed together to form a thicker pipe at the ingress and egress. However, such multiplexing, de-multiplexing and switching are not allowed at the intermediate nodes. Such a kind of *ODU* switching is not popular in dynamic optical networks, primarily because of the uncertainty in equipment requirement at the intermediate nodes [54]. We, however, support *OTN* switching between ports of an *OTN* Add-Drop Multiplexer, wherever necessary.

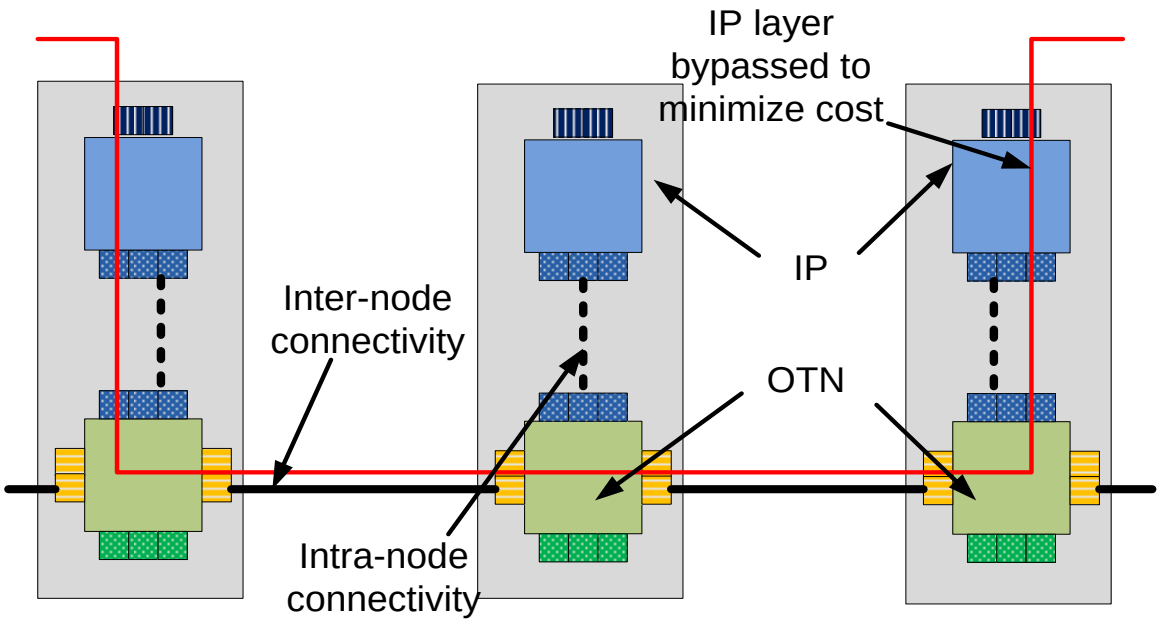


Fig. 3.2. Design approach and Routing options: *IP+OTN* only.

3.2.3 All-Optical model

In this case, traffic demands arrive as requests that are directly mapped over wavelengths using multi-rate transponders. Transponders can be of rates 2.5Gbps, 10Gbps and 40Gbps. Transponders send a wavelength that is multiplexed in the frequency domain by wavelength-selective switches of a *ROADM*. We assume a colorless, directionless and contention-less *ROADM* [55, 57].

Given a set of traffic demands and network deployment over a planning horizon, our goal in this work is to minimize the traffic provisioning cost from a capital expenditure perspective. To this end, an optimization model is proposed. The optimization model takes into consideration the three models mentioned earlier and chooses the appropriate model for each demand, while minimizing network-wide cost.

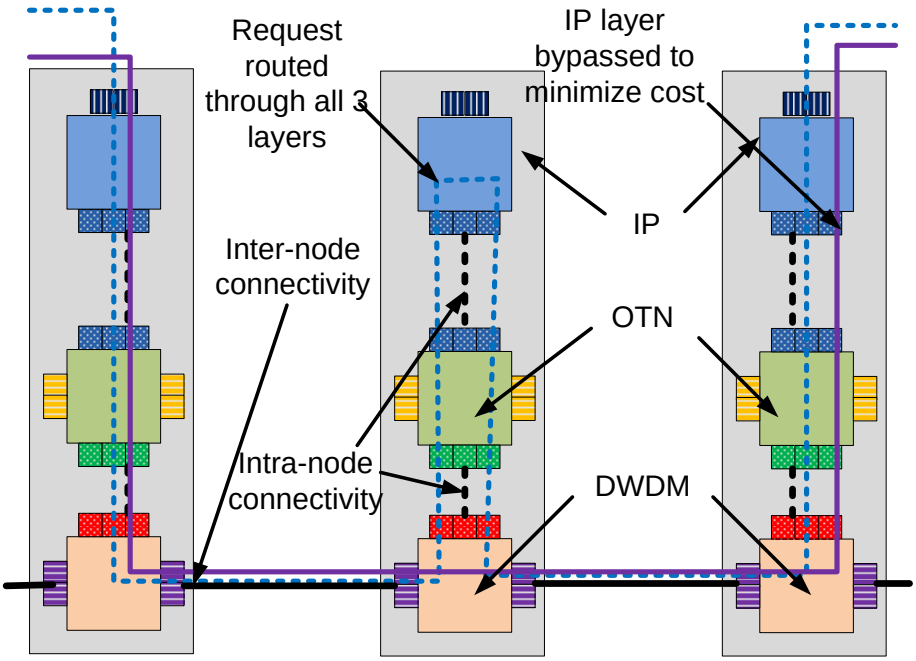


Fig. 3.3. Design approach and Routing options: *IP+OTN+DWDM*.

3.3 Optimization Model and Heuristic Algorithm

In this section we present a constrained optimization model for a 3-layer network capacity planning problem. The traffic demands are realized by the means of flows

assigned to paths of the *IP* layer. We have assumed that a matrix of traffic demand at the *IP* layer is given in advance. Every traffic demand in this matrix originates and terminates at the *IP* layer. However, a particular traffic demand can be routed using one of the three models, as explained in the previous section. In addition, we assume that the set of links at *OTN* and *DWDM* layer are also given.

The ILP works well for small sized networks. For contemporary provider networks that have a large number of core nodes, such as nation-wide backbone topologies, the ILP provides results only after significant computational overhead. As a result, it is desirable to develop a fast heuristic that can obtain quick results while preserving the correctness of the solution in-line with what is expected from the LP. The dynamic traffic case, that is, when the traffic matrix is not known a *priori* is also considered by the heuristic.

3.3.1 Optimization Model (ILP)

A typical network consists of edge and core devices. In [51, 55], it has been proved that the optimization problem on edge devices is trivial. Hence, we consider capacity allocation on core devices only. This eliminates the local traffic at every node and only transit traffic that passes through a node of an appropriate layer [55], is considered. The objective of the resulting optimization model is to minimize the total cost of the interfaces to be installed on all the three layers, while all the traffic demands are satisfied. An *Integer Linear Program* (ILP) formulation is described below. For the sake of simplicity we will refer to the formulation as the *Linear Program* (LP).

Indices:

T: Number of traffic demands at the *IP* Layer.

P: Number of candidate logical paths in the *IP* Layer (or *IP*-tunnels) for every traffic demand *t*, $t \in T$.

L: Number of links in the *IP* Layer. Please note that these are the logical tunnels which will be implemented either by the *OTN* or the *DWDM* layer.

M: The number of links in the *OTN* Layer.

N: The number of links in the *DWDM* Layer.

I = {1, 2, 3, 4}: Set of modular interfaces of an *IP* layer, for example, 1G, 10G, 40G and 100G

O = {1, 2, 3, 4}: Set of modular interfaces of an *OTN* layer, for example, 1.25G, 2.5G, 10G and 40G.

W = {1, 2, 3}: Set of modular interfaces (in this case transponders) of a *DWDM* layer, for example, 2.5G, 10G and 40G.

Input Constants:

V: Granularity of traffic demands.

$V = \{v_1, v_2 \ldots v_t\}$, $t \in T$

We assume that the traffic demands are bi-directional. It means that if v_1 is a volume of traffic demand between nodes i and j, then the same volume of traffic needs to be provisioned from j to i.

A set of different interfaces at every layer, their capacities and costs are given below.

Ω: Capacity of *IP* interfaces. Or $\Omega = \{\omega_1, \omega_2 \ldots \omega_i\}$, $i \in I$,

for example, $\Omega = \{1, 10, 40, 100\}$.

Θ: Capacity of *OTN* interfaces. Or $\Theta = \{\theta_1, \theta_2 \ldots \theta_o\}$, $o \in O$, for example, $\Theta = \{1.25, 2.5, 10, 40\}$.

Λ: Capacity of *DWDM* interfaces. Or $\Lambda = \{\lambda_1, \lambda_2 \ldots \lambda_w\}$, $w \in W$, for example, $\Lambda = \{2.5, 10, 40\}$.

A: Cost of *IP* interfaces. Or $A = \{\alpha_1, \alpha_2 \ldots \alpha_i\}$, $i \in I$.

B: Cost of *OTN* interfaces. Or $B = \{\beta_1, \beta_2 \ldots \beta_0\}$, $o \in O$.

Γ: Cost of *DWDM* interfaces. Or $\Gamma = \{\gamma_1, \gamma_2 \ldots \gamma_w\}$, $w \in W$.

Note that we will be considering the maximum $Z*log(D)$ number of paths eligible for any traffic demand in $C^{P\ X\ T}$ out of the total paths available in the network (selection is done using greedy approach), where Z is an integer constant and D is a diameter of the network (by diameter we meant the distance between two farthest node in the graph). This assumption reduces the size of the LP significantly and makes it tractable. It is a common practice to assume that all routing paths are set up according to *Open Shortest Path First* (*OSPF*) protocol [51, 56] and will be given in advance as input. However,

this is not the case in most real-time scenarios. A network designer needs to consider more than one possible candidate paths to provision a particular traffic demand due to constraints such as link capacity [55].

Consider $C^{P\ X\ T}$ as an input matrix to the ILP, specifying candidate paths to satisfy a given traffic demand $t \in T$. In our case we, have considered different values of $Z = \{1, 3, 5, 7, 9\}$ to generate the results. The rest of the input constants are mentioned below.

$$C^{P\ X\ T} = \left\{ \begin{matrix} 1, \text{if path } p \text{ can be used to satisfy traffic} \\ \text{demand } t \text{ at } IP \text{ layer or 0 otherwise} \end{matrix} \right\},\ t \in T, p \in P$$

$$\delta^{L\ X\ T} = \left\{ \begin{matrix} 1, \text{if link } l \text{ can be used to comprise a path} \\ p \text{ at } IP \text{ layer or 0 otherwise.} \end{matrix} \right\},\ l \in L, p \in P$$

$$\psi^{M\ X\ L} = \left\{ \begin{matrix} 1, \text{if link } m \text{ at } OTN \text{ layer can be used to} \\ \text{implement link } l \text{ at } IP \text{ layer or 0 otherwise.} \end{matrix} \right\}, m \in M, p \in P$$

$$\rho^{N\ X\ M} = \left\{ \begin{matrix} 1, \text{if link } n \text{ at } DWDM \text{ layer can be used to} \\ \text{implement link } m \text{ at } OTN \text{ layer} \\ \text{or 0 otherwise.} \end{matrix} \right\}, m \in M, n \in N$$

In practice, there is a limit on the number of wavelengths that can be supported by the physical links. We consider that every fiber link in the network topology can allow a maximum of 800Gbps traffic to pass through (80 wavelengths per fiber) [49].

Variables:

We define a variable $g^{P\ X\ T}$ to indicate which path is actually used amongst the available candidate paths to satisfy a particular traffic demand.

$$gP\ X\ T = \left\{ \begin{matrix} 1, \text{path } p \text{ is used to satisfy traffic} \\ \text{demand } t \text{ or 0 otherwise.} \end{matrix} \right\}, t \in T, p \in P$$

For each traffic demand $t \in T$ we define a set of binary variables b_t^{IP}, b_t^{OTN} and b_t^{DWDM}. b_t^{IP} is 1 if traffic demand t is to be routed using the *IP*-centric model, else it is 0. The same is the case for variables b_t^{OTN} and b_t^{DWDM} at *OTN* and *DWDM* layers, respectively.

$$b_t^{IP} = \left\{ \begin{matrix} 1, \text{if traffic demand } t \text{ is to be routed on} \\ IP \text{ layer or 0 otherwise.} \end{matrix} \right\}, \forall t \in T$$

$$b_t^{OTN} = \left\{ \begin{matrix} 1, \text{if traffic demand } t \text{ is to be routed on} \\ OTN \text{ layer or 0 otherwise.} \end{matrix} \right\}, \forall t \in T$$

$$b_t^{DWDM} = \left\{ \begin{matrix} 1, \text{if traffic demand } t \text{ is to be routed on} \\ DWDM \text{ layer or 0 otherwise.} \end{matrix} \right\}, \forall t \in T$$

We define a set of variables to indicate the number of interfaces to be installed on every link of a particular layer so that total traffic demands on that layer are satisfied. We recall that a traffic matrix *V* for each traffic demand *t* and a path matrix *C* are given as an input to the LP.

x^{LXI} = Number of *IP* interfaces *i* to be installed on the *IP-OTN* link *l*, $l \in L$, $i \in I$.

y^{MXO} = Number of *OTN* interfaces *o* to be installed on *OTN*-layer link *m*, $m \in M$, $o \in O$.

z^{NXW} = Number of *DWDM* interfaces w to be installed on *DWDM*-layer link *n*, $n \in N$, $w \in W$.

Objective:

Our objective is to minimize the network cost by minimizing the cost of the interfaces at every single layer. Hence, our objective function becomes:

Minimize:

$$\sum_{l \in L} \sum_{i \in I} x^{LXI} A + \sum_{m \in M} \sum_{o \in O} y^{MXO} B + \sum_{n \in N} \sum_{w \in W} z^{NXW} \Gamma$$

We seek to minimize the above function satisfying the following constraints.

Constraints:

1. The first constraint ensures that any traffic demand $t \in T$ must be satisfied by only one path out of the candidate paths in C^{PXT}.

$$\sum_{p \in P} g^{PXT} \leq 1, \forall t \in T$$

2. We now define an integrity constraint. For a given traffic demand $t \in T$, a path from candidate paths $C^{P\,X\,T}$ only must be selected as an actual path to satisfy that particular demand in $g^{P\,X\,T}$.

$$g^{PXT} \leq C^{PXT} \quad \forall t, p$$

3. We need to ensure that each traffic demand is satisfied on only one layer out of the three available layers.

$$b_t^{IP} + b_t^{OTN} + b_t^{DWDM} \leq 1 \quad \forall t \in T$$

4. The IP interfaces installed on the Layer-1 should accommodate all the traffic passing through each link $l \in L$, comprising the different paths $p \in P$.

$$\sum_{t \in T} V_t * b_t^{IP} \sum_{p \in P} \delta^{LXP} g^{PXT} \leq \sum_{i \in I} x^{LXI}\, \Omega, \ \forall\ l \in L$$

5. We now consider the *OTN* link capacity constraint. The Layer-2 links $m \in M$ should have sufficient interfaces installed to accommodate a link $l \in L$ in the upper *IP* layer.

$$\sum_{t \in T} V_t * b_t^{OTN} \sum_{l \in L} \psi^{MXL} \sum_{p \in P} \delta^{LXP} g^{PXT} \leq \sum_{o \in O} y^{MXO}\, \Theta, \ \forall m \in M$$

6. The capacity of each *OTN* link $m \in M$ is the demand to be satisfied by the DWDM layer links $n \in N$.

$$\sum_{t \in T} V_t * b_t^{DWDM} \sum_{m \in M} \rho^{NXM} \sum_{l \in L} \psi^{MXL} \sum_{p \in P} \delta^{LXP} g^{PXT} \leq \sum_{w \in W} z^{NXW}\, \Lambda, \ \forall n \in N$$

7. Link capacity constraint can be written as follows.

$$\sum_{i \in I} x^{LXI}\, \Omega \leq 100, \ \forall\ l \in L \text{ (for IP layer)}$$
$$\sum_{o \in O} y^{MXO}\, \Theta \leq 100, \ \forall\ m \in M \text{ (for OTN layer)}$$

$$\sum_{w \in W} z^{NXW} \Lambda \leq 100, \forall n \in N \text{ (for DWDM layer)}$$

We will be considering normalized costs for the interfaces that we have mentioned in the LP above. We have considered a cost matrix shown in Table 3.1 below. Note that these are not the actual costs of the components, but the relationship between them, based on a model defined by European research project, to be the same as that in [40, 55, 108]. While considering the interfaces on the *DWDM* layer, we will be considering the costs for transponders as well as muxponders. In addition, we have assumed that all the cost values of this table refer to complete bidirectional network elements [39, 41-42].

Layer and Interface		Capacity	Cost
IP	Interface 1	1G	0.35
	Interface 2	10G	1.25
	Interface 3	40G	8.625
	Interface 4	100G	25.625
OTN	Interface 1	1.25G	0.35
	Interface 2	2.5G	0.45
	Interface 3	10G	1.5
	Interface 4	40G	5
DWDM	Interface 1	2.5G	0.33
	Interface 2	10G	2.17
	Interface 3	40G	8.25

TABLE 3.1: Cost model for different interfaces.

3.3.2 Heuristic Algorithm

In our heuristic solution, we have used a top-down greedy approach to provision the traffic demands in the network. The heuristic constructs an initial multilayer network graph that consists of physical fiber links in the *DWDM* layer. It then assigns a fiber capacity in the *DWDM* layer based on the demand volume and attempts to accommodate demands across the three layers. All the links in this topology are

assumed to be bidirectional. The optical links can have multi-wavelength fibers [52, 54, 57].

We assume that the topology matrix and traffic matrix are input to the heuristic. The topology matrix is a connectivity matrix between the nodes at that particular layer. In addition, we are given a set of interfaces that can be deployed at the respective layer with the capacity, along with the cost of those interfaces. Traffic demand is a tuple comprising a source node, a destination node and the desired granularity for that traffic demand. The proposed heuristic provisions each traffic demand as it arrives at the source node.

The heuristic calculates the candidate paths for each traffic demand using the three traffic models. Extra interfaces are installed depending on whether there is at least one path from the set of candidate paths to accommodate the traffic demand or not. The traffic demand is discarded if no such path is found in any of the layers or no extra interfaces could be installed. Once all the paths are calculated at all the three layers, a procedure is invoked to allocate appropriate interfaces and calculate the total cost. The heuristic decides the layer at which the demand is to be satisfied depending on the cost of the interfaces at each layer. For each OTN link, multiplexing is done for the incoming traffic demand to minimize the cost. The heuristic bypasses the *IP* layer at a particular node whenever no routing decision is to be taken at that node, depending on the input traffic matrix. The choice between the *OTN* and the *DWDM* layer is based on the minimum cost of the required interfaces. Note that D is the diameter of the network and Z is some integer constant, in our case $Z = \{1, 3, 5, 7, 9\}$.

Algorithm: *IP-OTN-DWDM*

Input: *Graph G(V, E), H(V, F) and P(S, T) representing the existing topology in IP, OTN and DWDM layers respectively.*

Output: Total cost of all the interfaces at each layer

procedure main

for *each incoming traffic demand* $t(s, d, g) \in T(S, D, G)$

- *find feasible candidate paths for traffic demand t as per the value of Z and diameter D.*
- *sort the paths in ascending order of the no-of-hops on that particular path.*
- ***for** each path p in the sorted list*
  - *at each node on the path, check whether the routing decision is to be made on that node.*
  - ***if** no routing decision to be made, discard the IP layer implementation for that node (IP-transparent implementation).*
  - ***if** remaining_capacity of the interfaces at the OTN layer for the node >= g, select OTN layer for that node.*
  - *C_{OTN} = total cost of interfaces at OTN layer*
    C_{DWDM} = total cost of interfaces at DWDM layer.
  - ***if** $C_{OTN} < C_{DWDM}$, select OTN layer to provision the traffic demand else select DWDM layer.*

  end

- *select a path p on which total cost of the interfaces is minimum.*
- *return selected path, selected layer and selected interfaces on that particular layer for traffic demand t.*

end

end procedure main

The heuristic returns the total cost of all the interfaces and allocates appropriate interfaces on the selected layer. Finally all the link capacities and remaining interface capacities are adjusted accordingly.

3.3.2.1 Time Complexity:

We now discuss the time complexity of the proposed heuristic. We observe that the complexity of the algorithm is bounded by the complexity of the candidate path calculation module. Recall that the number of nodes in the network is denoted by N. We select Z*log$D$ shortest paths as candidate paths for a particular traffic demand where D is the diameter of the network and Z is some integer constant. In the worst case scenario, $D = N$. We have implemented Dijkstra's algorithm to find the shortest paths. Hence, the complexity of the module to find K ($K = Z$*log$D$) shortest path for a traffic demand is O($Z*N^2*logD$) or O($Z*N^2*logN$). Since Z is a small constant, the complexity is approximated to O(N^2*logN). Now, let us concentrate on the number of traffic demands

in the network. The total set of traffic demands in the network is bounded by O(N^2). Hence, the complexity is O(N^2*N^2*$logN$) or O(N^4*$logN$).

3.4 Performance and Simulation

In this section, we present the results of computational experiments that demonstrate the performance of the proposed heuristic and compare them with the performance of the LP. We have generated random cluster graphs using the "*networkx*" library [58] and Python. We have considered a network size with the nodes ranging from 100-1000, where the average degree of a node is three (classical metropolitan networks). Traffic demands and all the links in this topology are assumed to be bi-directional. Average traffic volume to be generated for the simulation is 8.5 TB. We have generated different sets of traffic demands where the rate of arrival traffic requests is characterized by a *Poisson* distribution and where service granularity is exponentially distributed. Finally, we have taken an average of all results for all sets of traffic demands. In our simulation we consider that a fiber can accommodate 80-wavelengths, though a larger number of wavelengths should not have an impact on the nature of the results. The cost ratios of *DWDM* transponders/muxponders, *IP*-layer and *OTN*-layer interfaces are as shown in Table 3.1.

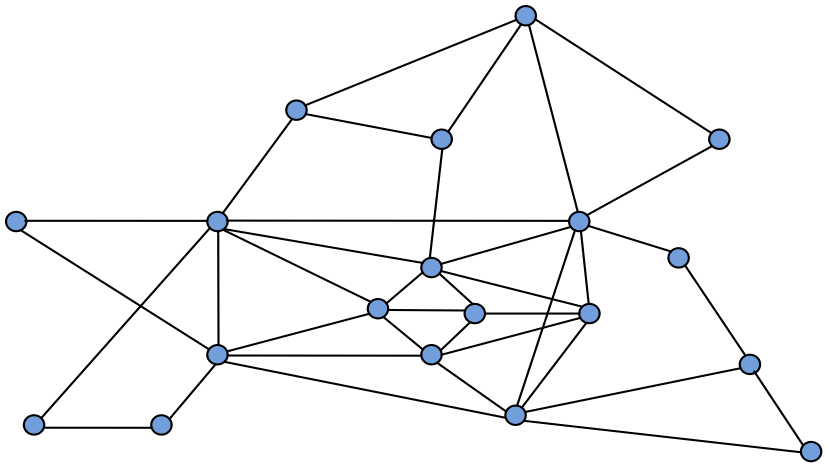


Fig. 3.4. A 19-node EON Topology.

The graph in Fig. 3.5 shows a comparison between the LP and the heuristic for Z = 3. We have considered smaller topologies, networks of up-to 50-nodes with two different sets of traffic requests for the comparison. The first set has 100 traffic demands while the second set has 200 traffic demands. As mentioned in section III, the ILP solves only smaller data-instances due to a large number of variables and its computational complexity.

It can be observed from the graph below that the total cost of the interfaces with the IP-centric approach (only-*IP*) is substantially higher (that is. 236.42 units and 415.95 units for the two sets of traffic demands, respectively) as compared to the optimal cost (127.65 units and 193.55 units, respectively). In addition we can observe that the heuristic with 3-layer network architecture shows a significant improvement in the total cost of the interfaces in the network (136.86 units and 210.27 units, respectively).

