SOLUTIONS TO CONSTRAINED PATH COMPUTATION
IN MULTI-LAYER NETWORKS

by

Shujia Gong

A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Electrical and Computer Engineering

Committee:

Dr. Bijan Jabbari, Dissertation Director

Dr. Shih-Chun Chang, Committee Member

Dr. Brian L. Mark, Committee Member

Dr. Robert Simon, Committee Member

Dr. Andre Manitius, Department Chairman

Dr. Lloyd J. Griffiths, Dean, The Volgenau School of Information Technology and Engineering

Date: 10/19/2007

Fall Semester 2007
George Mason University
Fairfax, VA

Solutions to Constrained Path Computation in Multi-layer Networks

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

By

Shujia Gong
Master of Science
George Mason University, 2003
Bachelor of Science
Beijing University of Posts and Telecommunications, 1993

Director: Dr. Bijan Jabbari, Professor
Department of Electrical and Computer Engineering

Fall Semester 2007
George Mason University
Fairfax, VA

Dedication

I dedicate this dissertation to my wife, Jinglu Xu and my parents Zhonghe Gong and Qiuying Wang for their caring and support.

Acknowledgments

This dissertation marks the end of my Ph.D. study in the Department of Electrical and Computer Engineering (ECE) of George Mason University. Life is full of uncertainties. In July, 1999, the Chinese Communist regime launched the brutal persecution of *Falun Gong* (*http://www.falundafa.org*), a peaceful meditation practice with the principles of *Truthfulness, Compassion and Tolerance*. Faced with the defamation and demonization of my practice by the state propaganda machine and possible detention for the expression of my belief, I decided to leave China. Studying abroad was the only quick and feasible way. I was fortunately admitted to George Mason University. The admission notice came at the same time as the message that my mother was thrown into a forced-labor camp without trial for her peaceful appeal in Tiananmen Square for the end of the persecution.

In August, 2000, I came to the United States with my wife, but without any financial aid.

That was the hardest period in my life. I was fortunate to meet Dr. Shih-Chun Chang. With his recommendation, on the second day of the new semester, I started working as a TA in the ECE department. After the first mid-term exam, Dr. Chang recommended me to Dr. Bijan Jabbari. Dr. Jabbari became my mentor and has been helping me both academically and professionally. It was the GRA position he offered me that helped me to support my parents, who came to the United States in 2002, after a long period of physical and mental torture in the detention center and the labor camp.

I would like to thank all the professors at George Mason University who helped me in my academic development. I would especially like to thank Dr. Robert Simon and Dr. Brian L. Mark for sitting on my dissertation committee.

I would like to thank Dr. Christopher St. Jean and Dr. Payam Torab, who worked with me in our Communications and Networking Laboratory (CNL), for the inspiring discussion between us on various technical issues.

I would like to thank Mr. Randy Anderson, Mr. Derek Kan and Mr. Benjamin Allen, who sponsored my dissertation work on the Dynamic Resource Allocation in GMPLS Optical Networks (DRAGON) project, and Ms. Qian Xu, who coded a part of my algorithm for production use.

Last but not least, I am also very grateful to those who protected and encouraged me when I was in China. For their safety, I cannot reveal their names, but without their help, I might not have come to the United States.

There are many people who helped me during this long journey. I am unable to list their names individually, but I want to extend my heartfelt thanks and to wish each and every one of them a very bright future.

Table of Contents

List of Tables

List of Figures

Abstract

SOLUTIONS TO CONSTRAINED PATH COMPUTATION IN MULTI-LAYER NETWORKS

Shujia Gong, PhD

George Mason University, 2007

Dissertation Director: Dr. Bijan Jabbari

Traffic Engineering methods as applied to traditional IP networks rely on link attributes advertised by link state protocols, such as Open Shortest Path First (OSPF) or Intermediate System to Intermediate System (IS-IS). Extending link state protocols to include heterogeneous transport layer attributes brings a more comprehensive view of networks for path computation. A unified control plane, which enables horizontal cooperation between peer layers and vertical integration across layers, facilitates the optimization of network resource and instantiation of cross-layered paths, yet brings to path computation additional challenges. These include but are not limited to *Generalized Label Continuity Constraints*, such as wavelength continuity and VLAN (*Virtual Local Area Network*) tag continuity and *Interface Specific Adaptation Constraints* such as switching type adaptation constraints when a cross-layered path needs to be setup. These constraints cannot be satisfied by traditional CSPF (*Constrained Shortest Path First*) or integer linear programming. Moreover, the network graph may not be enough to describe the connectivity of network resources associated with wavelength, VLAN tag or switching type adaptation capabilities.