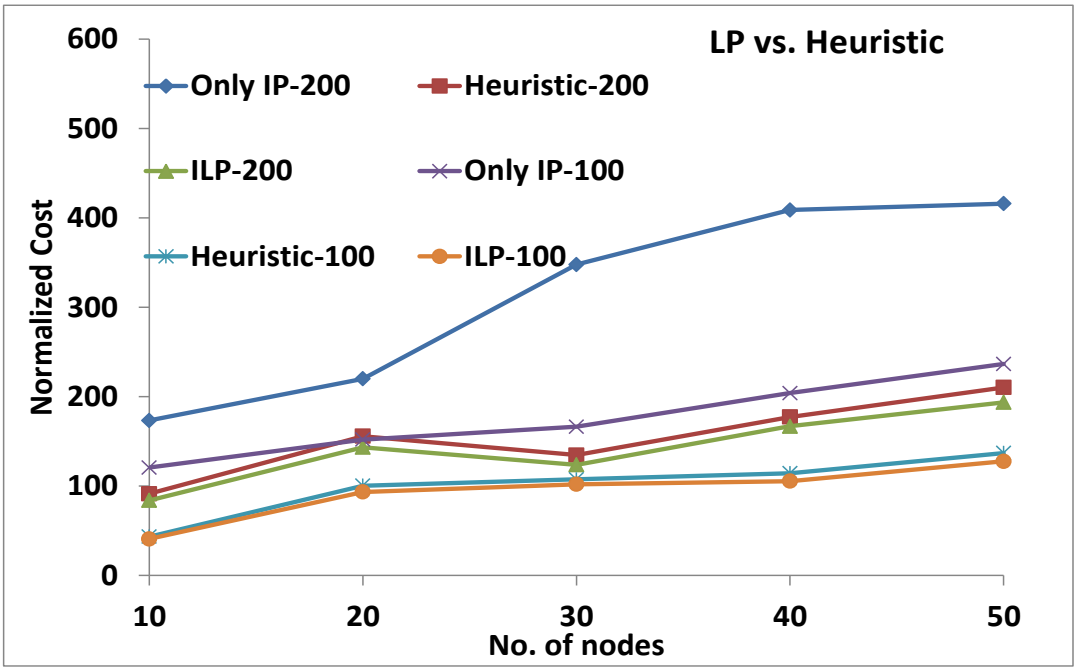


Fig. 3.5. ILP versus Heuristic—Till 50 Nodes, Z = 3.

Fig. 3.6 displays a graph where the performance of the heuristic is compared against that of the LP in a real-world network scenario. A 19-node *European Optical Network* (EON) topology (as shown in Fig. 3.4) has been considered for the same. We have considered a set of 100 traffic demands. We have compared the heuristic and the

LP results in the *EON* topology for $Z = 3$ and $Z = 5$. Graphs shown in Fig. 3.5 and Fig. 3.6 demonstrate that the heuristic matches the LP solution closely within a range of 10-12%. It should be noted that the branch and bound LP actually deploys an approximation—choosing the top-Z*log$D$ shortest paths. The total cost in the only-*IP* networks continues to be much higher compared to the total cost in 3-layered networks.

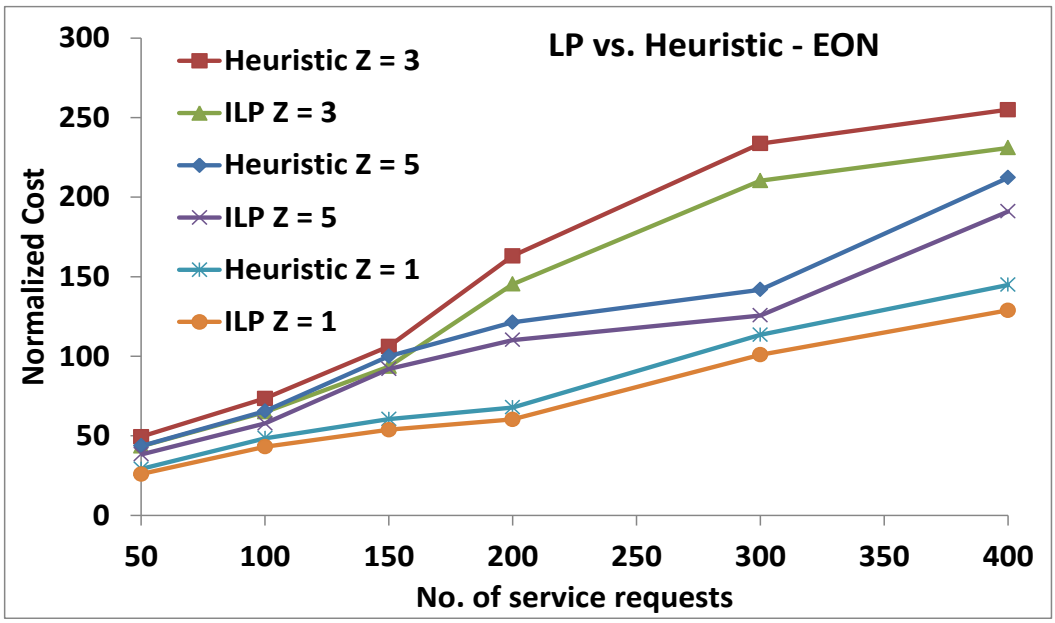


Fig. 3.6. ILP versus Heuristic—*EON* Topology, $Z = 3$ & 5.

In Fig. 3.7 we have plotted a graph of the network cost generated by the LP with Z = {1, 3, 5, 7}. It can be observed from the graph that as the value of Z increases, the total cost reduces since more paths in every layer are considered to route the traffic demands. This, perhaps, can be explained as follows: the larger the value of Z, the more the number of paths that will be considered to route a given traffic demand at every layer. The fact that more routing options become available in the lower layers where the interface cost is lesser as compared to the higher layers, reduces the overall cost of the network.

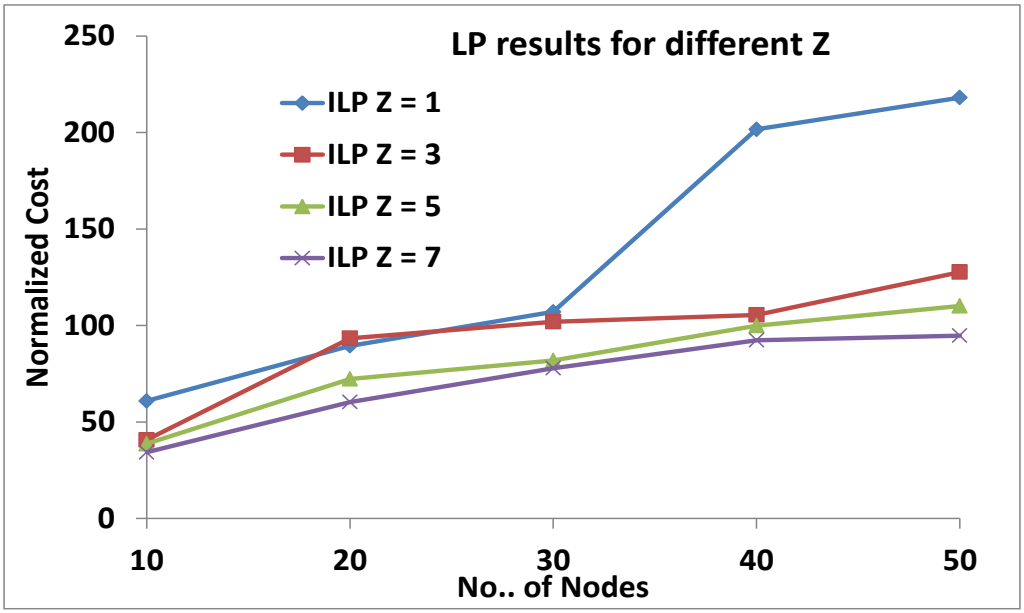


Fig. 3.7. ILP output till 50-nodes, $Z = \{1, 3, 5, 7\}$.

We have also generated results to demonstrate the performance of the heuristic in larger networks (up-to 1000-nodes) with dynamic traffic demands. The graph in Fig. 3.8 compares the performance of the heuristic in only-*IP* and *IP+OTN+DWDM* networks. The size of the network varies from 100-nodes to 1000-nodes. We have compared the performance of the heuristic for different values of *Z*. Fig. 3.8 shows the performance of the heuristic in a network where Z = {1, 3, 5, 7, 9}. The results demonstrate a significant improvement in the total cost of the interfaces which need to be installed in *IP+OTN+DWDM* networks as compared to only-*IP* networks. For example, in networks with 1000 nodes and $Z = 1$, the total cost for only-*IP* networks is 50008.71 units, whereas, the total cost for *IP+OTN+DWDM* networks is 10048.31 units only—an improvement of 79.9%—whereas, the improvement with 100-nodes is 82.01%, which is quite significant.

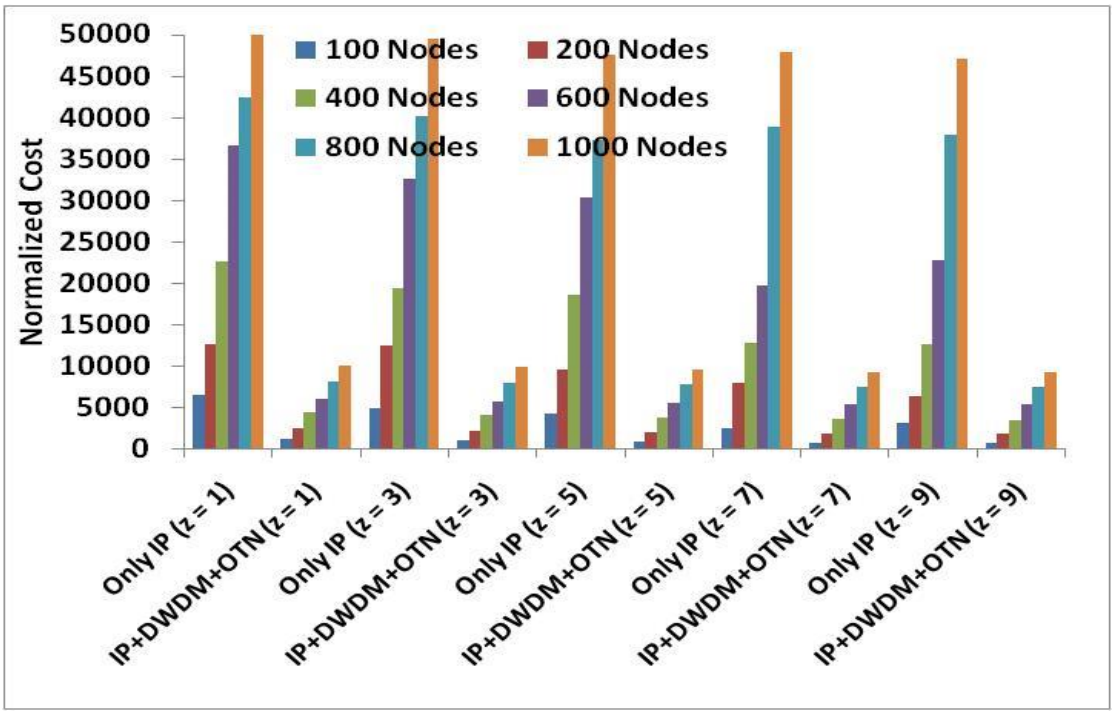


Fig. 3.8. Heuristic Performance in 100-nodes to 1000-nodes Topology with $Z = \{1, 3, 5, 7, 9\}$.

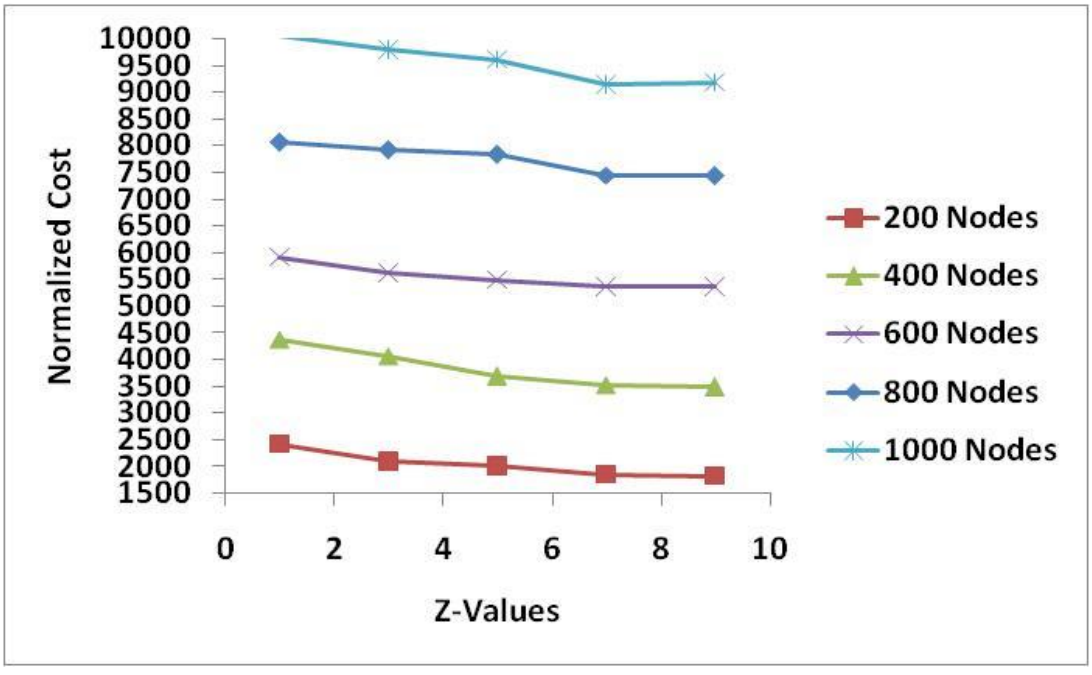


Fig. 3.9. Cost variation with $Z = \{1, 3, 5, 7, 9\}$.

Finally we compare the performance of the heuristic in only *IP+OTN+DWDM* networks with different values of *Z* to demonstrate the effect of *Z* on the total cost of the network. From the graphs shown in Fig. 3.9 we can further conclude that the total cost of the network keeps on decreasing as the value of *Z* increases. For example, the total cost of the interfaces for 1000-nodes in *IP+OTN+DWDM* networks with $Z = 1$ is 10048.31 units while the cost keeps reducing to 9796.21 units, 9610.41 units and 9153.02 units for the same network with $Z = 3$, 5, and 7, respectively. However, we can observe that as the value of *Z* changes from 7 to 9, the total cost remains almost unchanged. This is because, after a certain value of *Z* (in this case 7) there are no more extra paths to be considered as candidate paths. In other words, all candidate paths are considered to provision a traffic demand.

3.5 Conclusions

In this work we have described a 3-layer (*IP+OTN+DWDM*) network capacity planning problem, especially, from the perspective of the network interface cost optimization. The network interface cost is tightly coupled with the total cost of the network. We demonstrated that network cost reduces significantly in 3-layer networks where OTN acts as an intermediate cross-connect layer between the IP and the DWDM layers. An ILP formulation is presented to solve the problem for static traffic (connection) demands in the network. We aim to minimize the total interface deployment cost in a network where maximum traffic demands are satisfied. In addition to the ILP, we have proposed a heuristic approach to minimize the total interface deployment cost for dynamic traffic demands in large networks (up to 1000-nodes). The heuristic runs within a reasonable computational time. The results obtained by implementing the heuristic algorithm and comparing it with the ILP are showcased. The results show that the heuristic compares favorably with the optimal algorithm in terms of computational times and the size of problems that can be solved without compromising the quality of the solution.

Chapter 4

Transport Technology Choices for Virtual Machines (VMs)

4.1 Introduction

Virtual Machines (VMs) form the central processing entity in data-centers and are crucial to facilitating cloud computing environments. To make cloud computing a reality in service provider domains, the applicability of VMs to metropolitan networks is important. The technology in the metro domain is progressively moving from circuit switched SONET/SDH to packet based Carrier Ethernet. An interesting question that we seek to answer is: how does Carrier Ethernet perform for VM migration in data-center and cloud environments? To this end, we perform an extensive simulation study measuring the performance of VM migration over both flavors of Carrier Ethernet –

namely PBB-TE and MPLS-TP. Our study concludes in the feasibility of Carrier Ethernet as a transport technology in data-centers and clouds.

Cloud computing and data-centers are expected to dominate much of metro transport and strongly affect IT-applications and businesses. A paradigm shift has been happening from local computation to cloud computing. One technology that has empowered this paradigm shift is virtualization. Virtualization brings with it capabilities of isolating, consolidating and migrating workload. With the ability to move virtual machine between hosts, live migration has been a core feature of virtualization. Virtual Machine (VM) migration has been fundamental to making data-centers and clouds a reality from the point of the scalability and availability of applications.

The underlying network that connects the disparate servers and storage devices to the rest of the network plays an important role in facilitating migration of content (of the VMs) across servers and enables ease of connectivity for the end-user. The underlying network fabric that binds disparate servers, storage entities and other IT appliances has a stronger role to play when we move from a single data-center (DC) to a collection of DCs—such as in a cloud environment. Traditionally, SONET/SDH systems and more recently IP/MPLS has served as an effective interconnection fabric. SONET/SDH is primarily plagued with cost and TDM-related rigidity in bandwidth provisioning, while IP/MPLS does not manifest the carrier-class features that are necessary for transport and is also seen as an expensive technology for cloud/DC domains. Deploying the recently standardized Carrier Ethernet technology seems an attractive option from both, cost and performance perspective. Two flavors of Carrier Ethernet are available today—the MPLS-TP (as in RFC 5317) [63] and the Provider Backbone Bridged-Traffic Engineering (PBB-TE) (as in IEEE 802.1Qay) [62].

In this chapter, we propose the use of Carrier Ethernet as a transport technology specific to DCs and cloud environments. We describe the specific needs of VM Migration in DC and clouds, and the requirements from the transport network. Then we discuss implementation of Carrier Ethernet in a DC/cloud environment. A detailed simulation model to quantify the benefits of Carrier Ethernet (both flavors) is presented.

4.2 VM Migration and Technology Choices

4.2.1 Virtual Machines (VMs) and VM Migration

Nowadays, service providers agree to very stringent service agreements that have penalties associated with the service downtimes encountered if the physical machine has to undergo maintenance. This mandates high availability of the applications in the data-centers since Virtual Machines (VMs) form the central processing entity in data-centers. High-availability is a key to facilitating cloud computing environments. Live VM migration is a technology innovation that enables an entire running VM to be moved from one server to another without causing service interruption. Live migration brings across several advantages to data center management. VMs can be dynamically migrated depending on workload, offering more efficient usage of computing resources. It allows VMs to be consolidated on a few physical machines when workload is low, allowing operators to power off servers, leading to significant power savings. Before machines are brought down for maintenance, administrators can relocate VMs to other servers without any noticeable interruption of service for users, thus making it very easy to maintain the hardware without any service interruption. Thus, live migration achieves the goal of high-availability of critical applications in data-centers. We have considered VMware Hypervisor Architecture [61, 67] for our simulation purpose as shown below.

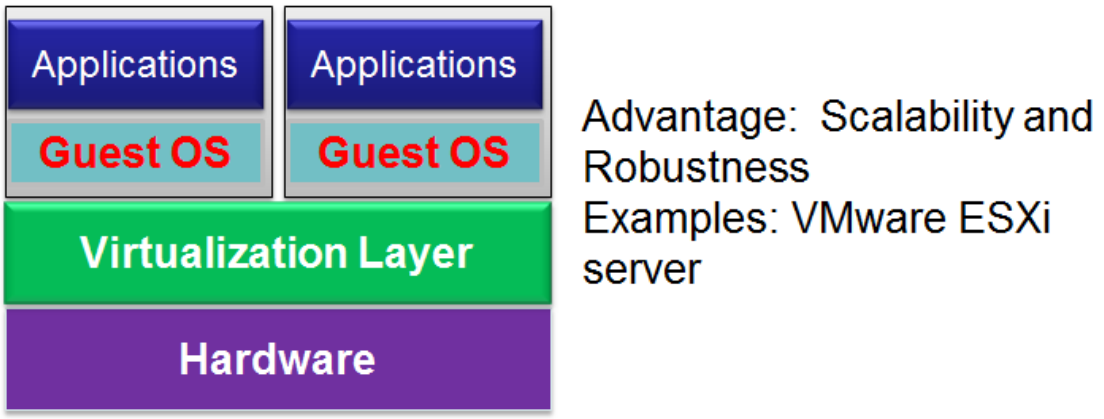


Fig. 4.1. VMware Hypervisor Architecture.

The different VM Live Migration techniques have trade-offs across two parameters—total VM migration time and service downtime. *VM migration* time is the duration between the time migration was initiated and the time when the original VM may finally be discarded on the source machine. Service downtime is the period during which service is unavailable due to there being no currently executing instance of the VM. This period is directly visible to clients of the VM as service interruption. There are various approaches for VM Live Migration. We will refer to a standard *Pre-copy Migration* approach [59-61]. Pre-copy Migration involves six stages. Initially, the VM to be migrated is selected. This is called the initialization stage. The reservation stage is next. In this stage, the resources for the selected VM are reserved at the destination host. This is followed by the iterative pre-copy stage. The memory pages are transferred iteratively to the destination host. During the first iteration, all of the memory allocated to the VM is transferred to the destination host. From the second iteration onwards, only the pages modified during the previous iteration are transferred. These pages are known as dirty pages. When the number of dirty pages in the previous iteration goes below a threshold, the VM is halted for a final transfer round.

This final iteration is called the stop-and-copy phase. After this, the destination host acknowledges to the source host that it has received a consistent copy of the VM. This stage is known as the commitment stage. The destination then attaches the resources to the VM and activates the VM. This is known as the activation stage. On receiving the acknowledgement, the source releases the resources which were assigned to the VM. The iterative pre-copy stage may continue indefinitely if the maximum numbers of iterations are not set because it is not ensured that the dirty pages of all applications will converge to a small writable working set over multiple rounds. Thus, the definition of stop conditions is critical in terminating this stage in a timely manner [60, 65]. The total VM migration time in the pre-copy approach refers to the total time taken by the six stages together, whereas, the service downtime is the time taken by the stop-and-copy phase. Given the criticality of IT-applications and a mechanism to move VMs between machines, the network fabric now assumes gigantic proportions in terms of its importance. The critical decision that we need to make is to determine which technology solution would best fit the DC/cloud environment. Our goal, specifically, is

to evaluate PBB-TE and MPLS-TP versions of Carrier Ethernet and understand the quantitative and qualitative differences.

4.2.2 Carrier Ethernet Technology Choices

Carrier Ethernet manifests itself in two flavors—the Provider Backbone Bridged Traffic Engineering (PBB-TE) [2] and the MPLS-Transport Profile (MPLS-TP) [3]. In the PBB-TE flavor, a customer payload is mapped onto a service provider frame using the concept of MAC bridges. Multiple VLAN tags are used to segregate customer VLANs from provider VLANs. A customer service is identified by a service VLAN identifier called the ISID. Spanning tree protocol and MAC learning are disabled in the provider domain, facilitating convergence through a control plane and forwarding based on assigned core identifiers. These core network identifiers are used by the forwarding plane at each core switch to forward a frame. The core identifiers include a 12-bit BVID and a 48-bit BMAC which, when used together, form a globally unique, 60-bit forwarding address. In the MPLS-TP manifestation, the rich data-centric features of MPLS, such as MPLS label merging, are turned off; there is no support for equal cost multiple paths and the label distribution is done by a well-defined control plane mechanism. To cater to service-oriented protection, each LSP is defined with a backup LSP that essentially provides end-to-end protection (with no support for MPLS fast reroute). Both flavors of Carrier Ethernet use a common control plane mechanism—as defined in the ITU.T Y.1731 and the IEEE 802.1ag as well as the corresponding RFCs in the IETF.

The rich features of Carrier Ethernet can be used as an efficient transport mechanism in DC/Cloud environments. A key challenge is to utilize the control plane as an interface between the VM migration layer and the transport layer. The architecture of such an interface could be provided through proprietary APIs or through Open Flow [63] type APIs, which is beyond the scope of this article. Assuming the existence of such an API between the VM migration layer and the transport layer control plane, it is of interest to us to quantify the performance of both PBB-TE and MPLS-TP. To this end, a detailed simulation model is developed for both intra data-centers VM Migration as well as for VM migration across the network, as in a cloud set-up.

4.3 Simulation and Results for VM Migration

This Section describes the simulation model and results obtained with the given simulation model for VM migration over Carrier Ethernet.

4.3.1 VM Traffic and Simulation Model

In this section, we propose a simulation model for VM migration across the servers in a cloud. As shown in the Fig. 4.2, two data-centers are connected by a set of core routers which operate on either PBB-TE or MPLS-TP. There are eight servers in the cloud. In the interest of simplification of the system, we assume that the physical server environment is homogeneous. It is assumed that each sever has a memory of 8*1024 MBs. Also, it is assumed that the storage is SAN or NAS [64]. Hence, we need to migrate only the memory and the CPU state of that particular VM.

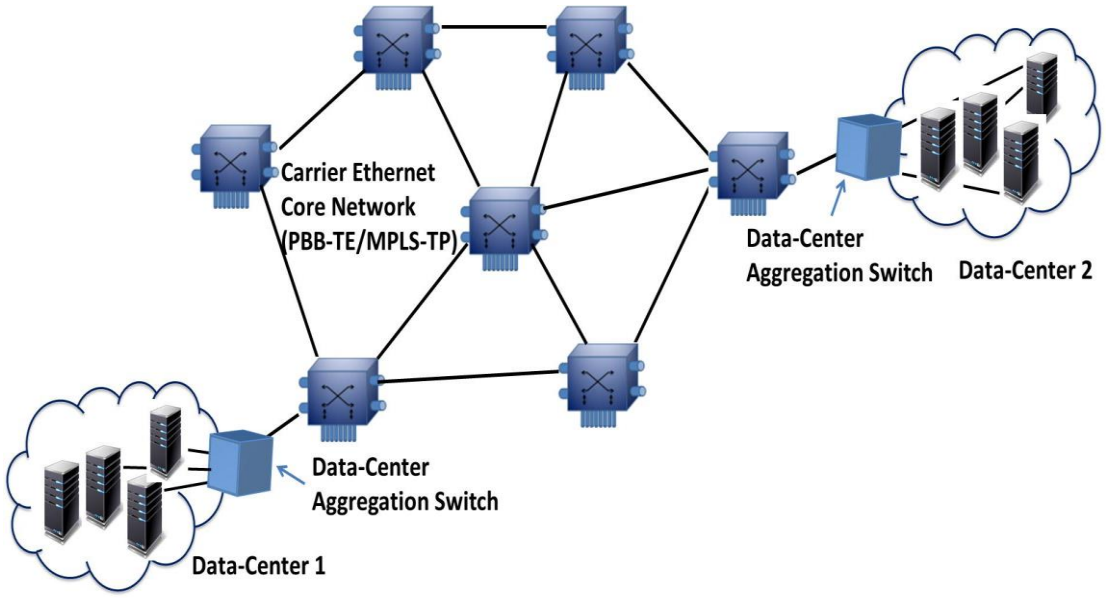


Fig. 4.2: Topology diagram for cloud setup.

In the simulation model, each VM migration request has been mapped to a service request in the Carrier Ethernet domain. This allows us to associate Service Granularity with VM migration requests. Hence, the effect of different service granularities on the

VM migration time and VM Downtime can be observed. In this work we have used the words Service Granularity and Network Traffic (or only traffic) interchangeably.

We have simulated the cloud model using discrete event simulation. Our focus is inter-DC VM migration. All the traffic in the network originates and terminates at the servers. The packet arrival at a node is characterized by a Poisson distribution and packet sizes are exponentially distributed. Each server has a list of VMs associated with it. A VM can be migrated from one server to another, either within a data-center or in between two data-centers. The VM memory size and VM *Page Dirty Rate* (PDR) are critical parameters affecting the VM migration. PDR is the rate at which memory pages in the VM are modified. This depends on the applications running on the VM at that instance. PDR in turn, directly affects the number of pages to be transferred in the next iteration of the pre-copy stage. A higher value of PDR results in more data being sent per iteration. Along with the memory and CPU of a VM, the service granularity (or the network load) and the transport technology being used also play an important role [63, 64]. The simulation model takes the number of nodes in the network and the network graph as an input. It also takes as input the protocol stack to be used for the underlying network, that is, whether VMs are being migrated over PBB-TE or MPLS-TP network. Also, details of the VM (memory size of the VM and PDR of the VM) are provided as input to the simulator. We have simulated the migration of the virtual machine from one physical machine to another using the Pre-Copy approach. The iterative pre-copy step is stopped when one of the following conditions is reached: either the number of pages dirtied in the previous iteration of pre-copy are less than 50, or 29 iterations have been carried out [60].

The model also takes into consideration the non-VM migration traffic flowing in the network. VM migration traffic flow in the network is specified by Service Granularity. We ensure that the network bandwidth is not clogged by VM migration traffic. Also, it has been ensured that no service suffers starvation in terms of network resources. We are given the link capacity and the VM characteristics, that is, the PDR of the VM and the memory size of the VM. The page size is constant at 4KB. VM memory is transferred in terms of pages. Each page is transferred in terms of packets. The

simulation records the time taken for a VM to be migrated as well as the service downtime. The Implementation details and Experimental results for VM migration in a cloud environment are summarized in the sections below.

4.3.2 Implementation Details

We are given the link speed, the VM characteristics, that is, the page dirty rate of the VM and the memory size of the VM. The page size is constant at 4KB. VM memory is transferred in terms of pages. Each page is transferred in terms of packets. The stop conditions used are as discussed above.

Pseudo Code for Pre-Copy Approach

Given: The link speed, the page dirty rate (VM_{pdr}), the memory size of the VM (VM_{mem}), Page size(pg_sz).

Start Simulation.
MAX_ITERATIONS = 29
MIN_PAGES = 50
_sent = 0
_iterations = 0;
_migration_time = 0;
_downtime = 0;
num_pg_transmit = VM_{mem} / pg_sz ;
while (num_pg_transmit < MIN_PAGES and _iterations < MAX_ITERATION)
_iteration_start_time = global_clock.
send num_pg_transmit.
_iteration_end_time = global_clock(when the last packet is received at the destination)
_iterations = _iterations + 1;
_iteration_time = _iteration_end_time - _iteration_start_time.
_migration_time = _migration_time + _iteration_time
*num_pg_transmit = VM_{pdr} * _iteration_time.*
end while
_iteration_start_time = global_clock.
send num_pg_transmit.
_iteration_end_time = global_clock(when the last packet is received at the destination)
_iteration_time = _iteration_end_time - _iteration_start_time.
_migration_time = _migration_time + _iteration_time
_downtime = _downtime + _iteration_time.
End Simulation

The simulation records the time taken for a VM to be migrated and the service downtime due to unavailability of the VM.

4.3.3 Simulation Results

VM Migration time and VM Downtime are functions of VM Memory size (server RAM allocated to VM) and *Page Dirty Rate* (PDR). As the memory size of the VM increases, the time to migrate the VM and the VM Downtime also increases. This is demonstrated with the help of the graphs displayed below. In Fig. 4.3, we have compared the time taken to migrate a VM with different memory sizes against the PDR when the traffic, which is allowed to flow through the network, is constant at 90%. We have considered VMs with memory sizes of 256Mb, 512Mb and 1024Mb. From Fig. 4.3 it is observed that VM migration time increases with an increase in PDR and reaches a constant value as the PDR increases beyond a certain threshold.

This can be explained as follows: as the PDR increases, the number of pages to be transferred in each iteration also increases; this in turn increases the time taken to migrate the complete VM. However, as the PDR increases beyond a threshold value, the number of pages to be transferred in each iteration of the iterative pre-copy stage of pre-copy approach does not converge to less than 50; therefore, the first stop condition is not met. This triggers the second stop condition of the pre-copy approach, that is, 29 iterations of the VM migration have been completed. Thus, the stop condition forces the migration to enter the stop-and-copy phase. This behavior is observed in both PBB-TE and MPLS-TP networks.

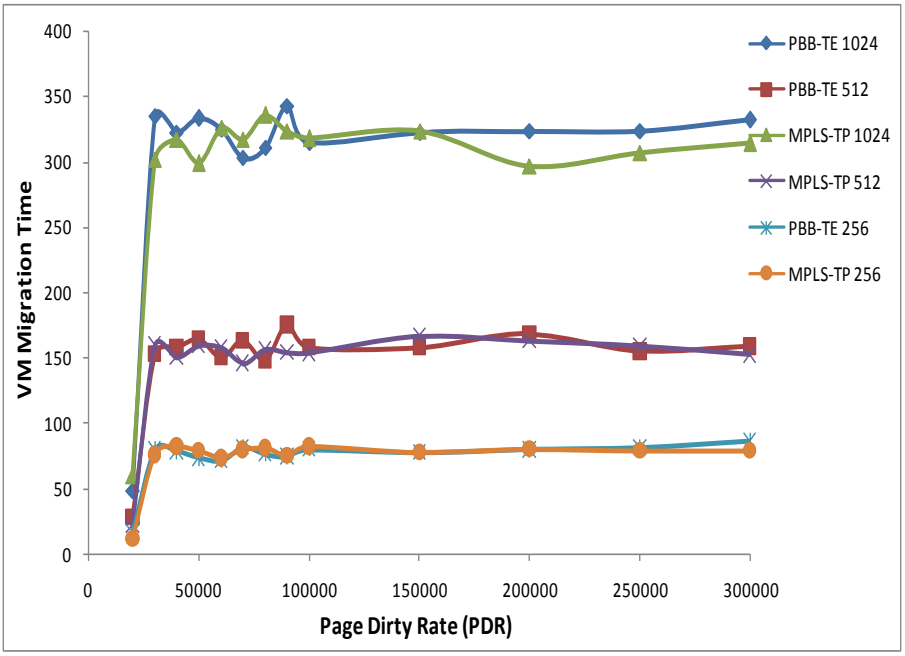


Fig. 4.3: VM migration time versus PDR at constant load of 90%

As the memory size doubles, that is, as memory size increases from 256Mb to 512Mb, and then from 512Mb to 1024Mb, the VM migration time increases by 100%. This is because the entire memory needs to be transferred in the first iteration, and thereafter, only the pages dirtied in previous iteration are to be transferred. Thus, the memory size of the VM directly determines the VM migration traffic flowing in the network. This graph shows that the variation in time required to migrate a VM, is almost the same for both PBB-TE and MPLS-TP, provided that the memory size of the VM and the traffic load of the network is similar.

Fig. 4.4 shows the VM migration time against PDR at the different traffic loads that are allowed to flow through the network with a fixed VM Memory size of 1024Mb. It can be inferred from the graph that the VM migration time increases as the traffic load decreases. In Fig. 4.5, we have compared service downtime of a VM when migration takes place with different memory sizes against the PDR when the traffic load of the network is constant at 90%.

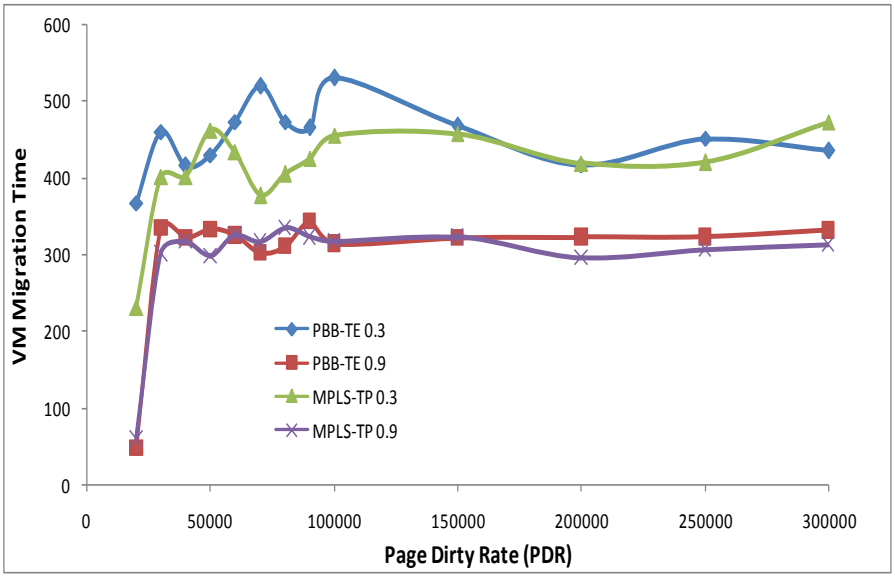


Fig. 4.4: VM migration time versus PDR at a constant memory size of 1024M

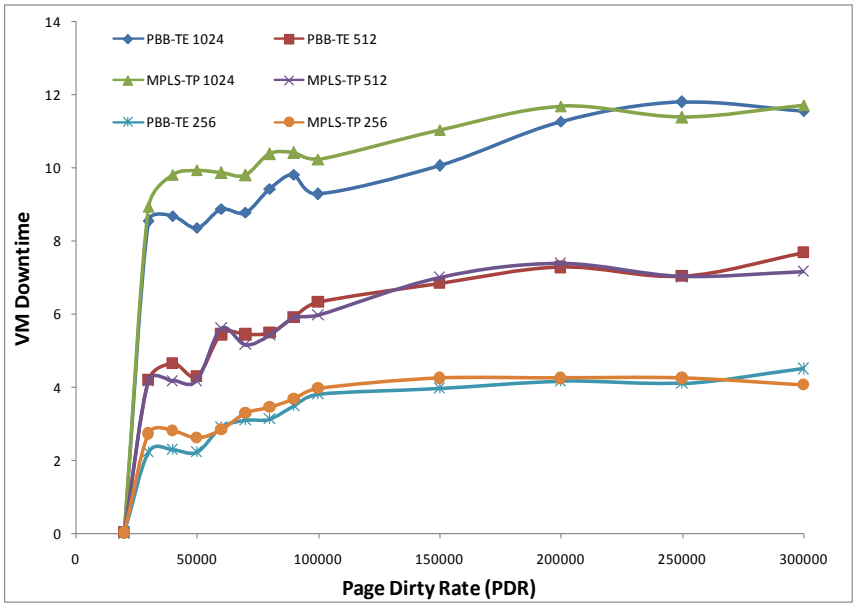


Fig. 4.5: VM Downtime versus PDR at constant load of 90%

We have considered VMs with memory sizes of 256Mb, 512Mb and 1024Mb. The graphs in Fig. 4.4 show the results for both, PBB-TE and MPLS-TP. As the memory size is doubled, that is, from 256Mb to 512Mb and 512Mb to 1024Mb, the VM downtime also doubles. This can be attributed to the fact that at higher memory sizes,

the number of pages that need to be transferred in the stop-and-copy phase would also be high.

The variations in the VM downtime at lower PDR can be best explained as follows: at lower values of PDR, the iterative pre-copy stage terminates with the first stop condition, that is, when less than 50 dirty pages are remaining to be transferred. The number of pages to be transferred in the stop-and-copy phase, and in turn, the service downtime depends on the number of pages dirtied in the last iteration of the iterative pre-copy stage. There can be two cases in this scenario. In the first case, the remaining dirty pages to be transferred are slightly less than 50. The iterative pre-copy stage terminates when the first condition is triggered. Since the remaining number of dirty pages to be transferred is significantly large (close to 50), additional time is taken by the stop-and-copy phase, hence, the VM downtime is also large, that is, in the range of 14-16 seconds.

In contrast, if the remaining dirty pages to be transferred are slightly greater than 50, it will result in one more iteration before the pre-copy algorithm enters the stop-and-copy phase, resulting in a lesser number of remaining dirty pages to be transferred when compared to the first case. So, the time taken by the stop-and-copy phase is less; the VM downtime, therefore, is also less—in the range of 8-10 seconds. Fig. 4.6 shows the VM migration time as a function of a load of the network, that is, traffic allowed to flow through the network, for a VM of 1024Mb memory size at PDR of 8K and 9K. If we consider VM migration as a service in the network, traffic load is a parameter against which VM migration time can be measured.

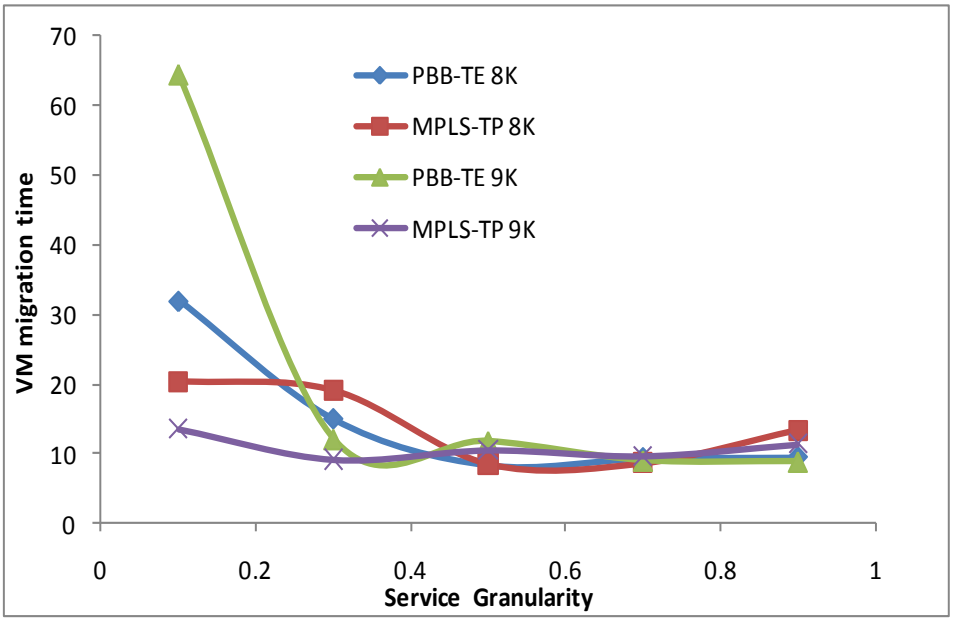


Fig. 4.6: VM migration Time versus Traffic load for a constant memory size.

As the traffic flowing through the network (traffic load) increases, the VM migration time decreases. VM migration time plummets as the granularity crosses a certain threshold value. From Fig. 4.6, it can be seen that the threshold is around 30% traffic load. A low traffic load implies that less service flow is allowed to pass through the network, resulting in more time to migrate the VM.

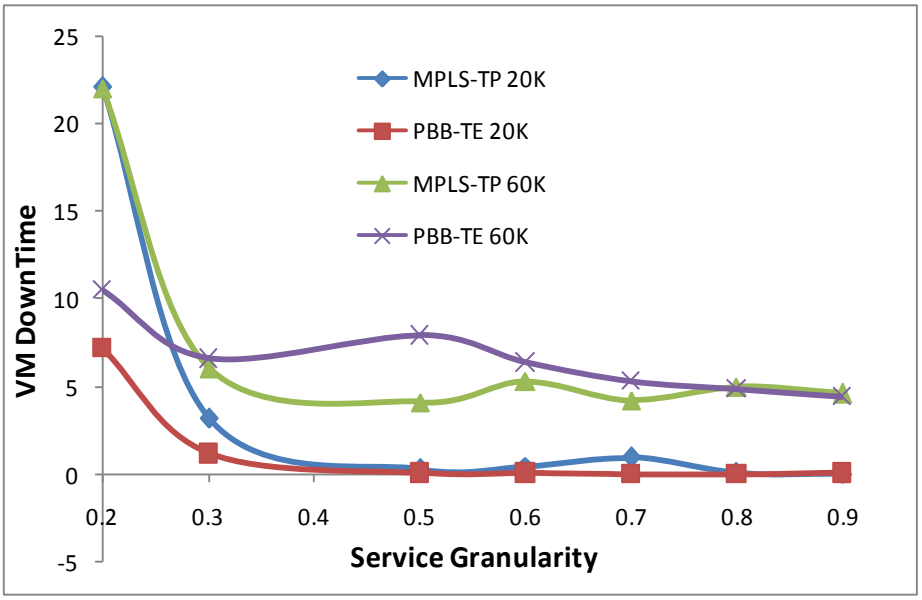


Fig. 4.7: VM Downtime versus Traffic load for a constant memory size 512M.

Fig. 4. 7 shows the VM Downtime as a function of traffic load for a VM of 512Mb memory size at PDR of 20K and 60K. As the traffic load increases, the VM Downtime also decreases. Similar to the VM migration time, VM Downtime plummets as the load crosses a certain threshold value. From Fig. 4.7 we can infer that the threshold is around 30% traffic load.

4.4 Conclusions

From the results and observations in the previous section, the following facts regarding VM Migration can be deduced. As the memory size of the VM (RAM) increases, the time taken to migrate the VM and the VM Downtime increases. With the increase in PDR, both, VM Migration time and VM downtime increase. The VM Migration time reaches a constant value after a certain threshold, and this threshold value is around 100K PDR. This is true for both transport network technologies, that is, MPLS-TP and PBB-TE. VM downtime variation is also approximately the same for both MPLS-TP and PBB-TE.

We have proposed the use of Carrier Ethernet in the data-center/cloud environments. The functional aspects have been discussed and performance issues have been measured through a rigorous simulations model.

Chapter 5

Models, Algorithms and Solution Methods for Centralized Control Planes to Optimize Control Traffic Overhead

5.1 Introduction

Carrier Ethernet (CE) networks have received significant attention in the standards, in both, the IEEE and the ITU. Although standards like IEEE 802.1ah (Provider Backbone Bridges) or PBB [5-7], ITU Multi-Protocol Label Switching (MPLS) [4, 32] are well-established in the industry, the networks built on these standards are unmanageable due to the complex mechanism that arise from the perspective of traffic engineering and network management. The problem lies in the inherent nature of existing technologies which rely on the distributed control plane managed by the routing elements in the network along with routing and data-forwarding [75, 77-80]. Hence, there is a need to

decouple the control and the management plane in the CE networks from the forwarding and routing plane. A new networking paradigm, Centralized Control Plane, has been recently proposed to overcome the aforementioned problem in CE networks [72, 80, 104]. Centralized Control Plane architecture enables the direct management of computer networks with a single controller domain [81, 103]. The centralized software entity (termed as "*controller*") maintains communication channels with the networking devices. It also fetches device/link-level events to create a global state of the network, such as network topology, link utilizations, failure conditions and many others. Updated information regarding the global state of the network is essential for the efficiency of complex control and management tasks such as traffic engineering, resource reservations, failure detection and recovery as well as policy-based service provisioning [76-78, 103-104].

Circuit technology, which formed the backbone of core networks, is being actively replaced with packet-optical integration to mitigate the additional bandwidth provisioning costs [86]. One approach is to map network layer-3 IP packets into Carrier Ethernet frames and then provision over the *optical transport network* (OTN) [36]. *Carrier Ethernet* (CE) networks have received significant attention in the standards in both the IEEE and the ITU. IEEE Standard 802.1ah (*Provider Backbone Bridges)* or PBB [2, 5, 66], ITU standard *Multi-Protocol Label Switching* (MPLS) [3, 4, 96-97] and specific control mechanisms such as IEEE 802.1ag , the *Connectivity Fault Management* (CFM) standard [66] and the ITU Y.1731 are well accepted [65]. In this chapter, we present an approach that relies on improving Carrier Ethernet control components. We try to decouple the control component from the switching element and make it centralized using a *Network Management System* (NMS). Design approaches for a centralized NMS are presented in this chapter—the architecture and implementation aspects of which have been taken into consideration. In particular, the challenges and design approaches in the implementation of an NMS for Carrier Ethernet (CE) networks are presented. Central to the design of such an NMS is the *Carrier Ethernet Switch-Router* (CESR) as a network element which performs forwarding and routing of data-packets across the network. The architecture of a *Carrier Ethernet Switch-Router* (CESR) is proposed in [74]. In addition to this, the control traffic overhead in the

network is analyzed in this chapter, using a simulation model. Simulation results demonstrate that control traffic grows beyond a threshold once the number of nodes and services provisioned in the network exceed a certain number. A high volume of control traffic in the network is not desirable as it might congest the network and cause delays to the data flows which carry the actual information in the network. Also, more traffic means more consumption of energy and network resources. Hence, a scheme to divide a larger network into smaller sub-networks is proposed so that the total control traffic is always below a certain threshold (some percentage of data-traffic). The results from a simulation model are also showcased. The results demonstrate that a good partitioning of the managed networks significantly reduces the control traffic overhead in managed networks.

We present an *Integer Linear Program* (ILP) for the placement of controllers in the partitioned network so as to minimize: (1) the total control traffic in the network, (2) the total controller implementation cost and (3) the overall response time in the network. This is akin to a location-allocation optimization problem in networks. In this chapter we present the problem as a "*Capacitated Single Allocation Hub Location Problem*" (CSAHLP) which has been previously presented in [89-91]. We also present and ILP of this problem. Since the ILP solves only smaller data-instances due to a large number of variables and its computational complexity, a heuristic approach is also presented for solving the problem for larger and real-time service-provider networks.

5.2 Related Work

The centralized software entity (termed as "controller") maintains communication channels with networking devices. It also fetches device/link-level status such as network topology, link utilizations, failure conditions and many others, to create a global state of the network. Updated information regarding the global state of the network is essential for the efficiency of complex control and management tasks such as traffic engineering, resource reservations, failure detection and recovery as well as policy-based service provisioning. Numerous network architectures have been proposed

in literature to design a centralized control plane [see for example, 55, 58, 60-63]. Two types of approaches have been suggested to make CE networks more manageable: 1) Approaches that rely only on improving Ethernet control components and 2) Approaches that rely on improving both Ethernet forwarding and control components [72]. The work presented in [58, 60-61] focuses on the approach of using a centralized control plane, while the work presented in [62, 63] seeks to modularize the control plane architecture and the functionality of the individual router. The work in [64, 67] demonstrates the concept of *Mobile Agents* and distribution of the decoupled control plane for a flexible implementation of the centralized control plane. The work presented in [58] proposes keeping communication channels for management information physically isolated from the paths used by the user data. However, a logical separation between the two is recommended in [60, 61].