Furthermore, the dynamic nature of the networks makes all exhaustive search or other NP-hard algorithms practically unattractive.

In this dissertation, we provide the *Common Vector* solution to the *Generalized Label Continuity Constraints*. Mathematical analysis and simulation results demonstrate that the algorithm addresses the scalability problem of the existing wavelength graph solution, yet only with minor performance degradation from blocking perspective when the traffic load is not high. Especially, when the label space grows fast, the blocking caused by the lack of common labels is further reduced. Link performance bounds in a ring topology, which can help evaluate the performance degradation of common vector solution more accurately, is also discussed.

For *Interface Specific Adaptation Constraints*, we provide the *Channel Graph* solution, which transforms the network graph to channel graph. We prove that this solution addresses both the optimality and scalability problems of path computation in multi-layer networks. We also prove that with assumption that the connectivity and cost of adaptation depends on switching type associated with an interface, the *Channel Graph* solution is most efficient. In a sparse network, the *Channel Graph* solution has the same order of computational complexity as running CSPF on network graph.

Simulation results that corroborate those from the analytical models are presented in this dissertation. The solutions to path computation, as discussed here, lend themselves as good candidates for Internet of future. The proposed solutions for switching type adaptation and VLAN tag have also been implemented and verified in practice [1].

[1]This is done as a part of path computation in Dynamic Resource Allocation in GMPLS Optical Networks (*DRAGON*) project, an NSF sponsored project, to create dynamic, deterministic, and manageable end-to-end network transport services for high-end e-Science applications.

Chapter 1: Introduction

High-end and e-Science applications such as particle physics, earth observation, bioinformatics and electronic very-long baseline interferometry (e-VLBI) generate terabyte and petabyte data and demand dynamic, deterministic, and manageable end-to-end network transport services. In dealing with these requirements, networks become more heterogeneous, connection-oriented and include various network elements and switching technologies.

The switching technologies include but are not limited to layer 1 switching such as wavelength (lambda), waveband and fiber switching; layer 2 switching such as Ethernet and cell switching; sub-layer 3 or label switching such as Multi-Protocol Label Switching (MPLS) [1], and layer 3 switching such as packet (e.g., Internet Protocol, IP) switching. All the above switching types of layer 1 may be modeled in space and time, while all switching types above layer 1 may be viewed as packet filtering. Nonetheless, in general a realization of these switching types within a network requires consideration of the specific type of switching on a per port basis at each node, hence adding a level of complexity to the network for multi-layer switching.

Along with the heterogeneity of the transport and switching technologies, the unification of control plane is under fast development, such as the set of standards defined under Generalized MPLS (GMPLS) [2] in the The Internet Engineering Task Force (IETF), to provide a high degree of flexibility and economy of network resource

utilization. Therefore, GMPLS has the potential to become an integral part of Internet core networks [3], [4] by providing end-to-end control, provisioning, protection and restoration in heterogeneous transport networks. Because the transport networks are already incorporating to a degree network automation and self-actualization, it is indeed needed to simplify bandwidth procurement, provisioning and management. Due to close interaction between these functions in transport network and packet networks to establish a traffic engineering (TE) path, a common control plane becomes increasingly an attractive proposition.

Link state protocols such as OSPF has been enhanced to provide all the resource availability information of a TE-link[5]. Therefore a single Traffic Engineering Database (TED) that integrates the latest topological and network state information is available for the Path Computation Element (PCE) [6] to search an optimal end-to-end path efficiently. Reference [5, 7–9] provide the signaling definition on the resource reservation and Label Switched Path (LSP) establishment and tear-down.

Resource optimization and performance optimization are thus possible, yet with many challenges. These challenges cannot be met with traditional shortest path searching algorithm. Network graph must undergo some transformation process before a path can be searched.

Early works include research on the end-to-end transport and path computations architecture such as [10] and [11]. However, path computation algorithms are needed in order to operate the network efficiently, cost-effectively and reliably.

In this chapter, we will first discuss the existing path computation methods, followed by the discussion of challenges to path computation when the networks become more heterogeneous.

1.1 Shortest Path Computation by Integer Linear Programming and Matrix Calculation

Path computation in data networks has been extensively studied. Algorithms to search shortest path, shortest pair of vertex-disjoint or edge-disjoint paths are provided in [12], where edge-disjoint and vertex-disjoint paths also involve transformation of the network graph by adding links with negative metric to the network graph. Although these algorithms are essential for the path protection and restoration, they are not applicable to path computation in a cross-layered network environment.