However, the literature mentioned earlier does not focus on the implementation aspects of the *Network Management System* (NMS) from a service providers' perspective. In addition to this, there is a dearth of simulation results to compare the performance of different approaches in terms of control traffic overhead and response time. In this work, we present an approach for implementing a centralized control plane. We have considered an approach implemented in the transport layer of contemporary SONET/WDH networks [71, 77-78], which means that networks are completely managed and traffic is engineered by the control plane implemented by the NMS. A simulation study demonstrates the performance of the NMS in terms of the control traffic overhead. More control traffic volume in the networks is undesirable as it is an overhead traffic and does not carry any actual data information. Hence, it is desirable that the total control traffic is kept below some threshold [100-102]. In our work, a scheme to partition the network into sub-networks is proposed so as to minimize the total control traffic in managed networks. In addition, an ILP for an optimal placement of the controllers is presented. We have extended the work presented by Ernst and Krishnamoorthy in [72, 74-75] and by Skorin-Kapov et al. in [73] to apply it to the controller placement problem and network partitioning scheme. A heuristic approach is also presented to solve larger data-sets with real-time network topologies.

5.3 Management and Challenges in a Centralized Control Plane

5.3.1 Management in a Centralized Control Plane

In this work, we investigate direct control through the NMS: this means that the control and management system has both, the ability and the responsibility, to set all the states in the data plane that direct packet forwarding [72]. It is of our interest to define and discuss key characteristics and the functionality provided by the NMS. In this pursuit, some of the high level tasks of the centralized NMS are discussed in this section. Fig. 5.1, below, shows the logical view of the centralized control plane implemented by the NMS.

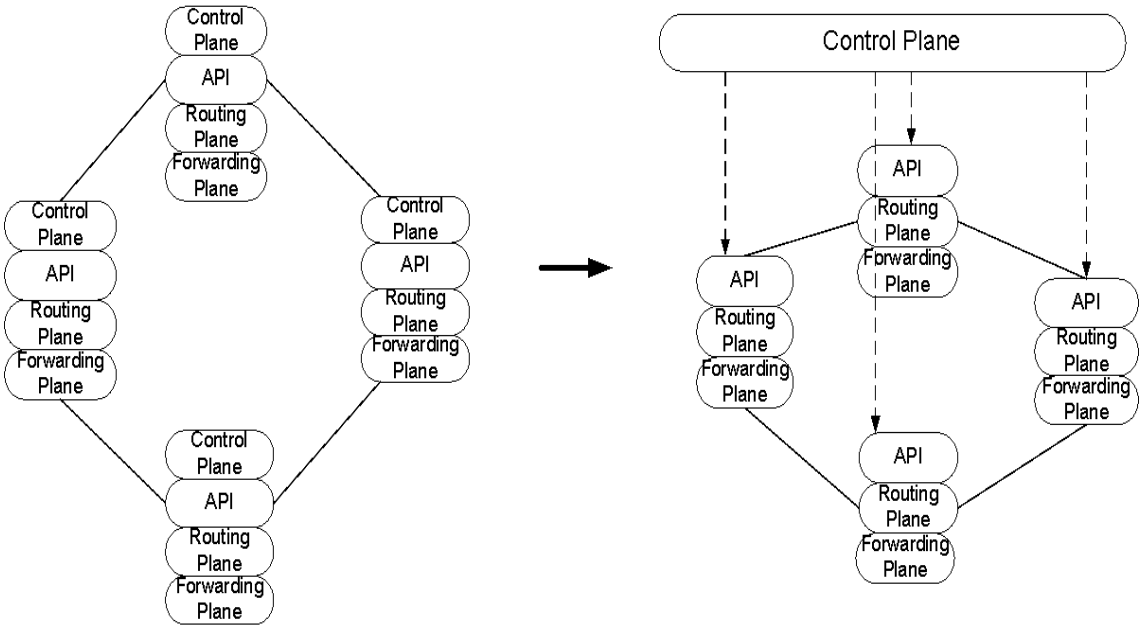


Fig. 5.1. Logical view of the distributed and centralized Control.

5.3.1.1 Topology discovery and Network Configuration:

Timely and accurate knowledge of the underlying network topology is necessary for robust networks. The NMS cannot perform configuration tasks unless it is fully aware of the global state of the network devices and interconnections (or links). So, topology discovery is a primary and crucial step in centralized network management. The NMS provides an interface to aid in creating the network topology and to verify it with the existing physical topology. Once the underlying network topology has been discovered and verified, the NMS becomes aware of the status of each device (parameters like device type, device MAC address, port status and others) and the connectivity between the devices. Only after the initial discovery and configuration, is the operator capable of making changes like adding/removing a new device/link, in the topology.

5.3.1.2 Device Configuration

When a network device, that is, the CESR is powered on, it boots in its default state where no packet-flows are permitted to pass through the network, while the NMS configures all the network devices. Once the NMS is aware of the underlying network topology, it configures the devices according to network-specific parameters. The parameters can either be device-specific, like categorizing the ports as either the edge port or the core port, or they can be network-specific, like the *Backbone Virtual LAN Identifier* (BVID) range allocation for PBB services [2, 5] or a route for a particular service. NMS configures the network devices by populating the related tables in the CESRs.

5.3.1.3 Service Configuration and Decision Plane

Since the control plane is abstracted in the form of the NMS in centralized control plane architecture, the NMS is solely responsible for configuring different services in the underlying network. The NMS provides a mechanism to specify service-specific parameters. For example, if the operator chooses S-tagged or C-tagged frames, Virtual

LAN IDs (VIDs) [6, 7] need to be provided. Then, the end devices and the edge ports are specified, indicating end points between which the services need to be configured. In addition, granularity, traffic priority, rate-limiting and other parameters are also specified. The NMS verifies all these parameters and provisions the requested services. Once a service is configured, the NMS notifies the operator with an appropriate status message regarding the service configuration.

5.3.1.4 Communication with the Hardware

The NMS communicates frequently with the underlying hardware. This communication is necessary to populate the forwarding tables of the CESRs so as to enable packet-flows in the network according to the services defined by the operator. In addition, the network devices exchange information related to network management and administration through this communication channel. A reliable and secure mechanism is necessary to make the communication happen between the NMS and the network devices. An NMS designer may choose to use standard protocols like *Simple Network Management Protocol* (SNMP) or to define proprietary protocols and frame formats to enable accurate information exchange between the NMS and the network devices [75, 77-80].

5.3.1.5 Resource Allocation

An additional important aspect of the NMS is the optimal allocation and utilization of network resources. Since the network resources are often limited (such as different identifiers, link capacities and others), the decision plane algorithms that manage these resources impact the utilization of the network resources acutely. For example, in PBB-TE networks, there is a limitation on the number of *Backbone VLAN Identifiers* (BVIDs) [2, 5]. The number of connections that can be provisioned for a particular destination is impacted by the mechanism of allocating BVIDs to the services [8]. So, it is imperative that the NMS utilizes network resources optimally.

5.3.1.6 Network Monitoring and Fault Management

Monitoring the network for its connectivity and traffic statistics is an essential aspect of performance and resource management planning of the NMS. The NMS needs to provide a mechanism for Network Monitoring and Fault Management. For this purpose, the NMS continuously checks the network topology and the statistics of the traffic profile flowing through the underlying network, such as frame loss, throughput and the like. Different fault conditions, such as link failure or device failure may occur in networks, affecting the already configured services. To meet the requirements of *Operation, Administration, Maintenance and Provisioning* (OAM&P) in managed networks, we seek to implement functionality similar to the one defined in the standards such as CFM [66], using *connectivity check messages* (CCMs).

5.3.2 Challenges in a Centralized Control Plane

Besides these tasks, the NMS is expected to perform forwarding of packets as per the rules defined by the operator, load-balancing, intelligent flow classification and flexible modification of frames, which are not discussed in detail in this work. The centralized control plane architecture transforms the traditional data-networks into more robust and manageable networks [72]. However, this approach has its own challenges, some of which are discussed below:

5.3.2.1 Complexity

Complexity is one of the most crucial aspects of the NMS as the scalability of the entire network depends on the time and space complexity of various NMS algorithms. While the centralized control plane philosophy separates control-plane logic from data-plane logic—providing robustness to the network—its advantages might be hindered if the underlying algorithms are too complex. The time and space complexity of the

algorithms implemented by the NMS should accommodate the growth of the underlying network because the resources available to the host system are limited (resources like computing ability, computing speed, storage ability and the like).

5.3.2.2 Convergence and Scalability

The presence of a central controlling entity provides flexibility in the choice of algorithms and mechanisms for controlling and provisioning of services in the deployed network. However, being centralized, the NMS may have to cope with issues such as scalability. As the number of devices in the network grows, it becomes difficult for the NMS to accommodate the growth of the network. The performance of the NMS is expected to remain stable as the network scales. For example, the NMS is required to handle all the '*hello message'* replies from all the devices in large networks. The NMS may not drop any of these reply-packets, either due to processing limitations or algorithmic complexity. It is desired that the centralized control plane does not remain a limiting factor in complex virtualized environments and others [67, 70]. This means that the NMS should be able to store large lookup tables required in such virtualized environments and populate the network devices in an acceptable time frame.

5.3.2.3 Response Time

As the size of the network increases, the round trip communication time of the control frames sent by the centralized control plane increases. This results in additional latency while communicating network-configuration updates to the network devices. The minimum duration within which the network status and updates are communicated to the network elements and to the user is directly proportional to the span of the network. For example, the maximum delay occurs when the NMS communicates with the farthest device in a topology that has all the nodes connected in series (one after another). To adhere to the carrier-class nature of the underlying network, the response time of the NMS is expected to be within acceptable limits of the service-provider networks.

5.3.2.4 Stability

It is desired that the NMS is in a stable-state for the duration of its execution in diverse networks conditions such as different network topologies with different network traffic (load) scenarios. By stable-state we mean that the performance measures of the NMS (such as CPU, memory utilization, response time) should remain within acceptable limits. The key parameters that may limit the stability of the NMS are the number of network-elements and the traffic load in the network (that is the number of connection requests to be provisioned in the network). For example, as the number of connection requests to be provisioned in the network increases, the size of the NMS data-structure, and the processing time for the algorithms increase. The performance of the NMS depends not only on the size of the network and the service load (number of service requests at a given instance of time) but also on the topology of the network, that is, the distribution of the nodes in a network and the interconnection between these nodes or the links in the network.

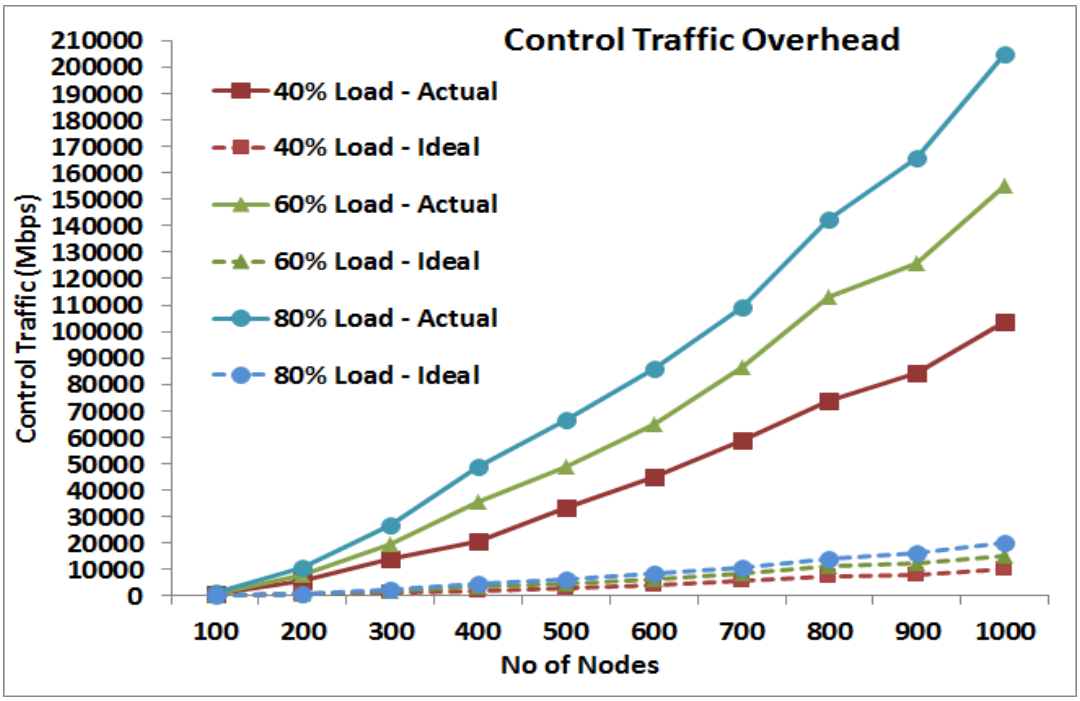


Fig. 5.2. Polynomial growth in control traffic with number of nodes in the network.

The interaction and information exchange between the NMS and network elements generate a significant amount of control traffic in the underlying network. The control traffic includes '*hello messages*' for topology discovery, '*write messages*' for information dissemination, '*connectivity check messages*' for fault detection and recovery and etc. It is desired that the control traffic generated due to the interaction between the centralized control plane and network elements be below a threshold value as it adds to the latency in the data traffic flow. Refer to the graph plotted in Fig. 5.2 displaying the growth in control traffic with the number of nodes in the network varying from 100 to 1000 at different tariff loads (20%, 40% and 80%) using a simulation setup that is explained in Section 5.6. Dotted lines in the graph indicate the maximum amount of control traffic that can be endured in the network so that the actual data traffic is not affected adversely [100, 102]. We observe that, the gap between the expected control traffic and the actual control traffic increases rapidly as the size of the network increases.

In the next section, we present possible approaches to designing the scalable and robust NMS and minimize the control traffic in the managed network. We also explain the motives in selecting a particular design approach.

5.4 Generic Design Approaches and Implementation Specifics for the NMS

An essential step in the implementation of scalable and reliable networks is the design of algorithms for the centralized control plane, and demonstrating their correctness. Improper design of the decision plane algorithms might result in degraded performance or even failure of the entire network. We discuss below, a possible framework for designing the NMS. In this paper we provide alternatives for the design of Interface, Discovery, Decision and Dissemination planes of the NMS. We then provide an optimized approach for the design of the Decision plane of the NMS.

5.4.1 Interface Plane

The design of the interface of the NMS is typically based on the principle of the client-server model, where a user interface acts as a client and the core processing system acts as a server. Hence, the design of the NMS is split into two modules: Client Interface design and Back-end/Core server design. Client Interface is either a stand-alone application or a web-based application.

The simplest approach for client interface design of the NMS is a stand-alone application. This approach has several drawbacks: One of its major drawbacks is that it requires installation of a thick client [71] at every end-user machine; this is an inconvenient option because it requires more memory and processing power at the user-end, compared to a web-based application. In contrast, a web-based application is a thin client and it supports the client-server architecture [71]. In addition, it allows multiple user-access to the NMS from distant locations. It also enables the NMS to scale for the larger networks by dividing it into multiple "Autonomous Systems" (AS) and managing each AS independently [85].

In the web-based approach, the NMS does not need to be online (connected to the network) at all times. Once the network is configured, the data-flow continues even if the NMS is not connected to the network. Service restoration is guaranteed for all the protected paths. Once the NMS is online again, it resumes monitoring the network. This provides additional flexibility with the NMS control algorithms, which can be modified at run-time without affecting the data-flow within the network. Fig. 5.3 below shows the different components in the interface plane of the NMS.

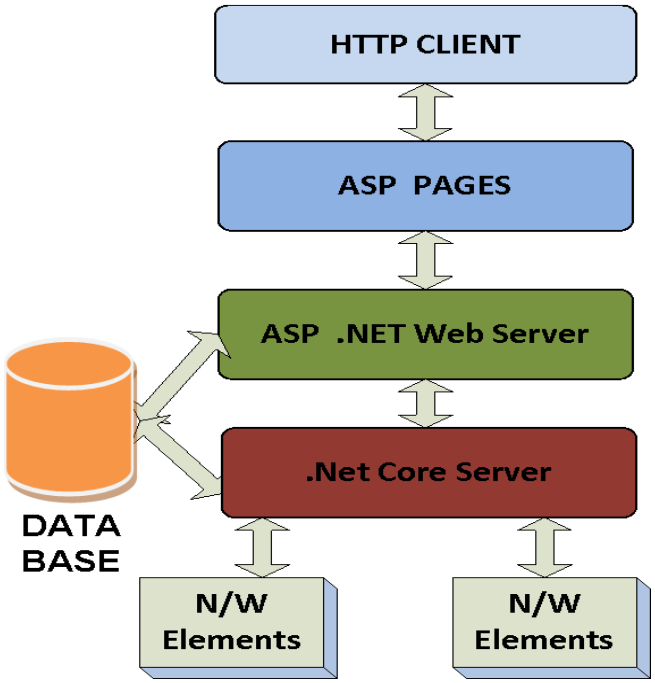


Fig. 5.3. NMS Interface Plane and Components.

5.4.2 Discovery Plane

Network topology discovery may be performed in two modes: a) *Offline* and b) *Online*. In the *Offline* mode, the operator manually configures the network topology and then uploads it to the network. The NMS verifies the user-defined topology against the underlying topology. This is an inefficient and inconvenient option from the perspective of operators from the point of view of scalability. In addition, this approach is not in line with our aim of designing the network-discovery plane of the NMS applicable to multiple network types that requires zero pre-configuration.

On the contrary, in the *Online* discovery mode, the NMS automatically fetches the physical topology once the NMS is connected to a network device and creates a corresponding pictorial view. Techniques for auto-discovery between neighbors have been proposed in optical networks with *Generalized Multiprotocol Label Switching* (GMPLS) [82] and *Link Management Protocol* (LMP) [83]. The *Online* mode reduces the complexity of the NMS and the operator's overheads. A designer can implement standard protocols such as SNMP or may define proprietary protocols for the

communication between the NMS and the network devices such as '*hello messages'* [68, 69], to enable the NMS to fetch the network topology and exchange information with network devices. We have defined and used proprietary frame formats for the communication between the CESRs and the NMS. Thus, the discovery plane aids the NMS in getting acquainted with the physical components in the network and the interconnectivity between them. The discovery plane builds the logical entities of the network-wide physical components which are used by the decision plane of the NMS. A logical view of the discovery plane is shown in the Fig. 5.4 below.

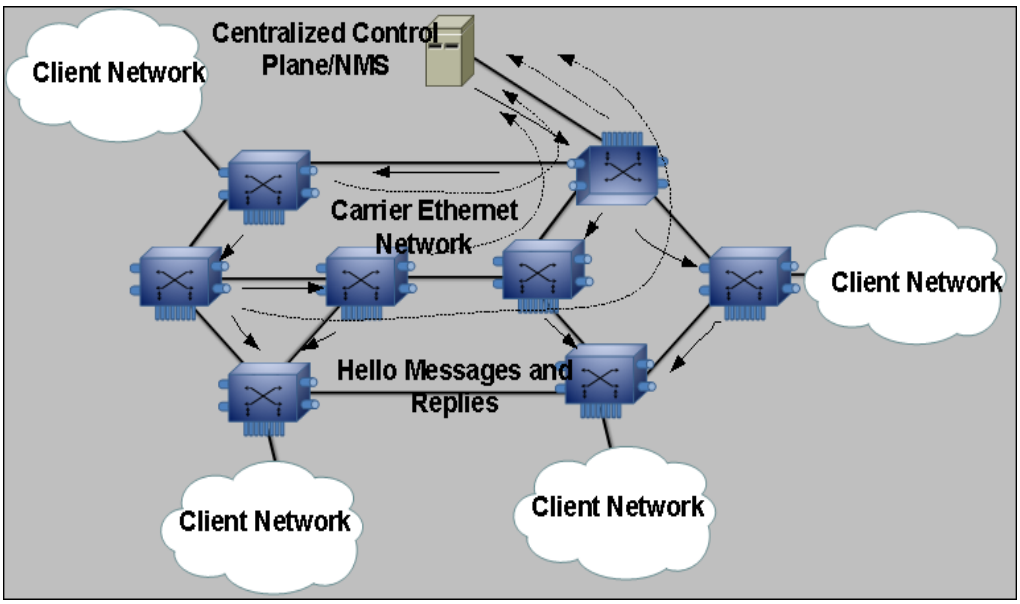


Fig. 5.4. NMS Discovery Plane: Logical View.

5.4.3 Network Information and Decision plane

Managing the information associated with the underlying network (such as the network topology, connectivity, device-specific details, service portfolio, monitoring and alarm information and others) and making decisions based on the information is an important aspect of a centralized control plane. The decision plane makes important decisions that drive service provisioning, network control, load balancing, security, monitoring and the like. Based on the information assembled by the discovery-plane of the NMS, decision-plane algorithms drive the data plane of the underlying network and allow flexible control over packet-flows passing through the network. The decision-plane algorithms

can be implemented in different ways depending on the choice of the NMS designers. For example, while provisioning the services, one may choose to implement pre-computed paths between the edge routers, while others may want to do dynamic path finding on the arrival of a service request [72, 76]. A detailed discussion of the implemented algorithms in our NMS is beyond the scope of this chapter.

The information and data generated regarding the state of the network and the configured services need to be stored in an orderly and secure manner. So, the database and its management is an important aspect of the NMS. The database provides stability to the NMS in case of failure, crash or abrupt shutdown. Database implementation includes, among others, database schema design and selection of the appropriate database tables [73]. A detailed database design discussion is beyond the scope of this article. Frequent database queries are expensive for any software system since these operations consume significant computing resources and memory. So, the NMS stores as much data as possible locally, for faster access. Data Structures are the variables of a process to which the NMS has a quick access. These variables reside in the memory till the NMS is up and running. Once the process stops, these variables are flushed from the (heap) memory. The NMS repopulates these data structures from the database in case they are flushed. For the stability of the NMS, it is important that there is synchronization between data stored in the database and data stored in the data-structures. An abstract view of the decision plane is shown in Fig. 5.5 below.

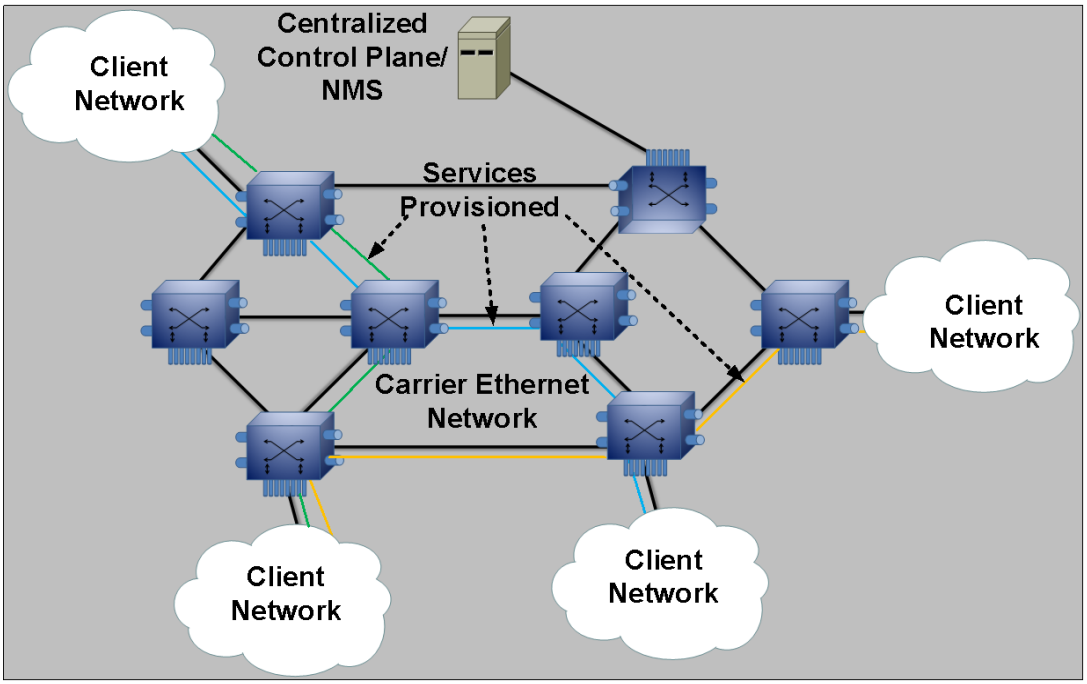


Fig. 5.5. NMS Decision Plane: Abstract View.

5.4.4 Dissemination plane

The dissemination plane connects the underlying network devices with the *Decision Plane* of the NMS. The control plane communicates with the network devices very often to exchange information (for topology discovery, populating forwarding tables to enable data-flow for the configured services, network monitoring, etc.). The NMS starts communicating with the network devices using '*hello messages'* as soon as the NMS boots up. The initial communication is intended for online topology discovery. Apart from the initial communication, the NMS sends various configuration frames to the network devices for service configuration, network monitoring and the like. All the configuration frames are either from the NMS to a specific device in the network, or from a particular device to the NMS—as a reply or acknowledgment of a particular message sent by the NMS. In our design, the standard Ethernet frame format has been implemented for control plane communication with the underlying network-devices. Each frame contains fields such as destination address, source address, *Ether-type* and the data field which includes the *Op-Code* (operation code), the sequence number, the

table number of the device on which processing is required, followed by the actual data to be written in the forwarding tables.

After online topology discovery, the NMS gets acquainted with the interconnectivity between the network devices. The NMS then calculates the shortest path from the *root device* (*root device* is a network device to which the NMS is directly connected) to the destination device, and back to the NMS. Both these routes are embedded in the frame sent by the NMS to populate the forwarding tables of the network devices [68, 69]. Each destination device sends an acknowledgment once it has received a valid configuration frame sent by the NMS. Only the reverse path computed by the NMS is embedded in this frame. In this case, the sequence number is learnt only at the destination device. If the NMS does not receive an acknowledgment from the destination device within a certain amount of time, the NMS resends the same configuration frame with the same sequence number. The following cases may cause this scenario: 1) The frame sent by the NMS is lost in the network and the intended device did not receive the frame. In this case, the duplicate frame sent by the NMS is treated as a new frame by the device and the sequence number is updated accordingly. 2) The network device received the frame sent by the NMS but an acknowledgment sent by the device is lost in the network before reaching the NMS. In this case, the duplicate frame sent by the NMS is discarded and the destined node resends an acknowledgment frame. This process is repeated for a fixed number of times, after which, the user is notified with an appropriate failure notification generated by the Fault Management and Monitoring part of the NMS. Fig. 5.6 above shows the logical view of the dissemination plane of the NMS.

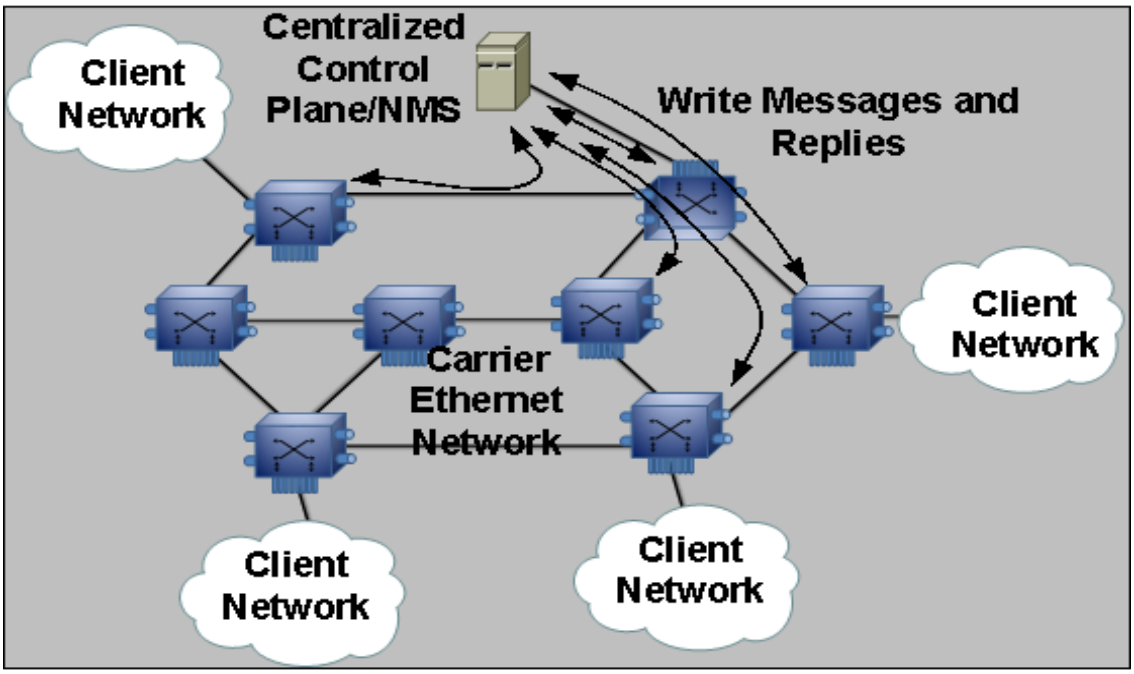


Fig. 5.6. NMS Dissemination Plane: Logical View.

The interaction between the NMS interface and different planes is shown in Fig. 5.7 below. The interaction and information exchange between the NMS and network elements generate a significant amount of control traffic in the underlying network. The control traffic includes *hello messages* for topology discovery, *write messages* for information dissemination, *connectivity check messages* for fault detection and recovery and so on. It is desired that the control traffic generated due to the interaction between the centralized control plane and network elements be below a threshold value as it adds to the latency in the data traffic flow. However, it is observed that the control traffic increases exponentially as the number of nodes in the network increases [76. 97]. In the subsequent sections we analyze the overhead of the control traffic in the network and propose a scheme to reduce the control traffic overhead. We demonstrate that the goal is achieved by splitting the larger networks into smaller sub-networks. The scheme is modeled, formulated and evaluated using an *Integer Linear Program* (ILP) and heuristic approaches are also proposed in subsequent sections.

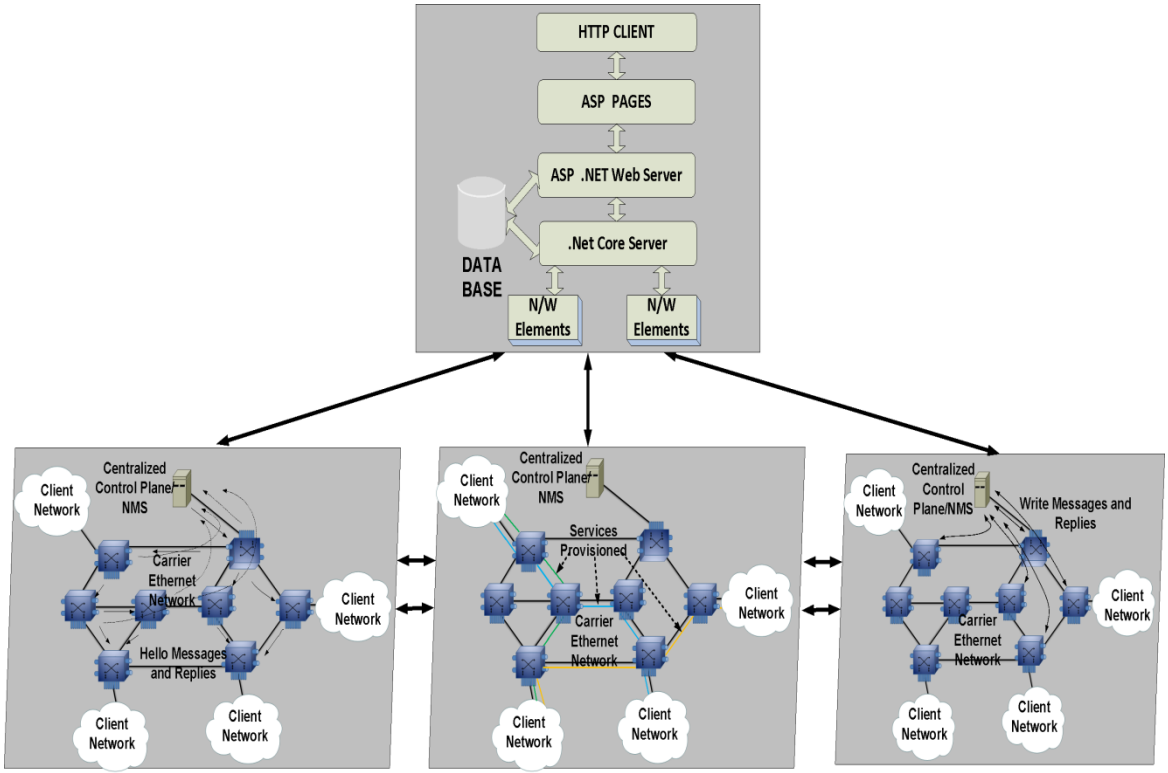


Fig. 5.7. Interaction between different NMS planes.

5.5 Integer Linear Program (ILP) for Controller Location and Allocation

We observe the polynomial form of n^{α} to represent the total control traffic in a sub-network of *n*-nodes, where α is a constant exceeding unity and *n* is total number of nodes managed by a single controller in the network. As a result, the percentage of data traffic drops below the threshold if *n* exceeds a certain number. Clearly, the polynomial form represents "loss-of-scale" [95]. In this section, we aim to divide larger networks into smaller sub-networks such that, each sub-network is managed by a separate controller and the number of nodes managed by each controller is less than the number which has been obtained in the simulation. However, there are certain issues associated with network partitioning, such as response time, controllers' placement, volume of control-traffic generated in each sub-network and the like. Hence, in this section an

Integer Linear Program (ILP) is formulated to optimize locations of the controllers in the network so that the following goals are achieved:

1. Control traffic generated in the network does not grow beyond the threshold. In other words, the number of nodes managed by a single controller is below the pre-defined threshold value.
2. The total control traffic volume in the network is minimized.
3. The response time of the controller of each node is minimized.
4. The total cost for controller installments is also minimized.

The controller backbone (virtual) network may or may not be fully interconnected. Here, a fully interconnected network is assumed and the path chosen is the shortest path between the controllers through the underlying network. Intuitively, the best case solution is to install a controller at every node. In this case, the control traffic in the network will be the least. However, this solution is the most expensive, since each controller has a fixed cost related to its installation. The most cost-effective solution is to have a single controller. However, since there will be a single network, the control traffic will grow beyond a threshold as the network size grows. In the subsequent section we try to find a solution which will achieve a balance between these two extreme solutions. The problem definition is explained in greater depth with the help of Fig. 5.8 below. Fig. 5.8A displays a sample network partitioned into three sub-networks. Each sub-network has a separate centralized controller. Fig. 5.8B displays the logical connection between the nodes and the corresponding controller.

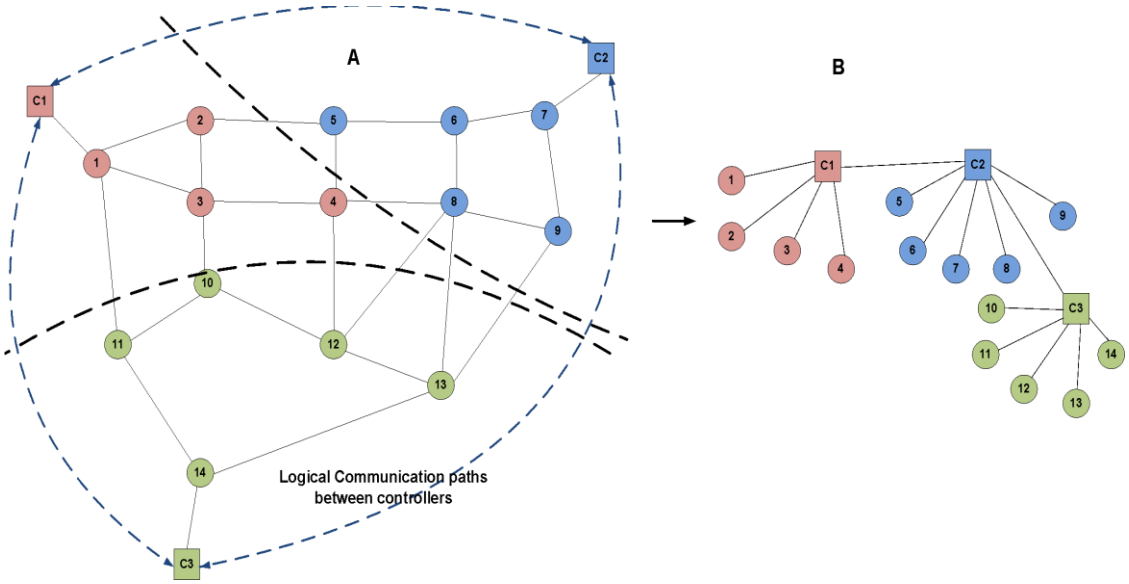


Fig. 5.8A. Sample network and division of the network in three sub-networks with controllers.

Fig. 5.8B. Logical Connection between different nodes and their respective controllers.

To manage the control traffic in centrally controlled networks, we need to understand the properties of the control traffic generated in such networks. The control traffic generated in the managed networks may be divided into two distinct types. The first type, referred to as intra-domain traffic in this article, comprises *Hello Frames*, *Hello replies* and *Management and OAM&P Frames* such as *CCMs*, which are limited to that particular sub-network only. This type of traffic is broadcast or multicast traffic. The second type of traffic is generated due to the services configured between different source-destination pairs in a single sub-network or spread across the sub-networks. It includes *write messages* which are generated to configure the forwarding table entries in the devices and replies sent by the devices as an acknowledgement of the successful configuration of the forwarding tables (*write message replies*). This type of traffic is unicast traffic. The volume of the intra-domain traffic, which is broadcast or multicast, is larger as compared to the service configuration traffic which is point-to-point or unicast. In addition, the frequency of *Hello Frames* (and Replies) and monitoring messages such as *CCMs* is also very high as compared to the *Write frames* (and replies).

Consider a complete graph $G = (N, E)$, where N is the set of n nodes and E is the set of e edges. Let $Z_{ij} \in \{0, 1\}$ be 1 if node i is allocated to a controller located at node j and 0, otherwise. In particular, $Z_{kk} = 1$ implies that node k is selected as a controller [90]. As mentioned in [89] by Skorin-Kapov et al. and in [90, 91] by Ernst and Krishnamoorthy, let X_{ij}^{kl} be defined as the flow from node i to node j via hubs located at nodes k and l. Further, C_{ij}^{kl} is defined as the cost of routing one unit of flow along this path. This cost is given by $C_{ij}^{kl} = \chi.d_{ik} + \alpha.d_{kl} + \delta.d_{lj}$ where χ, α, δ are the costs associated with the traffic sent from a source node to the controller (collection cost), from one controller to another (transfer cost) and from the final controller to the destination node (distribution cost), respectively. It is observed that the problem mentioned in this section is a variant of the *Capacitated Single Allocation Hub Location Problem* (*CSAHLP*) as mentioned in [89-91]. Let F_k be the fixed cost of establishing a hub or a controller at node k. Let $b_{ij} \in \{0, 1\}$ be 1 if nodes i and j have been assigned the same controller and if 0 otherwise. The detailed ILP formulation is given below. The input parameters to the ILP formulation are:

1. N: The maximum number of nodes in the network.
2. d_{ij}: A two-dimensional matrix showing the distance between each node $i, j \in N$. The distance matrix follows the triangular inequality given in the Euclidean space.
3. χ, δ, α: These are the costs associated with the traffic sent from a source node to the controller (*collection*), from the final controller to the destination node (*distribution*) and from one controller to another (*transfer*), respectively.
4. Γ: The maximum number of nodes that can be allocated to a single controller.
5. W_{ij}: The volume of traffic in-between two pairs of nodes in terms of the number of connection (service) requests.
6. $\hat{U}_{ij}$: The volume of intra-domain control traffic between two nodes i and j due to *Hello_Frames* and *CCMs*.
7. $\ddot{U}_{ij}$: The volume of control traffic between any two nodes i and j due to a single service configured between these two nodes.

   Let $\hat{U}_{ij} = \hat{U}$ and $\ddot{U}_{ij} = \ddot{U}, \ \forall\, i, j \in N$.
8. C_{ij}^{kl}: The cost of routing one unit of flow along the path i-k-l-j: $C_{ij}^{kl} = \chi.d_{ik} + \alpha.d_{kl} + \delta.d_{lj}$

9. F_k: The fixed cost of establishing a hub at node k,

   where $i, j, k, l \in N$ are indices for nodes in the topology.

The binary decision variables are:

$$Z_{ij} = \begin{cases} 1, \text{if node } i \text{ is allocated to a controller located at node } j \\ 0, \text{otherwise.} \end{cases} \quad (1)$$

Note that $Z_{kk} = 1$ implies that node k is selected as a controller.

$$X_{ij}^{kl} = \begin{cases} 1, \text{if the flow from node } i \text{ to node } j \text{ is via hubs located at nodes } k \text{ and } l \\ 0, \text{otherwise.} \end{cases} \quad (2)$$

$$b_{ij} = \begin{cases} 1, \text{if nodes } i \text{ and } j \text{ have been assigned the same controller} \\ 0, \text{otherwise.} \end{cases} \quad (3)$$

The objective function is written as:

Minimize:

$$\sum_{i \in N} \sum_{k \in N} \sum_{l \in N} \sum_{j \in N} C_{ij}^{kl} \; X_{ij}^{kl} \; [W_{ij} \ddot{U} + \hat{U} \, b_{ij}] \; + \; \sum_{k \in N} F_k \; Z_{kk} \quad (4)$$

Subject to:

$$\sum_{k \in N} \sum_{l \in N} X_{ij}^{kl} \; = \; 1, \qquad \forall \, i, j \in N \quad (5)$$

Equation 5 ensures that there is only one path from node i to node j via nodes k-l.

$$\sum_{k \in N} Z_{ik} \; = \; 1, \qquad \forall \, i \in N \quad (6)$$

Equation 6 mandates that only one controller be assigned to every single node.

$$\sum_{i \in N} Z_{ik} \leq \Gamma, \quad \forall k \in N \tag{7}$$

Equation 7 guarantees that every controller is assigned nodes below a certain number so that the control traffic in the sub-network is below the threshold.

$$Z_{ik} \leq Z_{kk}, \quad \forall i, k \in N \tag{8}$$

Equation 8 ensures that nodes are assigned to a controller node only.

$$b_{ij} = \sum_{k \in N} (Z_{ik} + Z_{jk}) - 1, \quad \forall i, j \in N \tag{9}$$

Equation 9 makes sure that b_{ij} is 1 if node *i* and node *j* have been assigned the same controller.

$$\sum_{k \in N} X_{ij}^{kl} = Z_{jl}, \quad \forall i, j, l \in N \tag{10}$$

Equation 10 is necessary to ensure that only one out of many *k* nodes which are connected to *i* is selected, such that the selected *k* is connected to the *l* node, which has been selected as a controller to node *j*.

$$\sum_{l \in N} X_{ij}^{kl} = Z_{ik}, \forall i, j, k \in N \tag{11}$$

Equation 11 can be explained in a manner similar to that of equation 10 above, in terms of *l*.

It is to be noted that the formulation above is a nonlinear integer programming formulation due to the term $X_{ij}^{kl}[\text{Ü} + \hat{\text{U}}\, b_{ij}]$ in the objective function. The non-linearity in the above term is eliminated by introducing a variable Y_{ij}^{kl} such that:

$$Y_{ij}^{kl} = \{1 \text{ if } X_{ij}^{kl} \text{and } b_{ij} = 1, 0 \text{ otherwise} \quad (12)$$

We add the following additional constraints:

$$Y_{ij}^{kl} \leq X_{ij}^{kl} \quad (13)$$

$$Y_{ij}^{kl} \leq b_{ij} \quad (14)$$

$$Y_{ij}^{kl} \geq X_{ij}^{kl} + b_{ij} - 1 \quad \forall\, i, j, k, l \in N \quad (15)$$

Equations (13)-(15) ensure that Y_{ij}^{kl} = 1 if and only if both X_{ij}^{kl} and b_{ij} = 1, else the value assigned will be 0. However, variables b_{ij} and Y_{ij}^{kl} can be eliminated, reducing the number of variables in the formulation, if the objective function is modified as shown below.

Minimize:

$$\sum_{i \in N} \sum_{k \in N} \sum_{l \in N} \sum_{j \in N} W_{ij}\, C_{ij}^{kl}\, X_{ij}^{kl}\, \text{Ü} + \sum_{i \in N} \sum_{k \in N} \sum_{j \in N} C_{ij}^{kk}\, X_{ij}^{kk}\, \hat{\text{U}} + \sum_{k \in N} F_k\, Z_{kk}$$

The variable X_{ij}^{kk} will be 1 if nodes i and j have been assigned to the same controller *k*, thus, eliminating the variable b_{ij}. With the linearization above, an exact solution can be obtained using the ILP formulation. However, the number of variables in the above formulation is of the order ($n^4 + n^2$). Hence, the ILP formulation is likely to work well only for small-sized networks. For larger networks, the ILP provides results only after significantly large computational overhead. As a result, a heuristic is required for obtaining good solutions quickly for large and practical problem instances.

5.6 Heuristic Approach and Results

Due to the computational complexity of the ILP presented in Section V, results for only small-sized networks can be obtained in a tractable amount of time. For contemporary service-provider networks, such as nation-wide backbone topologies, that have a large number of networking elements in the core networks, the ILP provides results only after significant computational overhead. As a result there is a need to develop a fast heuristic that can obtain quick results while preserving the correctness of the solution, in line with the ILP. We have proposed two heuristic approaches and compared the performance in the subsequent sections.

5.6.1 Heuristic—Random Greedy Approach

In this section, we discuss a *Random Greedy Approach* (RGA) heuristic approach based on a combination of a random approach and a greedy strategy for the controllers' placement problem. Initially, the heuristic performs location, in which, controller nodes are identified in the given network. Initially, all the nodes are sorted in the descending order of their degree of connectivity in the network. Nodes that have a degree of connectivity greater than the average degree of connectivity of the network are considered for candidate controller nodes. Actual controller nodes are selected from a set of candidate controller nodes, using random selection. A random selection code to select the controllers is executed for a fixed number of times to obtain different combinations of the controllers in different sets. A larger number of iterations ensure a larger probability of obtaining a global optimal solution; however, it means excessive computational time for larger data-sets. In this case, the fixed number is N, where N is the number of nodes in the given network instance.

For each combination of the selected controllers for every set, the heuristic performs allocation, that is, it assigns non-controller's nodes to controllers, using the

greedy approach. The allocation task is done as follows: initially, the heuristic performs a "*labeling"* and "*removal"* operation on the non-controller nodes. A controller is selected from a pre-calculated set of controllers and labels are assigned to the nodes which are at a one-hop distance from the controller node. These nodes are allocated to the corresponding controller and removed from the processing set. The heuristic iterates through the remaining non-controller nodes and assigns each node to its closest (shortest-path) controller node, given that the particular controller still has some capacity left to accommodate a new node, else, the heuristic searches for the next-closest controller. These steps are repeated until a controller is assigned to every non-controller node. This process is repeated for all sets of the controllers calculated in the location stage of the algorithm. After allocation is completed, for all the possible combinations of location and allocation generated, the total control traffic overhead is calculated using a formula, as explained in the ILP section, that is, Control Traffic = $\sum_{i,j,k.l \in N} W_{ij} * C_{ij}^{kl} * X_{ij}^{kl} * T_S + \sum_{i,j,k,l \in N} C_{ij}^{kk} * X_{ij}^{kk} * T_{HC}$. Finally, a set that gives an optimal solution, that is, the one that generates the lowest control traffic overheads, is retained as a final solution. The pseudo code for the proposed heuristic algorithm is given below. Complexity of the proposed heuristic is given as *O(N*R*N)*, where *R* is the number of controllers located in the outer loop. However, as $R << N$, we can express the complexity of the heuristic as $O(N^2)$, which is polynomial time.

```
Algorithm: Controller_Placement
Input: Topology Graph G(V, E),
       THRESH as max number of nodes those can be allocated to
       a single controller,
       COST as max cost allowed to be invested for controllers'
       placements
       C as cost for the installation of single controller.
Output: Optimal control-traffic, set of controllers and allocation set
Procedure Main
  nodeCounter ← |V|
  min_Controllers ← nodeCounter / THRESH
  max_Controllers ← COST/C
  sortedByDegree ← Sort_Nodes_byDegre_Descending()
  int Final_Control_Traffic = INFINITY;
 For i = min_Controllers to max_Controllers
  For j = 1 to N
    S ← Weighted_Random_Selection(i) //location
   Allocation(G, S, THRESH)
   Total_Control_Traffic = Calculate_Control_Traffic()
   if (Total_Control_Traffic < Final_Control_Traffic)
      Final_Control_Traffic = Total_Control_Traffic
      Current_Solution = S
   End if
  End For
 End For
End Main
```

```
Algorithm: Allocation
Input: Topology Graph G(V, E), Set S of selected controller nodes,
THRESH.
Output: Non-controller nodes-to-controllers allocation strategy.
Procedure Allocation
  int minDist = INFINITY;
  initialize nodesAssigned = 0 for all nodes
  initialize ParentController = 0 for all nodes
  Forall controllers label all the nodes which are at single hop
  from that controller
  Allocate the labelled nodes to the selected controller
  Remove all the ladled node from the set 'unassigned_nodes'
  Foreach node var1 in set of unassigned_nodes
    For var2 = 0 to noControllers
      current_controller = sortedByDegree[var2]
      int distance = ShortestPath (var1, current_controller)
      if (distance < minDist && nodesAssigned[current_controller]
            < THRESH)
        assigned_Controller = current_controller;
        minDist = distance;
      End if
    End For
  ParentController.Add(var1, assigned_Controller);
  nodesAssigned[assigned_Controller]++;
  End For
End Allocation
```

TABLE 5.1: RGA Heuristic algorithm for Controller Placement and Allocation.

The performance of the RGA heuristic is compared against that of the ILP as shown in TABLE 5.2 below. We observe that the gap between the ILP and heuristic performance is a maximum of 3.68%.

	40% Network Load					80% Network Load				
	ILP		Heuristic			ILP		Heuristic		
Data-sets	Control Traffic Volume (KBytes)	Time taken (S)	Control Traffic Volume (KBytes)	Time taken (S)	GAP (%)	Control Traffic Volume (KBytes)	Time taken (S)	Control Traffic Volume (KBytes)	Time taken (S)	GAP (%)
10-nodes	6.43	22.78	6.54	24.45	1.74	8.99	51.92	9.01	27.44	0.24
15-nodes	19.18	431.06	19.53	48.69	1.87	30.29	705.80	31.22	51.92	3.09
20-nodes	46.17	16997.40	47.23	103.55	2.30	69.50	24667.60	70.96	127.00	2.10
25-nodes	86.11	21450.00	89.20	165.26	3.59	139.10	71616.30	142.45	178.10	2.41
30-nodes	14.26	45426.90	14.75	304.10	3.45	259.65	71205.80	268.37	321.53	3.36
35-nodes	23.94	57253.68	24.24	396.00	1.24	443.15	82250.60	457.89	411.33	3.33
40-nodes	31.78	68311.83	32.48	519.38	2.20	596.73	83702.40	615.14	559.00	3.09
45-nodes	51.13	65771.20	53.01	691.51	3.68	1233.73	58441.80	1271.26	697.34	3.04
50-nodes	---	---			---	---	---			---

TABLE 5.2: Comparison between ILP and the proposed RGA heuristic.

5.6.2 Heuristic—Weighted Random Approach

In this section, a *Weighted Random* heuristic *Approach* (WRA) for the "controllers placement problem" is proposed. As mentioned in [90-91, 98-99], the random meta-heuristic approach is well suited for large size difficult problems and provides a solution in a tractable amount of time with close-to-optimal results. We use a "*weighted random approach"* by assigning weights to the elements which need to be selected randomly. The functioning of the proposed heuristic is divided into two distinct parts, that is, location and allocation. Initially, the heuristic performs a location task in which controller nodes are identified in the given network. Various combinations of controllers are chosen using a weighted random approach. The WRA performs transitions to obtain an improved solution by adjusting the locations of controllers, locations such that the

overall distance between each pair of nodes through their respective controllers is minimized. Initially, all the nodes are sorted in the descending order of their degree of connectivity. Nodes that have a degree of connectivity greater than the average degree of connectivity of the network are considered as candidate controller nodes. Actual controller nodes are selected from a set of candidate controller nodes, using weighted random selection.

By weighted random selection, we mean that, the controller nodes are selected randomly; however, the nodes that have a higher degree of connectivity have higher probability of getting selected as compared to the nodes that have a lesser degree of connectivity. Different sets of controllers with sizes of the sets varying from the minimum number of controllers that are required to the maximum number of controllers that can be installed are obtained—so that the cost and the threshold requirements are met. A weighted random selection code is executed for Ω number of times where Ω is a large number that is dependent on the size of the problem instance. A larger value of Ω ensures greater probability of obtaining a global optimal solution. However, a larger value of Ω also means excessive computational time for larger data-sets. In this particular case, we tried various numbers of iterations for the outer and inner loops. Results till the 100-node topology for various combinations of iterations have been obtained (at 80% traffic load) and displayed in TABLE 5.3 below.

No of iterations (outer & inner)	N*D & R^2*D'			∛(N^2)*D & R^2*D'			∛(N^2)*D & N*D		
Data-sets	Control Traffic Volume (KBytes)	Time taken (S)	GAP (%)	Control Traffic Volume (KBytes)	Time taken (S)	GAP (%)	Control Traffic Volume (KBytes)	Time taken (S)	GAP (%)
10-nodes	9.10	31.49	1.20	9.20	27.15	2.31	9.10	38.87	1.21
15-nodes	30.89	64.17	1.98	31.27	55.32	3.23	30.88	79.21	1.95
20-nodes	71.56	139.07	2.96	71.73	119.89	3.21	71.48	171.66	2.85
25-nodes	141.65	204.80	1.84	142.76	176.55	2.63	141.76	252.79	1.91
30-nodes	266.52	390.43	2.65	267.69	336.58	3.10	267.73	481.92	3.11
35-nodes	448.64	524.20	1.24	457.58	411.90	3.26	454.24	763.24	2.50
40-nodes	614.08	720.80	2.91	614.16	521.38	2.92	615.24	1049.49	3.10
45-nodes	1265.33	1026.46	2.56	1267.69	684.88	2.75	1267.13	1494.53	2.71
50-nodes	1526.21	1648.63	---	1539.46	1021.23	---	1536.67	2647.67	---
100-Nodes	2289.46	3398.82	---	2292.19	1498.99	---	2294.35	5448.69	---

No of iterations (outer & inner)	2*N*R/D & 4*R^2*D'			4*N*D/R & R^2*D'			4*N*D/R & 2*N*D/R		
Data-sets	Control Traffic Volume (KBytes)	Time taken (S)	GAP (%)	Control Traffic Volume (KBytes)	Time taken (S)	GAP (%)	Control Traffic Volume (KBytes)	Time taken (S)	GAP (%)
10-nodes	9.10	45.59	1.21	9.10	33.56	1.20	9.18	44.44	2.15
15-nodes	30.72	81.37	1.44	31.24	67.38	3.15	31.38	76.73	3.62
20-nodes	70.94	195.29	2.07	72.38	143.27	4.15	72.09	169.99	3.73
25-nodes	140.11	425.82	0.72	141.06	219.53	1.41	140.65	242.55	1.11
30-nodes	264.15	823.62	1.73	265.93	395.35	2.42	267.15	675.74	2.89
35-nodes	448.37	1087.07	1.18	449.28	527.92	1.38	453.97	588.93	2.44
40-nodes	612.35	1381.57	2.62	614.00	720.77	2.89	615.05	1105.80	3.07
45-nodes	1264.19	1755.89	2.47	1298.46	1021.33	5.25	1267.58	1581.36	2.74
50-nodes	1524.44	3063.99	---	1698.02	1634.46	---	1538.45	1731.07	---
100-Nodes	2288.23	5777.66	---	2308.16	2943.85	---	2293.19	3962.71	---

TABLE 5.3: Various combinations of iterations for the outer and inner loops.

We chose the combination which produced good solutions regularly in a reasonable amount of computational time. That is, we chose $\Omega = 2*N*R/D$ and $\Psi =$

$4*R^2*D'$ (Ψ is a number for iterations of the inner loop explained in the subsequent section of this paragraph), where N is the total number of nodes in the network, D is the average degree of connectivity of the network, R is number of controllers located in the outer loop and D' is the average degree of connectivity of the set of controllers. For every set of the selected controllers in the step above, the heuristic performs an allocation task, that is, it assigns non-controller nodes to controller nodes, using the *Weighted Random Approach* again. A value Φ is calculated for every pair of the non-controller nodes, i and j, such that, $\Phi_{ij} = W_{ij}/D_{ij}$, where W_{ij} is the volume of services configured between i and j and D_{ij} is the distance between i and j. The pair that has a higher value of Φ has a higher probability of getting allocated to the same controller. The rationale here is that, the more the traffic volume between the two nodes and the lesser the distance between them, the more it is desirable that both the nodes are allocated to the same controller. This process is repeated for Ψ number of times, where $\Psi = 4*R^2*D'$.

Total control traffic overhead is calculated for all the combinations of controller nodes and various allocation strategies spawned by the WRA heuristic, using a formula explained in the ILP section: Control Traffic = $\sum_{i,j,k.l \in N} W_{ij} * C_{ij}^{kl} * X_{ij}^{kl} * T_S + \sum_{i,j,k,l \in N} C_{ij}^{kk} * X_{ij}^{kk} * T_{HC}$. Finally, a solution that generates minimum control traffic overhead is retained. The pseudo code for the heuristic algorithm is given in TABLE 5.4 below. It is assumed that the topology matrix and the traffic demand matrix is given as an input. A threshold value indicating the maximum number of nodes that can be allocated to a single controller is also provided. The heuristic provides an output which indicates the locations of the controllers and an allocation strategy to assign non-controller nodes to their corresponding controller nodes. The complexity of the proposed WRA heuristic is given as $O(\Omega*\Psi*N)$, that is, $O(8*R^3*N^2*D'/D)$. However, as $(D, D') < R << N$, the complexity of the heuristic can be expressed as $O(N^2)$.

```
Algorithm: Controller_Placement
Input: Topology Graph G(V, E), Traffic Demand Matrix W,
       Γ as max number of nodes those can be allocated to a single controller,
       COST as max cost allowed to be invested for controllers' placements,
       C as cost for the installation of single controller.
Output: Total control-traffic overhead, S as a set of controllers, A as a allocation strategy.
Procedure Main
    nodeCounter ← |V|
    min_Controllers ← nodeCounter / Γ
    max_Controllers ← COST/C
    sortedByDegree ← Sort_Nodes_byDegre_Descending()
    int Final_Control_Traffic = INFINITY;
    For i = min_Controllers to max_Controllers
        For j = 1 to Ω
            S ← Weighted_Random_Selection(i) //locating the controllers
            For k = 1 to Ψ
                A = Weighted_Random_Allocation(G, W, S) //allocation strategy
                Total_Control_Traffic = Calculate_Control_Traffic()
                if (Total_Control_Traffic < Final_Control_Traffic)
                    Final_Control_Traffic = Total_Control_Traffic
                    Current_Solution = (S, A)
                End if
            End For
        End For
    End For
End Main
```

TABLE 5.4: Heuristic algorithm for Controller Placement and Allocation.

The performance of the WRA heuristic is now compared against that of the ILP to demonstrate that the WRA heuristic follows the ILP closely in a tractable amount of time. The ILP has been solved using MOSEK ILP, an open-source optimization tool, while the heuristic algorithm was implemented in a .Net environment using C# programming language. The results are displayed in TABLE 5.5 below. The results for two different traffic scenarios, that is, 40% and 80% traffic load are obtained. An entry of '–' in the TABLE 5.5 indicates that the ILP was unable to generate results in a reasonable computational time. We observe that the gap (obtained as a heuristic-solution minus optimal-solution expressed as a percentage) between the ILP and heuristic is a maximum of 2.66%, which is well within acceptable limits. However, the time taken by the heuristic to solve the problem is significantly lesser compared to that of the ILP. The computational effort required for the ILP degrades as problem size increases.

	40% Network Load					80% Network Load				
	ILP		Heuristic (Weighted Random)			ILP		Heuristic (Weighted Random)		
Data-sets	Control Traffic Volume (KBytes)	Time taken (S)	Control Traffic Volume (KBytes)	Time taken (S)	GAP (%)	Control Traffic Volume (KBytes)	Time taken (S)	Control Traffic Volume (KBytes)	Time taken (S)	GAP (%)
10-nodes	6.43	22.78	6.60	20.78	2.74	8.99	51.92	9.10	22.15	1.20
15-nodes	19.18	431.06	19.53	75.34	1.84	30.29	705.80	30.72	81.37	1.44
20-nodes	46.17	16997.40	47.14	181.35	2.09	69.50	24667.60	70.94	195.29	2.07
25-nodes	86.11	21450.00	88.35	418.35	2.60	139.10	71616.30	140.11	425.82	0.72
30-nodes	142.57	45426.90	143.98	790.35	0.99	259.65	71205.80	264.15	823.62	1.73
35-nodes	239.39	57253.68	245.77	1011.33	2.66	443.15	82250.60	448.37	1087.07	1.18
40-nodes	317.83	68311.83	323.64	1278.35	1.83	596.73	83702.40	612.35	1381.57	2.62
45-nodes	511.33	65771.20	517.35	1703.99	1.18	1233.73	58441.80	1264.19	1755.89	2.47
50-nodes	—	—			—	—	—			—

TABLE 5.5: Comparison between ILP and the proposed WRA heuristic.

5.6.3 Comparison—RGA versus WRA

Next, we compare the performance of a *Weighted Random Approach* (WRA) with that of a *Random Greedy Approach* (RGA) for larger data-sets, that is, till a 1000-node topology with traffic load varying from 10% to 90%, in different types of topology. The Results are displayed in TABLE 5.6 below. We observe that, WRA outperforms RGA (91% of the total data-sets).

No of Nodes	Link Density	Traffic (%)	100-Split			200-Split			No Split		
			Weighted Random	Random Greedy	Difference (MB)	Weighted Random	Random Greedy	Difference (MB)	Weighted Random	Random Greedy	Difference (MB)
200		10	531.79	623.94	92.14	1390.42	1506.78	116.36	1390.42	1506.78	116.36
		20	1542.03	1658.90	116.87	1725.34	2006.78	281.44	1725.34	2006.78	281.44
	Ring	30	2041.33	2434.90	393.58	3333.74	3654.02	320.28	3333.74	3654.02	320.28
		40	3001.96	3706.99	705.03	4870.85	5749.03	878.18	4870.85	7449.03	2578.18
		50	3123.48	3748.79	625.31	6031.86	7476.42	1444.56	6031.86	7476.42	1444.56
	Mesh-sparse	60	3082.52	3791.13	708.61	6692.03	7813.58	1121.55	6692.03	7813.58	1121.55
		70	3290.65	3886.82	596.17	8588.16	10702.83	2114.67	8588.16	10702.83	2114.67
		80	3775.99	4067.23	291.24	9608.07	11008.53	1400.46	9608.07	11008.53	1400.46
	Mesh-Dense	90	3510.62	3957.17	446.56	11379.63	12483.62	1103.99	11379.63	12483.62	1103.99
300		10	1214.45	1126.78	-87.67	870.87	967.39	96.52	3127.69	3564.91	437.22
		20	1864.15	2235.12	370.97	2161.43	2381.69	220.26	6880.80	7455.70	574.89
	Ring	30	2201.28	2545.65	344.37	2151.56	2653.99	502.43	9995.19	10908.56	913.37
		40	3113.13	3807.88	694.75	3690.43	3957.19	266.76	12307.55	14263.03	1955.48
		50	3589.62	3846.87	257.25	3741.37	4028.62	287.25	13572.56	13412.53	-160.03
	Mesh-sparse	60	3210.76	3856.43	645.67	4352.89	4033.70	-319.20	19464.03	19553.03	89.00
		70	3413.49	3985.62	572.13	4517.52	5149.39	631.88	22383.97	26638.64	4254.67
		80	3397.52	4067.23	669.71	6471.51	6966.01	494.51	23969.65	26825.87	2856.22
	Mesh-Dense	90	3651.27	4068.54	417.27	8204.49	8204.49	0.00	29321.45	28422.18	-899.27
400		10	1071.53	1057.92	-13.61	1708.11	1890.54	182.42	4383.46	5078.73	695.27
		20	1751.90	2031.83	279.93	4247.87	5245.90	998.03	9772.77	11241.78	1469.00
	Ring	30	2507.53	3046.80	539.27	6381.03	7620.89	1239.86	13860.20	16459.82	2599.62
		40	3728.55	4202.92	474.37	8108.69	10029.06	1920.37	21835.22	20976.20	-859.02
		50	4128.85	4681.12	552.27	11329.50	13357.00	2027.50	21730.09	21886.82	156.73
	Mesh-sparse	60	4102.17	4772.68	670.51	9429.47	10508.59	1079.12	32785.37	35875.98	3090.61
		70	4295.17	4943.70	648.53	12443.07	15355.66	2912.59	33805.87	38388.79	4582.92
		80	4575.57	4974.33	398.76	15927.80	18245.23	2317.44	41458.47	48727.67	7269.20
	Mesh-Dense	90	4341.51	4918.42	576.90	18483.32	21621.39	3138.07	43615.72	49899.99	6284.27
500		10	941.82	1140.98	199.16	3270.22	3187.30	-82.91	7098.94	7224.96	126.02
		20	2512.80	3115.87	603.07	6126.40	6728.14	601.74	14573.09	15877.69	1304.60
	Ring	30	4447.00	4918.73	471.73	7722.42	9572.23	1849.82	21410.26	23417.66	2007.40
		40	5256.56	5846.90	590.34	10719.84	11894.53	1174.70	28703.79	33356.21	4652.42
		50	5276.68	5859.79	583.12	12793.21	14822.18	2028.97	32881.78	36320.33	3438.55
	Mesh-sparse	60	5649.80	6378.86	729.05	17060.91	19494.82	2433.91	45016.15	48796.29	3780.14
		70	5346.48	6681.39	1334.91	20927.35	25351.85	4424.50	48431.67	56418.80	7987.13
		80	5608.90	6773.11	1164.21	28728.22	33032.14	4303.92	56341.90	66614.83	10272.93
	Mesh-Dense	90	6647.15	7984.80	1337.65	33626.10	37897.28	4271.18	64509.71	77122.93	12613.22
600		10	2141.96	2478.76	336.80	3840.29	4796.25	955.96	10195.87	10105.77	-90.09
		20	6023.06	7024.53	1001.47	9029.05	10299.85	1270.79	21255.84	22671.52	1415.68
	Ring	30	9904.03	10668.47	764.44	11692.68	12945.56	1252.88	26148.82	31160.02	5011.19

		40	11721.55	12711.38	989.83	12950.19	16178.72	3228.53	38811.43	44816.63	6005.19
		50	12843.22	14178.48	1335.26	21332.61	24269.71	2937.11	45941.78	50837.61	4895.83
	Mesh-sparse	60	12482.19	13630.69	1148.50	23450.90	26757.26	3306.36	57417.36	64852.44	7435.07
		70	13361.52	15912.49	2550.97	30501.83	36365.49	5863.66	70047.02	76696.13	6649.11
		80	13720.70	15419.37	1698.68	38005.99	41595.09	3589.09	75866.03	85722.92	9856.88
	Mesh-Dense	90	16301.04	19607.63	3306.59	39351.98	48208.67	8856.68	88520.53	98360.82	9840.29
700		10	2611.37	2812.09	200.72	4675.61	5413.50	737.89	11889.43	12822.49	933.06
		20	4539.00	5062.65	523.64	9679.72	12095.31	2415.59	22728.05	27327.39	4599.34
	Ring	30	8660.22	9814.71	1154.49	13100.75	14408.50	1307.75	35285.51	38802.15	3516.65
		40	11110.45	13731.49	2621.04	14207.88	17485.25	3277.37	53895.02	58952.77	5057.75
		50	12731.30	15590.84	2859.54	22555.09	26017.74	3462.65	58872.74	64771.78	5899.05
	Mesh-sparse	60	12836.20	15329.22	2493.02	24671.71	27956.57	3284.85	78409.42	86419.01	8009.59
		70	15258.91	17845.08	2586.18	33262.86	38109.06	4846.20	96198.99	101748.14	5549.16
		80	15137.68	16985.86	1848.18	34768.32	42897.00	8128.67	107777.49	109065.50	1288.01
	Mesh-Dense	90	16881.96	20976.42	4094.46	42398.52	49417.67	7019.15	111253.34	125392.14	14138.81
800		10	2580.83	2980.15	399.32	5106.34	6268.09	1161.75	16008.69	17803.09	1794.40
		20	5724.70	6478.96	754.26	10791.69	13237.29	2445.60	31347.51	34671.76	3324.25
	Ring	30	9441.39	11043.29	1601.90	14383.11	16937.89	2554.78	47679.60	51229.79	3550.18
		40	13386.69	15114.15	1727.46	16611.38	19013.63	2402.24	61400.50	73891.38	12490.88
		50	15234.35	16621.69	1387.34	25363.01	27694.46	2331.45	68372.84	84254.92	15882.08
	Mesh-sparse	60	15720.18	17116.06	1395.89	27134.98	29289.34	2154.36	101640.54	112889.83	11249.29
		70	16563.71	19304.57	2740.86	36745.32	39797.26	3051.93	119670.87	135200.35	15529.48
		80	17186.77	18554.38	1367.61	47832.72	54802.97	6970.24	126529.65	142303.29	15773.64
	Mesh-Dense	90	18055.34	22063.77	4008.43	40600.83	50514.01	9913.18	142612.72	161529.32	18916.59
900		10	3783.70	4263.46	479.76	6580.14	7800.43	1220.30	19101.99	21347.36	2245.37
		20	7282.47	8424.78	1142.31	13137.77	14397.99	1260.22	39478.68	41996.23	2517.55
	Ring	30	10661.91	12757.55	2095.64	17522.54	18920.91	1398.36	53505.89	60517.94	7012.05
		40	15342.69	16637.79	1295.10	18996.50	20940.32	1943.82	74942.64	84307.71	9365.06
		50	16290.56	17810.48	1519.92	27184.16	29159.08	1974.92	88783.69	101724.17	12940.48
	Mesh-sparse	60	16980.03	18586.41	1606.38	24809.15	30785.94	5976.79	116095.23	125807.49	9712.26
		70	17287.54	21050.73	3763.20	38143.88	41175.18	3031.30	127390.70	155358.50	27967.80
		80	15864.79	19827.14	3962.35	49661.35	56782.83	7121.48	154233.51	165698.61	11465.10
	Mesh-Dense	90	19929.92	23338.36	3408.44	48612.06	52217.56	3605.50	155787.42	183724.54	27937.12
1000		10	3221.05	3863.92	642.87	5942.46	7123.09	1180.63	21102.76	22477.82	1375.06
		20	7610.25	8464.54	854.29	12451.16	13949.50	1498.34	40479.18	44559.63	4080.45
	Ring	30	9195.95	10505.54	1309.59	17903.04	20179.75	2276.71	61182.44	65800.67	4618.23
		40	9230.89	10802.94	1572.05	17596.31	21944.55	4348.24	80816.79	91757.63	10940.84
		50	15484.24	19206.66	3722.42	25564.93	30820.98	5256.04	97149.74	108875.78	11726.04
	Mesh-sparse	60	17756.62	20376.01	2619.39	30242.94	32523.98	2281.04	125778.61	137087.42	11308.81
		70	18919.27	22231.57	3312.30	43292.77	48322.34	5029.56	149657.35	163168.50	13511.15
		80	25311.62	29288.63	3977.02	57340.52	65831.32	8490.81	166441.96	180513.90	14071.94
	Mesh-Dense	90	34852.97	38051.09	3198.12	61053.06	68362.14	7309.08	185413.55	198573.74	13160.19

TABLE 5.6: RGA versus WRA.

5.6.4 WRA Evaluation and Results

More rigorous results for a larger topology with up to 1000-nodes and varying traffic loads are obtained using the WRA heuristic. The results of computational experiments are presented in TABLE 5.7 to demonstrate the effectiveness of the proposed heuristic algorithm for large-size service-provider networks. The results demonstrate that the growth in control traffic within the networks is exponential—when one takes into account the number of nodes and services configured in the network, hampering the advantages of the centralized control plane architecture. To overcome the problem of control traffic overhead in centrally managed networks, a division of the larger network into multiple sub-networks is proposed so that the percentage of data-traffic in the network does not drop below a threshold. In other words, the total control traffic does not grow beyond a threshold. We have generated random cluster graphs using the "networkx" library [58] and Python. We have considered a network size with the nodes ranging from 100-1000, where the average degree of a node is three (classical metropolitan networks). We have considered different topologies such as: (1) Ring, (2) Sparse Mesh and (3) Dense Mesh. Control traffic at different network traffic loads varying from 10% to 100% is noted down. All traffic demands and links in the simulation are assumed to be bi-directional. Different sets of traffic demands have been generated for various network loads, where the rate of arrival of traffic demand is characterized by a *Poisson* distribution and where service granularity is exponentially distributed [100]. The volume of the total data traffic and control traffic in the network is measured and the results are tabulated in TABLE 5.7, below. An entry of "*NA*" in the table below indicates that splitting the networks into smaller sub-networks of corresponding size is not possible.

No of Nodes	Link Density	Traffic (%)	100-Split		200-Split		300-Split		400-Split		500-Split		No Split	
			Total CT (MB)	CT (%)	Total CT (MB)	CT (%)	Total CT (MB)	CT (%)	Total CT (MB)	CT (%)	Total CT (MB)	CT (%)	Total CT (MB)	CT (%)
200		10	531.79	2.66	1390.42	6.95	NA	NA	NA	NA	NA	NA	1390.42	6.95
		20	1542.03	3.86	1725.34	4.31	NA	NA	NA	NA	NA	NA	1725.34	4.31
	Ring	30	2041.33	3.40	3668.24	6.11	NA	NA	NA	NA	NA	NA	3668.24	6.11
		40	3001.96	3.75	4870.85	6.09	NA	NA	NA	NA	NA	NA	4870.85	6.09
		50	3123.48	3.12	6031.86	6.03	NA	NA	NA	NA	NA	NA	6031.86	6.03
	Mesh-sparse	60	3082.52	2.57	6692.03	5.58	NA	NA	NA	NA	NA	NA	6692.03	5.58
		70	3290.65	2.35	8588.16	6.13	NA	NA	NA	NA	NA	NA	8588.16	6.13
		80	3775.99	2.36	9608.07	6.01	NA	NA	NA	NA	NA	NA	9608.07	6.01
	Mesh-Dense	90	3510.62	1.95	11379.63	6.32	NA	NA	NA	NA	NA	NA	11379.63	6.32
300		10	1214.45	4.05	870.87	2.90	3127.69	10.43	NA	NA	NA	NA	3127.69	10.43
		20	1864.15	3.11	2161.43	3.60	6880.80	11.47	NA	NA	NA	NA	6880.80	11.47
	Ring	30	2201.28	2.45	2151.56	2.39	9995.19	11.11	NA	NA	NA	NA	9995.19	11.11
		40	3113.13	2.59	3690.43	3.08	12307.55	10.26	NA	NA	NA	NA	12307.55	10.26
		50	3589.62	2.39	3741.37	2.49	13572.56	9.05	NA	NA	NA	NA	13572.56	9.05
	Mesh-sparse	60	3210.76	1.78	4352.89	2.42	19464.03	10.81	NA	NA	NA	NA	19464.03	10.81
		70	3413.49	1.63	4517.52	2.15	22383.97	10.66	NA	NA	NA	NA	22383.97	10.66
		80	3397.52	1.42	6471.51	2.70	23969.65	9.99	NA	NA	NA	NA	23969.65	9.99
	Mesh-Dense	90	3651.27	1.35	8204.49	3.04	29321.45	10.86	NA	NA	NA	NA	29321.45	10.86
400		10	1071.53	2.68	1708.11	4.27	3377.99	8.44	4383.46	10.96	NA	NA	4383.46	10.96
		20	1751.90	2.19	4247.87	5.31	7640.74	9.55	9772.77	12.22	NA	NA	9772.77	12.22
	Ring	30	2507.53	2.09	6381.03	5.32	14213.80	11.84	13860.20	11.55	NA	NA	13860.20	11.55
		40	3728.55	2.33	8108.69	5.07	17058.23	10.66	21835.22	13.65	NA	NA	21835.22	13.65
		50	4128.85	2.06	11329.50	5.66	18587.65	9.29	21730.09	10.87	NA	NA	21730.09	10.87
	Mesh-sparse	60	4102.17	1.71	9429.47	3.93	21185.72	8.83	32785.37	13.66	NA	NA	32785.37	13.66
		70	4295.17	1.53	12443.07	4.44	26093.69	9.32	33805.87	12.07	NA	NA	33805.87	12.07
		80	4983.84	1.56	15927.80	4.98	29534.93	9.23	41458.47	12.96	NA	NA	41458.47	12.96
	Mesh-Dense	90	4341.51	1.21	18483.32	5.13	36385.42	10.11	43615.72	12.12	NA	NA	43615.72	12.12
500		10	941.82	1.88	3270.22	6.54	5320.19	10.64	6520.10	13.04	7098.94	14.20	7098.94	14.20
		20	2512.80	2.51	6835.40	6.84	12333.68	12.33	14160.98	14.16	16190.09	16.19	14573.09	14.57
	Ring	30	4447.00	2.96	7722.42	5.15	18106.05	12.07	19395.88	12.93	21410.26	14.27	21410.26	14.27
		40	5256.56	2.63	10719.84	5.36	21000.52	10.50	23581.29	11.79	28703.79	14.35	28703.79	14.35
		50	5276.68	2.11	12793.21	5.12	26837.80	10.74	26173.62	10.47	32881.78	13.15	32881.78	13.15
	Mesh-sparse	60	5649.80	1.88	17060.91	5.69	37127.88	12.38	35695.24	11.90	45016.15	15.01	45016.15	15.01
		70	5346.48	1.53	20927.35	5.98	40825.17	11.66	39596.77	11.31	48431.67	13.84	48431.67	13.84
		80	5608.90	1.40	28728.22	7.18	45833.64	11.46	52451.30	13.11	56341.90	14.09	56341.90	14.09
	Mesh-Dense	90	6647.15	1.48	33626.10	7.47	51236.49	11.39	53870.67	11.97	64509.71	14.34	64509.71	14.34
600		10	2592.96	4.32	3840.29	6.40	6499.70	10.83	8914.81	14.86	8451.99	14.09	10195.87	16.99
		20	6023.06	5.02	9029.05	7.52	15318.46	12.77	18262.30	15.22	19935.55	16.61	21255.84	17.71
	Ring	30	9904.03	5.50	11692.68	6.50	23376.55	12.99	26815.84	14.90	35210.83	19.56	26148.82	14.53
		40	11721.55	4.88	12950.19	5.40	31352.81	13.06	33835.08	14.10	37934.88	15.81	38811.43	16.17
		50	12843.22	4.28	21332.61	7.11	43784.26	14.59	41005.99	13.67	46513.51	15.50	45941.78	15.31
	Mesh-sparse	60	12482.19	3.47	23450.90	6.51	53489.41	14.86	52084.53	14.47	54734.81	15.20	57417.36	15.95

		70	13361.52	3.18	30501.83	7.26	61693.18	14.69	64532.08	15.36	57905.31	13.79	70047.02	16.68
		80	13720.70	2.86	38005.99	7.92	81081.98	16.89	76188.91	15.87	83017.28	17.30	75866.03	15.81
	Mesh-Dense	90	16301.04	3.02	39351.98	7.29	80967.27	14.99	85913.69	15.91	85027.44	15.75	88520.53	16.39
700		10	2611.37	3.73	4675.61	6.68	7577.26	10.82	7692.57	10.99	10253.87	14.65	12889.43	18.41
		20	4539.00	3.24	9679.72	6.91	14735.50	10.53	16722.39	11.94	18458.07	13.18	22728.05	16.23
	Ring	30	8660.22	4.12	13100.75	6.24	21421.58	10.20	27420.15	13.06	30341.74	14.45	35285.51	16.80
		40	11110.45	3.97	14207.88	5.07	28448.23	10.16	39220.61	14.01	39356.84	14.06	53895.02	19.25
		50	12731.30	3.64	22555.09	6.44	36609.23	10.46	51256.05	14.64	50911.58	14.55	58872.74	16.82
	Mesh-sparse	60	12836.20	3.06	24671.71	5.87	45900.13	10.93	59265.19	14.11	64557.68	15.37	78409.42	18.67
		70	15258.91	3.11	33262.86	6.79	56875.18	11.61	74495.23	15.20	73433.54	14.99	96198.99	19.63
		80	15137.68	2.70	34768.32	6.21	76487.35	13.66	83392.66	14.89	76535.39	13.67	107777.49	19.25
	Mesh-Dense	90	16881.96	2.68	42398.52	6.73	77712.68	12.34	95391.08	15.14	89032.04	14.13	111253.34	17.66
800		10	2580.83	3.23	5106.34	6.38	9034.53	11.29	9813.50	12.27	11743.11	14.68	16008.69	20.01
		20	5724.70	3.58	10791.69	6.74	20711.13	12.94	25768.16	16.11	25359.77	15.85	31347.51	19.59
	Ring	30	12041.39	5.02	14383.11	5.99	27527.64	11.47	31886.58	13.29	38356.74	15.98	47679.60	19.87
		40	13386.69	4.18	16611.38	5.19	50694.92	15.84	49338.02	15.42	53358.55	16.67	61400.50	19.19
		50	15234.35	3.81	25363.01	6.34	53425.92	13.36	64235.53	16.06	48548.62	12.14	68372.84	17.09
	Mesh-sparse	60	17720.18	3.69	27134.98	5.65	68111.09	14.19	79390.70	16.54	83163.41	17.33	101640.54	21.18
		70	16563.71	2.96	36745.32	6.56	74446.92	13.29	86954.09	15.53	87091.49	15.55	119670.87	21.37
		80	18186.77	2.84	47832.72	7.47	92132.02	14.40	126204.35	19.72	108148.63	16.90	126529.65	19.77
	Mesh-Dense	90	18055.34	2.51	40600.83	5.64	109844.85	15.26	125645.17	17.45	113920.81	15.82	142612.72	19.81
900		10	3783.70	4.20	6580.14	7.31	13398.27	14.89	12625.40	14.03	14529.72	16.14	19101.99	21.22
		20	7282.47	4.05	13137.77	7.30	22993.32	12.77	29118.75	16.18	28933.54	16.07	39478.68	21.93
	Ring	30	10661.91	3.95	17522.54	6.49	36430.05	13.49	36168.26	13.40	39218.32	14.53	53505.89	19.82
		40	15342.69	4.26	18996.50	5.28	48606.24	13.50	52601.46	14.61	57220.85	15.89	74942.64	20.82
		50	16290.56	3.62	27184.16	6.04	60589.37	13.46	66262.75	14.73	65010.25	14.45	88783.69	19.73
	Mesh-sparse	60	19980.03	3.70	24809.15	4.59	88291.32	16.35	84296.96	15.61	82807.28	15.33	116095.23	21.50
		70	17287.54	2.74	42143.88	6.69	89032.20	14.13	98670.74	15.66	97238.46	15.43	127390.70	20.22
		80	15864.79	2.20	49661.35	6.90	103215.20	14.34	118508.68	16.46	132639.02	18.42	154233.51	21.42
	Mesh-Dense	90	19929.92	2.46	48612.06	6.00	124008.49	15.31	125412.22	15.48	126832.28	15.66	155787.42	19.23
1000		10	3221.05	3.22	5942.46	5.94	11166.73	11.17	14183.18	14.18	14547.71	14.55	21102.76	21.10
		20	8754.25	4.38	12451.16	6.23	26653.35	13.33	28640.37	14.32	29632.76	14.82	40479.18	20.24
	Ring	30	9195.95	3.07	17903.04	5.97	46090.73	15.36	40602.11	13.53	47289.62	15.76	61182.44	20.39
		40	9230.89	2.31	17596.31	4.40	57844.35	14.46	57251.95	14.31	63585.95	15.90	80816.79	20.20
		50	15484.24	3.10	25564.93	5.11	63131.69	12.63	86790.46	17.36	72749.94	14.55	109149.74	21.83
	Mesh-sparse	60	17756.62	2.96	30242.94	5.04	85789.36	14.30	93531.16	15.59	92887.47	15.48	125778.61	20.96
		70	18919.27	2.70	43292.77	6.18	91548.07	13.08	101993.37	14.57	98592.76	14.08	149657.35	21.38
		80	25311.62	3.16	57340.52	7.17	100199.46	12.52	124321.63	15.54	115618.85	14.45	182441.96	22.81
	Mesh-Dense	90	34852.97	3.87	61053.06	6.78	123235.77	13.69	147408.72	16.38	134849.95	14.98	185413.55	20.60

TABLE 5.7: WRA Heuristic Results—larger data-sets.

It is observed from the results that the control traffic in a network grows exponentially with the number of nodes and the network traffic. It is also observed that, larger networks (typically beyond 200-nodes) which are controlled by a single controller generate significantly higher control traffic, beyond the acceptable limit of

standard service-provider networks [100-102]. For example, in a dense mesh network of 600-nodes, the control traffic is in the range of 15-20% of the data traffic at all traffic loads. The control traffic increases exponentially with the size of the network. For example, in a 400-node dense mesh network, total control traffic is approximately 21 GB and total control traffic generated in an 800-node network at the same traffic load is 126 GB, demonstrating the exponential growth. This exponential growth in the control traffic is not desirable as it may cause delays in the actual data-flow by congesting the network. The control traffic in larger networks is reduced significantly if networks are split into smaller sub-networks. For example, in the same dense mesh network of 600-nodes, control traffic gets reduced to less than 5% and 8% of the total traffic, if the network is split into smaller sub-networks of sizes 100-nodes and 200-nodes each, respectively; which is acceptable in standard service-provider networks. For a more intuitive understanding of the results, graphs are plotted and displayed in Fig. 5.9 below. The same pattern can be observed in the graphs plotted below for all sizes of networks with different traffic loads. The results displayed in Fig. 5.9 demonstrate the fact that the growth in the control traffic with the network size is exponential and the percentage of the control traffic is beyond the threshold (which is 10% of the total traffic). However, if a larger network is divided into sub-networks of a smaller size, especially of 100 or 200-nodes, the control traffic overhead remains within the acceptable limit. This is a result of the fact that, by dividing larger networks into sub-networks, the broadcast domain for Hello Frames, Hello replies and Management and OAM&P Frames such as CCMs gets limited to networks of smaller sizes. It reduces the total burst size of such broadcast messages. This fact can be observed from graphs displayed in Fig. 5.9 below.

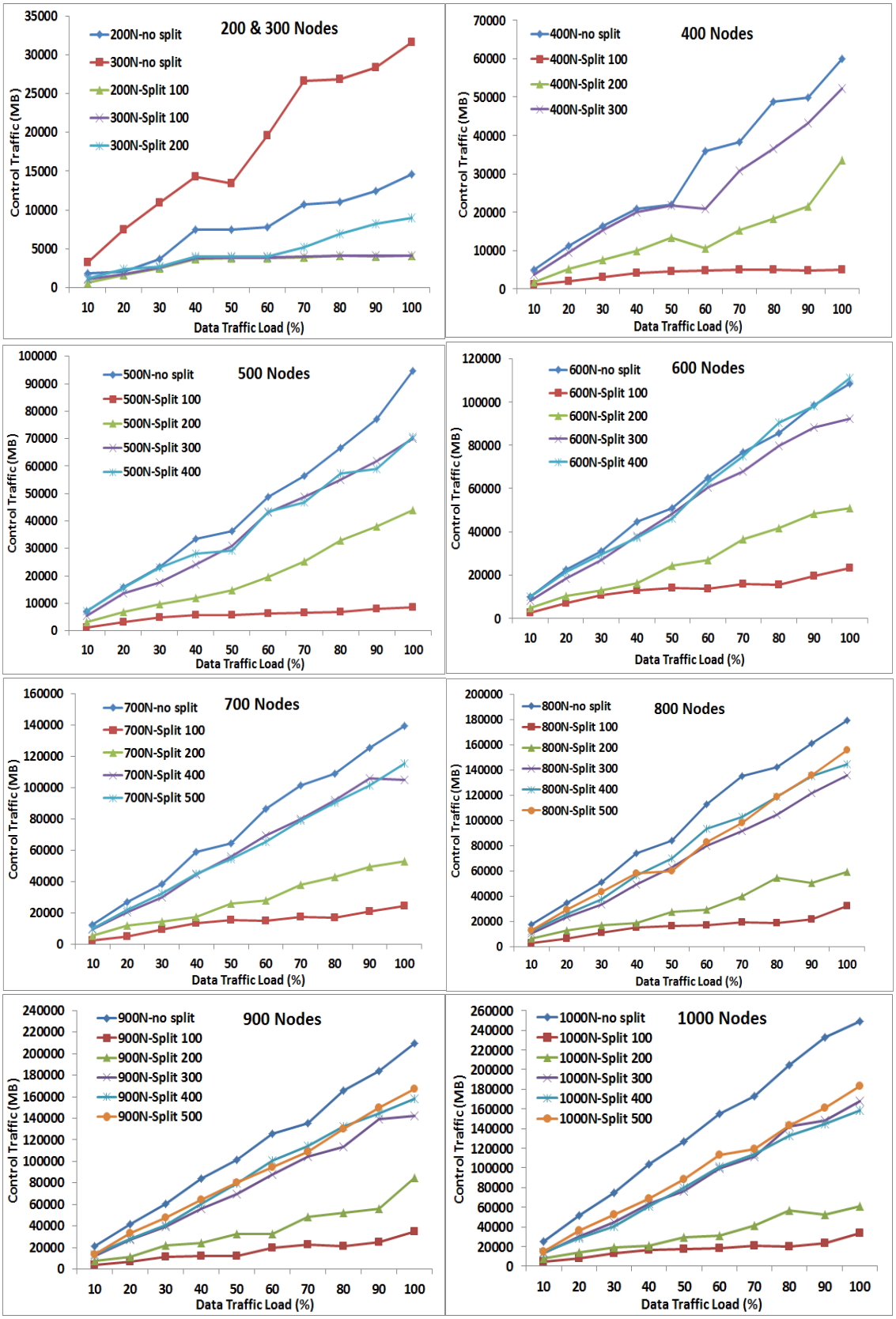


Fig. 5.9. Graphs displaying Control-Traffic in networks of varying sizes—before and after splitting the networks.

In Fig. 5.10 below, graphs have been plotted keeping the network load constant at 80% and by varying the network size from 100-nodes to 1000-nodes. It is observed that the percentage of the control traffic grows exponentially as the network size grows beyond 200-nodes. However, if the networks are split into sub-networks of 100 or 200-nodes, the total control traffic in the network remains below 8% of the total traffic in the network, which is acceptable in standard service-provider networks [99-101].

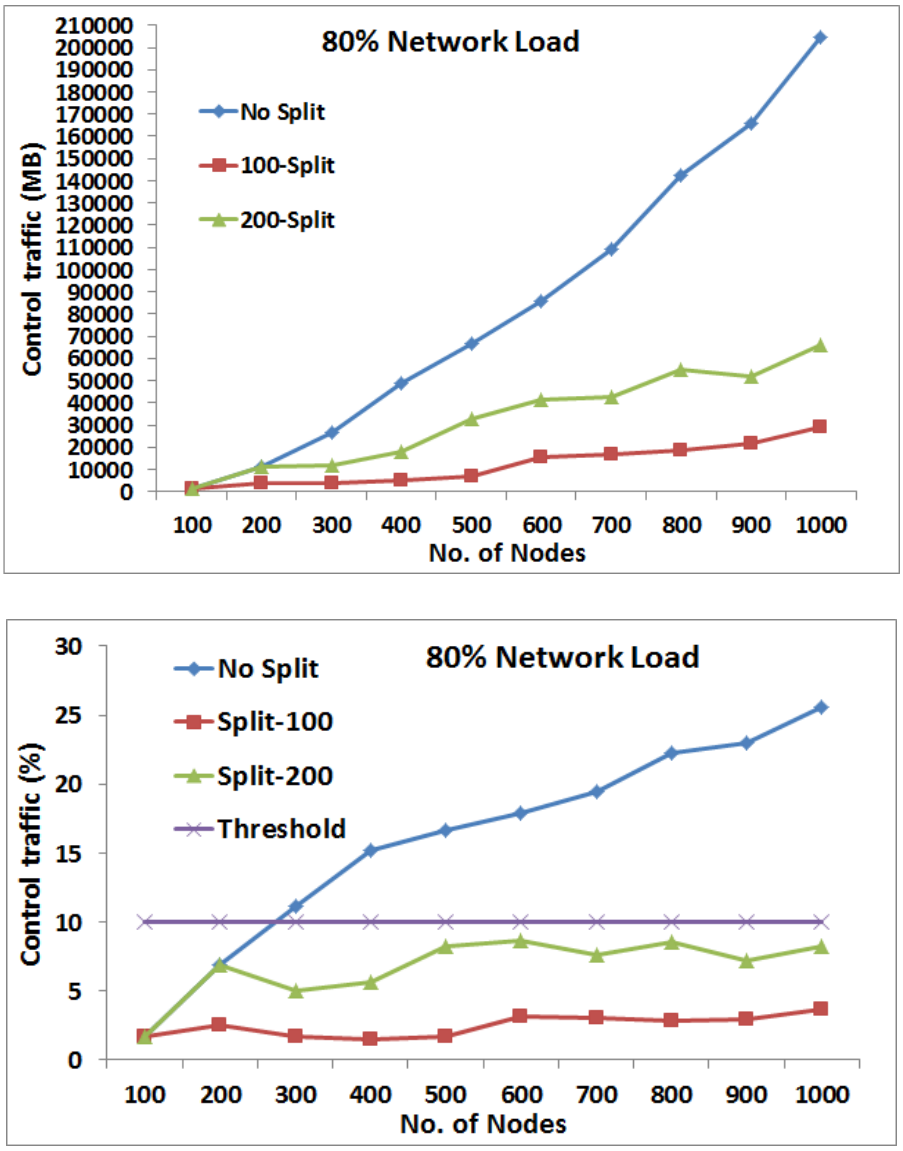


Fig. 5.10. Total control traffic and percentage of control traffic versus network size, before and after splitting the networks.

5.7 Conclusions

In this work, we have focused on engineering and architectural issues related to the design of a centralized control plane. Key characteristics of the centralized control plane, challenges in the centralized architecture of the network and its possible design approach using the Network Management System (NMS) have been discussed.

In addition, an analysis of the control traffic overhead in the network has been performed using a simulation model. Simulation results demonstrate that control traffic grows beyond a threshold at a certain point as the number of nodes and traffic load in the network increases. A scheme to divide the network into smaller sub-networks is proposed so that total control traffic is always below some threshold (some percentage of data traffic) and the control traffic overhead is within acceptable limits. The results from a simulation model have also been showcased. The results demonstrate that partitioning of the network significantly reduces the control traffic overhead in the managed networks.

An ILP formulation is presented to solve the problem for smaller networks (up to 50-node). We aim to minimize the total control traffic, the total controllers' implementation cost and the overall response time in the network. In addition, a heuristic approaches based on the *Random Greedy* strategy and *Weighted Random* strategy has been proposed to solve the problem for larger networks (up to 1000-nodes). The proposed heuristics runs within a reasonable computational time. The results obtained by implementing the heuristic algorithms are compared with the ILP. The results show that the heuristics compare favorably with the ILP without compromising on the quality of the solution. We also note that WRA heuristic outperforms RGA heuristic.

Chapter 6

Summary and Future Work

In this dissertation, we have focused on the analysis of high-speed Carrier Ethernet networks, especially, from the perspective of cost and performance and cost optimization. In the first problem, we have minimized the number of network identifiers in high-speed networks, since available identifiers are limited (4094 BVIDs in PBB-TE Networks). In addition to that, we have attempted to maximize the number of services that can be provisioned in core networks with the limited number of BVIDs, so that TCAM utilization of network equipment is less. We presented an ILP for optimal solution and solved the problem for larger instances by proposing four different heuristic algorithms. We then tried to optimize the cost of the contemporary high-speed networks so that internet-service providers can have the benefit of maximum profits by reducing the CAPEX and OPEX. Specifically we focus on reducing the total cost of the

network interfaces at different network layers satisfying all the traffic demands. A 3-layer (*IP+OTN+DWDM*) network capacity planning model is presented in this work to optimize the cost. The work presented in "multilayer optimization" can be extended to include dynamic traffic cases. Different machine learning techniques can be applied to gain better insights of traffic characteristics in contemporary service provider networks. In addition, the issue of fault detection and recovery as well as network survivability in multi-layered networks needs to be studied in depth. Since there are separate protocols running at every layer (for example, ARP at Ethernet level and OSPF, RIP at IP level), there is a need of optimized solution for fault detection and fault propagation across the various network layers.

In the latter part of the research, we have focused on the implementation of the "Centralized Control Plane" for contemporary service provider networks. We have focused on engineering and architectural issues related to the design of a centralized control plane. In addition, an analysis of the control traffic overhead in the network has been performed. We have proposed a scheme to divide larger networks into smaller sub-networks to reduce the control traffic overhead in the managed networks. Also, we solved the controller placement problem in divided networks using an ILP and proposed two different heuristic approaches. The work may be extended to accommodate the "stochastic optimization" model by identifying different states of the NMS. Control traffic analysis for each state of the NMS can be done separately for a more intuitive understanding of the nature of the control traffic in managed networks. Also, we have then tried to answer an interesting question—how does Carrier Ethernet perform for Virtual Machines (VM) migration in data-center and cloud environments? The question arises since VMs form the central processing entity in data-centers and are crucial to facilitating cloud computing environments. We have proposed the use of Carrier Ethernet in data-center/cloud environments. This work can be extended to include "load balancing" techniques for the VMs and to optimize resources like CPU, Memory and VM Migration Time and the others.

Due to the unprecedented growth in the Internet traffic, Application Service Providers (ASPs) need to adjust huge application driven traffic in already deployed

complex networks in short span of time [104]. Hence, there is a need of designing a standard centralized data plane that will allow Application Service Providers to implement service level traffic routing and management policies. The key motivation for further research in designing such abstraction is to standardize a set of service deployment protocols across a various types of applications deployed over the Internet [103, 104]. We believe that the work presented in this dissertation may be extended for further investigation in the following ways.

A. Carrier Ethernet and BVID allocation Problem

- Mathematical Model to determine the probability of the BVID Conflict occurring in the Core Networks.

B. Multilayer Optimization in High-Speed networks

- Extend the work to a journal paper with more sophisticated ILP.
- Implementation for 'Dynamic Traffic' case using Stochastic Optimization model.
- Apply 'Machine Learning' technics to study different traffic patterns.

C. Transport Technology Choices for Virtual Machines (VMs) in Data-Center and Cloud Environments

- Implement Load Balancing for the VMs and optimize resources like CPU, Memory and VM Migration Time.
- Apply 'Ant Colony Optimization' or other relevant technics for the same.
- Write a simulator for 'Omnipresent Ethernet' (OE), obtain the results for VM migration with the same and compare against PBB-TE and MPLS-TP.

D. Models, Algorithms and Solution Methods for Centralized Control Planes to Optimize Control Traffic Overhead

- Define different states for the NMS and determine the probability of NMS being in each of the state.
- Determine the Control Traffic at a given instance of a time using Stochastic Process.
- Consider the same problem for the ‘Un-capacitated’ & ‘Multiple-Allocation’ scenarios.

References

[1] B. Mukherjee, “Optical WDM Networks”, 2006, ISBN: 978-0-387-29055-3.

[2] IEEE Std 802.1Qay™-2009 “Amendment 10: Provider Backbone Bridge Traffic Engineering”.

[3] IETF RFC 5654, B. Niven-Jenkins, D. Brungard, M. Betts, N. Sprecher, S. Ueno, "Requirements of an MPLS Transport Profile", Sept. 2009.

[4] IETF RFC 3031, E. Rosen, A. Viswanathan, R. Callon, "Multiprotocol Label Switching Architecture", Jan. 2001.

[5] IEEE Std 802.1ah™-2008 “Amendment 7: Provider Backbone Bridges”.

[6] IEEE Std 802.1Q™-2005 “Virtual Bridged Local Area Networks”.

[7] IEEE Std 802.1ad™-2005 “Amendment 4: Provider Bridges”.

[8] D. Bhamare, A. Upadhyaya, S. Mehta, A. Kshirasagar, A. Gumaste, "The BVID Allocation Problem in 802.1Qay Provider Backbone Bridged Traffic Engineered Networks," Communications (ICC), 2011 IEEE International Conference on , vol., no., pp.1-6, 5-9 June 2011.

[9] A. Viswanathan, N. Feldman, Z. Wang, and R. Callon, “Evolution of multiprotocol label switching,” IEEE Commun. Mag., vol. 36, pp. 165–173, May 1998.

[10] IETF RFC 3032, E. Rosen, D. Tappan , G. Fedorkow , Y. Rekhter , D. Farinacci , T. Li, A. Conta, “MPLS Label Stack Encoding”, September 1998.

[11] IETF Internet Draft, R. Callon, P. Doolan, N. Feldman , A. Fredette, G. Swallow , A. Viswanathan , “A Framework for Multiprotocol Label Switching”, November 1997.

[12] R. Vaishampayan, A. Gumaste, S. Rana, N. Ghani, "Application Driven Comparison of T-MPLS/MPLS-TP and PBB-TE - Driver Choices for Carrier Ethernet," INFOCOM Workshops 2009, IEEE , vol., no., pp.1-6, 19-25 April 2009.

[13] S. Chiusano, F. Corno, P. Prinetto, M. Sonza, “Hybrid Symbolic-Explicit Techniques for the Graph Coloring Problem”, IEEE European Design and Test Conference, Paris (F), March 1997, pp. 422-426.

[14] F. Comellas and J. Ozón, “Graph coloring algorithms for assignment problems in radio networks,” in Applications of Neural Networks to Telecommunications 2, J. Alspector, R. Goodman, and T. X. Brown, Eds. Hillsdale, NJ: Lawrence Erlbaum Assoc., Inc., 1995, pp. 49–56.

[15] IETF RFC 3469, V. Sharma, F. Hellstrand, “Framework for Multi-Protocol Label Switching (MPLS)-based Recovery", Jan. 2001.

[16] R. Ramaswami and K. Sivarajan, “Optical Networks: A Practical Perspective”, 2009, ISBN: 978-0-12-374092-2.

[17] G. Srinivas, S. Vetrivel, N. Elango “ Applications of Graph theory in Computer Science an over view”, IJEST and Technology, Vol. 2(9), 2010, pp 4610 – 4621.

[18] P. Bottorff and P. Saltsidis, “Scaling provider ethernet,” IEEE Communications Magazine, vol. 46, no. 9, pp. 104–109, SEP 2008.

[19] R. Bhatia, M. Kodialam, T. Lakshman, “Fast network re-optimization schemes for MPLS and optical networks”, Computer Networks 50(3): 317-331 (2006).

[20] Abdul Kasim, “Delivering Carrier Ethernet: Extending Ethernet Beyond the LAN”, 2007, ISBN: 978-0-07-148747-4.

[21] W. Alanqar, A. Jukan, "Extending End-to-End Service Provisioning and Restoration in Carrier Networks: Opportunities, Issues, and Challenges,” IEEE Communications Magazine, pp. 52-60, January 2004.

[22] L. Caro, D. Papadimitriou, J. Marzo, "Improving Label Space Usage for Ethernet Label Switched Paths", ICC 08, IEEE International Conference on Communications, 2008, pp.5685-5691, 19-23 May 2008.
[23] S. Salam and A. Sajassi, "Provider Backbone Bridging and MPLS: Complementary Technologies for Next Generation Carrier Ethernet Transport," IEEE Commun. Mag., pp. 77–83, Mar. 2008.
[24] D. Fedyk and D. Allan, "Ethernet Data Plane Evolution for Provider Networks," IEEE Commun. Mag., Mar. 2008, pp. 84–89.
[25] D. Papadimitriou, "Routing protocols for carrier ethernet networks", 11th International Conference on Optical Networking Design and Modeling (ONDM), May 2007.
[26] S. Bhatnagar, S. Ganguly, and B. Nath, "Creating multipoint-to-point LSPs for traffic engineering", HPSR 2003, pages 201–207.
[27] W. Golab, R. Boutaba, "Path selection in user-controlled circuit-switched optical networks", Optical Switching and Networking, v.5 n.2-3, p.123-138, June, 2008.
[28] J. Wu, M. Savoie, H. Zhang, S. Campbell, G. v. Bochmann, and B. St. Arnaud, "Customer managed end-to-end lightpath provisioning," International Journal of Network Management, vol. 15, pp. 349–362, Sept./Oct. 2005.
[29] J. Holm, K. Lichtenberg, and M. Thorup, "Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity," Journal of the ACM, vol. 48, no. 4, pp. 723–760, July 2001.
[30] B. St. Arnaud and J. Wu, "Customer-controlled and managed optical networks," Journal of Lightwave Technology, vol. 21, pp. 2804–2810, Nov. 2003.
[31] M. Kodialam and T.Lakshman, "Minimum interference routing with applications to MPLS traffic engineering", INFOCOM 2000, pages 884–893.
[32] D. Applegate and M. Thorup, "Load optimal MPLS routing with N + M labels", INFOCOM 2003, IEEE, 2003.
[33] L. Fang, et. al, "The Evolution of Carrier Ethernet Services-Requirements and Deployment Case Studies" IEEE Commun. Mag., March 2008 Vol. 26 No. 3. pp 84-88. 9.
[34] F. Solano et al., "Label merging in all-optical label swapping networks", Proc. VI Workshop , G/MPLS networks, April 2007.
[35] M. Tafti, G. Mirjalily, S. Rajaee, "Topology design of Metro Ethernet Networks based on load balance criterion", IST 2008. International Symposium on , vol., no., pp.499,503, 27-28 Aug. 2008
[36] A. Gumaste and N. Krishnaswamy, "Proliferation of ITU 709 - Optical Transport Network (OTN) – A Use Case Based Study," IEEE Communications Magazine, September 2010.
[37] S. De Maesschalck, M. Pickavet, D. Colle, and P. Demeester, "Multilayer traffic grooming in networks with an IP/MPLS layer on top of a meshed optical layer," in Global Telecommunications Conference, vol. 7, December 2003, pp. 2750–2754.
[38] E. Palkopoulou, D. Schupke, and T. Bauschert. "Quantifying CAPEX savings of homing architectures enabled by future optical network equipment", Telecommunication Systems, pages 1–7, August 2011.
[39] R. Ranganathan, L. Blair, J. Berthold: "Architectural Implications of Core Grooming in a 46-Node USA Optical Network," OFC 2002, Anaheim, USA, March 2002.
[40] M. Scheffel, R.G. Prinz, C.G. Gruber, A. Autenrieth, D.A. Schupke, "Optimal Routing and Grooming for Multilayer Networks with Transponders and Muxponders," Global Telecommunications Conference, IEEE GLOBECOM '06. pp.1-6, Dec 2006.
[41] M. Ruiz, O. Pedrola, L. Velasco, D. Careglio, J. Fernández-Palacios, G. Junyent, "Survivable IP/MPLS-Over-WSON Multilayer Network Optimization," IEEE/OSA Journal of Optical Communications and Networking, , vol.3, no.8, pp.629-640, August 2011.
[42] R. Huelsermann, M. Gunkel, C. Meusburger, D. A. Schupke, "Cost modeling and evaluation of capital expenditures in optical multilayer networks," J. Opt. Netw. 7, 814-833 (2008).

[43] X. Cui, J. Wang, X. Yao, W. Liu, H. Xie, and Y. Li, “Optimization of multilayer restoration and routing in IP-over-WDM networks,” (OFC/NFOEC 2008), February 2008, pp. 1–10.
[44] E. Kubilinskas and M. Pi´oro, “Two design problems for the IP/MLPS over WDM networks,” in 5th International Workshop on Design and Reliable Communication Networks (DRCN 2005), October 2005, pp.241–248.
[45] S. Koo, G. Sahin, and S. Subramaniam, “Dynamic LSP routing in IP/MPLS over WDM networks,” IEEE Journal on Selected Areas in Communications, vol. 24(12), pp. 45–55, December 2006.
[46] A. Schmid-Egger and A. Kirstädter, "Ethernet in Core Networks – A Technical and Economical Analysis", in Workshop on High Performance Switching and Routing (HPSR), 2006.
[47] P. Chołda and A. Jajszczyk, “Recovery and its quality in multilayer networks,” J. Lightwave Technol., vol. 28, pp. 372–389, Feb. 2010.
[48] S. Gringeri, B. Basch, V. Shukla, R. Egorov, and T. J. Xia, "Flexible architectures for optical transport nodes and networks", IEEE Commun. Mag., vol. 48, no. 7, pp. 40–50, Jul. 2010.
[49] O. Gerstel and R. Ramaswami, "Optical layer survivability–An implementation perspective", IEEE J. Sel. Areas Commun.,vol. 18, no. 10, pp. 1885–1899, Oct. 2000.
[50] W. Bigos, B. Cousin, S. Gosselin, M. Le Foll, and H. Nakajima, “Survivable MPLS over optical transport networks: Cost and resource usage analysis,” IEEE Journal on Selected Areas in Communications, vol. 25(5), pp. 949–962, June 2007.
[51] P. Belotti, K. Kompella, and L. Noronh, “A comparison of OTN and MPLS networks under traffic uncertainty”, Working paper, 2011.
[52] I. Katib, D. Medhi, "A study on layer correlation effects through a multilayer network optimization problem," 23rd International Teletraffic Congress (ITC), 2011, vol., no., pp.31-38, 6-9 Sept. 2011.
[53] S. Sengupta, V. Kumar and D. Saha, "Switched optical backbone for cost-effective scalable core IP networks", IEEE Communications Magazine, 2003, 41(6).
[54] R. Krishnaswamy and K. Sivarajan, “Design of logical topologies: a linear formulation for wavelength-routed optical networks with no wavelength changers,” IEEE/ACM Tran. Networking, vol. 9, pp. 186–198, 2001.
[55] D.A. Schupke, "Multilayer and Multidomain Resilience in Optical Networks," Proceedings of the IEEE , vol.100, no.5, pp.1140-1148, May 2012.
[56] B. Fortz and M. Thorup, “Internet traffic engineering by optimizing OSPF weights,” Proc. IEEE INFOCOM, pp. 519–528, May 2000.
[57] D. Banerjee, and B. Mukherjee, "Wavelength-routed Optical Networks: Linear Formulation, Resource Budgeting Tradeoffs, and a Reconfiguration Study", IEEE/ACM Tran. Networking 8 (2000), 598–607.
[58] Online reference: http://networkx.github.com/
[59] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, “Live migration of virtual machines,” in Proc. USENIX Symposium on Networked Systems Design and Implementation (NSDI’05), Berkeley, CA, USA, pp. 273–286, 2005.
[60] S. Akoush, R. Sohan, A Rice, A. Moore, A. Hopper, "Predicting the Performance of Virtual Machine Migration," IEEE Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), Vol., No., pp.37-46, 17-19 2010.
[61] M. Fei, L. Feng, L. Zhen, "Live virtual machine migration based on improved pre-copy approach," IEEE International Conference Software Engineering and Service Sciences (ICSESS), Vol., No., pp.230-233 2010.
[62] IETF RFC 5317, "MPLS Architectural Considerations for a Transport Profile", Feb. 2009.
[63] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: enabling innovation in campus networks. SIGCOMM Comput. Commun. Rev. 38, 2 (March 2008), 69-74.

[64] A. Singh, M. Korupolu, D. Mohapatra, "Server-Storage Virtualization: Integration and Load Balancing in Data Centers", IEEE/ACM Supercomputing (SC) 2008..
[65] ITU-T Y.1731, “OAM functions and mechanisms for Ethernet based networks”, 2008.
[66] IEEE Std 802.1ag™-2007 “Connectivity Fault Management”.
[67] ONLINE: http://www.petri.co.il/managing-esxi4-with-vsphere-client.htm.
[68] R. Vaishampayan, S. Mehta and Ashwin Gumaste, "Omnipresent Ethernet: A Novel Metro Communication System using Binary + Source Routing and Carrier Ethernet," Post Deadline Paper, 25th IEEE OSA Optic Fiber Communications (OFC) conference, 2009, San Diego, USA, March 2009.
[69] A. Gumaste, et al, "Demonstration of Omnipresent Ethernet: A Novel Metro End-to-End Communication System using Binary+Source Routing and Carrier Ethernet," IEEE Journal of Lightwave Technology, 2010 March 2010.
[70] M. Hines and K. Gopalan, “Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning”, In Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments (VEE '09). ACM.
[71] ONLINE:http://social.msdn.microsoft.com/Forums/br/wpf/thread/197b933a-5c1e-4fe1-a372-3995e5e93ea9
[72] A. Greenberg, G. Hjalmtysson, D. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, H. Zhang, “A clean slate 4D approach to network control and management”, ACM SIGCOMM, Volume 35 Issue 5, October 2005, Pages 41 – 54.
[73] ONLINE: http://msdn.microsoft.com/en-us/library/dd193294%28v=versus100%29
[74] A. Gumaste, "Towards a 1-Microsecond Switch Router" Invited Paper, International Conference on Transparent Optical Networks, ICTON 2011, Stockholm, Sweden, June 2010.
[75] A. Chiu and J. Strand, “Control plane considerations for all-optical and multi-domain optical networks and their status in OIF and IETF,” Optical Networks Magazine, vol. 4, no. 1, pp. 26–35, 2003.
[76] M. Suchara, D. Xu, R. Doverspike, D. Johnson, J. Rexford, "Network architecture for joint failure recovery and traffic engineering", ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems, June 07-11, 2011, San Jose, California, USA.
[77] “Introduction to intelligent network (IN) capability set 1.” ITU-T Standard Q.1211.
[78] J. M. Smith and S. M. Nettles, “Active networking: One view of the past, present and future,” IEEE Transactions On Systems, Man and Cybernetics, vol. 34, pp. 4–18, Feb 2004.
[79] L. Peterson, Y. Gottlieb, M. Hibler, P. Tullmann, J. Lepreau, S. Schwab, H. Dandekar, A. Purtell, and J. Hartman, “A NodeOS interface for active networks,” IEEE J. Selected Areas in Communications, March 2001.
[80] T. V. Lakshman, T. Nandagopal, R. Ramjee, K. Sabnani, and T. Woo, “The SoftRouter architecture,” in Proc. ACM SIGCOMM Workshop on Hot Topics in Networking, November 2004.
[81] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker, “Onix: a distributed control platform for large-scale production networks”, In OSDI, 2010.
[82] L. Berger, Generalized Multi-Protocol Label Switching (GMPLS) Signaling Functional Description, 2003. RFC 3471.
[83] J. Lang, Link Management Protocol (LMP), draft-ietf-ccamp-lmp-10.txt, October 2003.
[84] A. Puliafito, and O. Tomarchio, “Using Mobile Agents to implement flexible management strategies”, Computer Communication Journal, 23(8): 708-719, April 2000.
[85] B. Quoitin, S. Uhlig, "Modeling the routing of an autonomous system with C-BGP," Network, IEEE , vol.19, no.6, pp. 12- 19, Nov.-Dec. 2005.
[86] A. Gumaste, A. Dhar and P. Gokhale, "On the State and Guiding Principles of Broadband in India," IEEE Communications Magazine, August 2009.

[87] N. Ghani, Q. Liu, A. Gumaste, J. Lankford, A. Shami, C. Assi, A. Khalil, D. Benhaddou, "Value-Added Services in Next-Generation SONET/SDH Networks," in IEEE Communications Magazine, Dec. 2008.
[88] A. Ernst, M. Krishnamoorthy, "Efficient algorithms for the uncapacitated single allocation p-hub median problem", Location Science, Volume 4, Issue 3, October 1996, Pages 139-154, ISSN 0966-8349, 10.1016/S0966-8349(96)00011-3.
[89] D. Skorin-Kapov, J. Skorin-Kapov, M. O'Kelly, "Tight linear programming relaxations of uncapacitated p-hub median problems", working paper, W. A. Harriman School for Management Policy, State University of New York at Stony Brook, Stony Brook, NY (to appear in European Journal of Operational Research), 1994.
[90] A. Ernst, M. Krishnamoorthy, "Exact and heuristic algorithms for the uncapacitated multiple allocation p-hub median problem", European Journal of Operational Research, Volume 104, Issue 1, 1 January 1998, Pages 100-112, ISSN 0377-2217, 10.1016/S0377-2217(96)00340-2.
[91] A. Ernst, M. Krishnamoorthy, "Solution algorithms for the capacitated single allocation hub location problem", Annals of Operations Research 86(1999)141–159.
[92] J. Klincewicz, "Hub location in backbone/tributary network design: a review", Location Science, Volume 6, Issues 1–4, May–December 1998, Pages 307-335, ISSN 0966-8349.
[93] A. Gumaste , D. Diwakar , A. Agrawal , A. Lodha , N. Ghani, "Light-mesh - A pragmatic optical access network architecture for IP-centric service oriented communication", Optical Switching and Networking, v.5 n.2-3, p.63-74, June, 2008, doi:10.1016/j.osn.2008.01.001.
[94] A. Gumaste, N. Ghani, P. Bafna, A. Lodha, A. Agrawal, T. Das, J. Wang; S. Zheng, "DynaSPOT: Dynamic Services Provisioned Optical Transport Test-Bed—Achieving Multirate Multiservice Dynamic Provisioning Using Strongly Connected Light-Trail (SLiT) Technology," Lightwave Technology, Journal of , vol.26, no.1, pp.183,195, Jan.1, 2008, doi: 10.1109/JLT.2007.913074.
[95] L. Kleinrock, F. Kamoun, "Optimal clustering structures for hierarchical topological design of large computer networks", Networks 10 (1980), 221-248.
[96] M. Kodialam and T.Lakshman, "Minimum interference routing with applications to MPLS traffic engineering", INFOCOM 2000, pages 884–893.
[97] D. Applegate and M. Thorup, "Load optimal MPLS routing with N + M labels", INFOCOM 2003, IEEE, 2003.
[98] N.Collins, R. Eglese and B. Golden, "Simulated annealing: An annotated bibliography", American Journal of Mathematical and Management Sciences 8(1988)209–307.
[99] S. White, "Concepts of scale in simulated annealing", Proceedings of the IEEE International Conference on Computer Design, November, 1984, pp. 646–651.
[100] M. Zukerman, T. Neame, and R. Addie, "Internet traffic modeling and future technology implications", In Proceedings of IEEE INFOCOM, 2003.
[101] K. Hyojoon and N. Feamster, "Improving network management with software defined networking", IEEE Communications Magazine 51.2 (2013): 114-119.
[102] Z. Arslan, A. Alemdaroglu, B. Canberk, "A traffic-aware controller design for next generation software defined networks," Communications and Networking (BlackSeaCom), 2013, vol., no., pp.167,171, 3-5 July 2013.
[103] R. Jain and S. Paul, "Network Virtualization and Software Defined Networking for Cloud Computing - A Survey," IEEE Communications Magazine, Nov 2013, pp. 24-31.
[104] J. Pan, S. Paul, and R. Jain, "A Survey of Research on Future Internet Architectures," IEEE Communications Magazine, Vol. 49, No. 7, July 2011, pp. 26-36.
[105] D. Bhamare, A. Gumaste, M. Krishnamoorthy, N. Dayama, "On the Backbone VLAN Identifier (BVID) Allocation in 802.1Qay Provider Backbone Bridged — Traffic Engineered Networks", IEEE Transactions on Network and Service Management, vol.PP, no.99, pp.1-16.

[106] D. Bhamare, A. Gumaste, P. Srivastava, M. Krishnamoorthy, "Multi-layer optimization for service provider transport networks", Local Computer Networks (LCN), 2013 IEEE 38th Conference, vol. no., pp.695,698, 21-24 Oct. 2013.

[107] A. Shankar, D. Bhamare, M. Krishnamoorthy, A. Gumaste, "Optimizing transport technology choices for Virtual Machines (VMs)in data-center and cloud environments", Communications (NCC), 2013 National Conference, vol. no., pp.1,5, 15-17 Feb. 2013.

[108] I. Katib, D. Medhi, "IP/MPLS-over-OTN-over-DWDM Multilayer Networks: An Integrated Three-Layer Capacity Optimization Model, a Heuristic, and a Study," IEEE Transactions on Network and Service Management, vol.9, no.3, pp.240,253, September 2012.

List of Publications

Accepted Journal Paper:

1. **D. Bhamare**, A. Gumaste, M. Krishnamoorthy and N. Dayama, "On the Backbone VLAN Identifier (BVID) Allocation in 802.1Qay Provider Backbone Bridged—Traffic Engineered Networks" submitted to IEEE Transactions on Network and Service Management.
   Submitted on: 11-Dec-2012
   First Revision: 27-Jun-2013
   Accepted on: 09-Nov-2013

Under Preparation for submission:

2. Title: "Models, Algorithms and Solution Methods for Centralized Control Planes to Optimize Control Traffic Overhead."
   Target Journal: "Computer Networks, the International Journal of Computer and Telecommunications Networking, Elsevier."

Accepted Conference Papers:

3. **D. Bhamare**, A. Upadhayaya, S. Mehta, A. Kshirasagar and A. Gumaste, "The BVID Allocation Problem in 802.1Qay Provider Backbone Bridged Traffic Engineered Networks" IEEE International Conference on Communications ICC 2011, Kyoto Japan June 2011.
4. A. Gumaste, S. Bidkar, C. Taunk and **D. Bhamare**, "Using MPLS-TP for Data-Center Interconnection" Invited paper, Broadnets 2010, Athens Greece Oct 2011.
5. C. Vaishampayan, S. Bidkar, S. Mehta, **D. Bhamare** and A. Gumaste. "Demonstrating OpenFlow over a Carrier Ethernet Switch Router (CESR)—A Services Perspective" (Invited), NOC 2012.

6. S. Bidkar, S. Mehta, **D. Bhamare**, N. Bajaj, A. Medhekar and A. Gumaste "Circuit Performance in a Packet Network: Demonstrating Integrated Carrier Ethernet Switch Router (CESR)+Optical Transport Network (OTN)" ICNC 2013.
7. A. Shankar, **D. Bhamare**, M. Krishnamoorthy and A. Gumaste, "Optimizing Transport Technology Choices for Virtual Machines (VMs) in Data-Center and Cloud Environments" NCC 2013.
8. **D. Bhamare**, A. Gumaste, P. Srivastava and M. Kishnamoorthy, "Multi-Layer Optimization for Service Provider Transport Networks", LCN 2013.
9. **D. Bhamare**, S. Bidkar and A. Gumaste, "Is Carrier Ethernet = Software Defined Networks?", Invited paper, IWON, 2013.

Patent:

10. "Method and apparatus for allocating backbone VLAN identifiers"

Publication number:	US20130028141 A1
Publication type:	Application
Application number:	US 13/245,867
Filing date:	Sep 27, 2011
Priority date:	Jul 29, 2011
Inventors:	**Deval A. Bhamare**, Ashwin Gumaste
Original Assignee:	Indian Institute of Technology Bombay

Acknowledgements

I would like to express my sincere gratitude to my IIT Bombay supervisor, Professor Ashwin Gumaste and my Monash supervisor, Professor Mohan Krishnamoorthy for their constant motivation and support in this work. My supervisors have always been willing to take time out of their busy schedules to guide me whenever I needed guidance or came across various challenges. Without their support, this work would not have been possible.

I would like to thank my RPC members, especially, Professor Ganesh Ramakrishnan and Professor Emanuele Viterbo for their valuable insights and encouragement during the course of this thesis. I would like, especially, to thank all my colleagues at Gigabit Networking Lab, CSE, IIT Bombay and at IITB-Monash Academy for their support, discussions and collaboration. They helped in creating a rich lab-atmosphere that was intellectual and fun-filled, leading to productive research. Also, I would like to thank the staff at IIT Bombay and IITB-Monash Academy for their constant support and help.

Last, but not the least, I would like to thank my family for their unconditional support and encouragement to pursue my interests and for always believing in me. To them I am indebted for life, and to the God, for eternity.

Deval A. Bhamare

April 2014