1.1.1 Integer Linear Programming Approach to Find Shortest Path

Reference [13] presents an approach to search shortest path through Integer Linear Programming, where shortest path problem is equivalent to the transshipment problem. The source node will be the supply point, the destination node will be the demand point, and the number of supplied units is one.

Suppose there are n nodes in a graph $G = \langle V, E \rangle$, where N is the node set and E is the arc set. We denote source node as s, destination node as d, and c_{ij} as the cost between node i and j, and

$$c_{ij} = \begin{cases} \text{metric of arc(i,j)} & \text{if there is a link between node } i \text{ and node } j \text{ ;} \\ 0 & \text{if } i = j; \\ \infty & \text{otherwise.} \end{cases} \tag{1.1}$$

We define x_{ij} as variables to be solved and

$$x_{ij} = \begin{cases} 1 & \text{if arc(i,j) is taken by the shortest path ;} \\ 0 & \text{otherwise.} \end{cases} \tag{1.2}$$

Then the general formulation is:

$$\begin{aligned} &\min \sum_{i=1}^{n}\sum_{j=1}^{n}(c_{ij}x_{ij}) \text{ for } c_{ij} \neq 0 \text{ and } c_{ij} \neq \infty; \\ &\quad \text{such that } \sum_{j=1}^{n}(x_{sj}) = 1; \\ &\quad \sum_{j=1}^{n}(x_{jd}) = 1; \\ &\quad \sum_{i=1}^{n} x_{ij} = \sum_{k=1}^{n} x_{jk} \qquad \text{for } j \neq s \text{ and } j \neq d; \\ &\quad 0 \leq x_{ij} \leq 1. \end{aligned} \tag{1.3}$$

By solving (1.3) with an integer linear programming solver, the shortest path can be obtained.

Integer linear programming can also be used to solve max-flow problem and critical path method. However, path computation with Generalized Label Continuity Constraints or Interface Specific Adaptation Constraints is non-linear [14].

1.1.2 Searching Shortest Path through Matrix Calculation

Reference [15] provides the matrix algorithms to search shortest path. This algorithm is an implementation of Dijkstra's algorithm through matrix operation. Given a graph

$G = \langle N, E \rangle$, matrix $L = (l_{ij})$ is defined as:

$$l_{ij} = \begin{cases} \text{length of arc(i,j)} & \text{if } \langle i, j \rangle \in E \ ; \\ 0 & \text{if } i = j; \\ \infty & \text{otherwise.} \end{cases} \tag{1.4}$$

We initialize $L^{(0)} = L$, and define

$$l_{ij}^{(k)} = \min(l_{ij}^{(k-1)}, l_{ik}^{(k-1)} + l_{kj}^{(k-1)}). \tag{1.5}$$

For a graph with n nodes, $L^{(n)}$ gives the distance of shortest path.

1.2 Multi-Layer Networks and Challenges to Traffic Engineering

Given the GMPLS control plane, we can setup lightpath to build a virtual topology embedded on a physical topology. Reference [16] formulate the virtual topology design problem to optimize either the packet delay for a given traffic demand matrix or minimize the maxflow on any lightpath. However, this problem is NP-hard and therefore, reference [16] gave the heuristics approach. Reference [14] also present an exact integer linear programming approach to design the virtual topology with minimizing the average packet hop distance as the objective function. If wavelength continuity constraint is not relaxed, the optimal solution of the objective function in [14] is non-linear and also NP-hard.

Building virtual topology is essentially to decompose the path computation to overlay model on a per layer basis, which means an upper layer routes is determined

by Forwarding Adjacency (FA) LSPs instantiated in a lower layer.

With the introduction of Multi-Region and Multi-Layer networks (MRN/MLN) in [17], a path can be setup end-to-end across different layers. Here, a region is defined as a switching technology domain and MRN is defined as a network of multiple switching types. MRN/MLNs bring in the switching capability constraint. In MRN/MLN, multiple switching technologies coexist and an LSP will traverse networks with different switching capabilities. Packet Switch Capable region (PSC), Layer-2 Switch Capable region (L2SC), Time Division Multiplexing capable region (TDM), Lambda Switch Capable (LSC), and Fiber Switch Capable (FSC) have so far been defined. All the resource availability information is integrated into a single Traffic Engineering Database (TED) and GMPLS provides a comprehensive framework to control the cross-layered Label Switched Path (LSP) setup through vertical and horizontal interaction and integration in MRN/MLN [18]. Therefore, efficient and optimal path computation across the whole MRN/MLN is enabled.

MRN/MLN may consist of single-switching-type-capable Label Switching Routers (LSR) and multi-switching-type-capable LSRs, as defined in [17]. Simplex and hybrid nodes are two types of multi-switching-type-capable LSRs, where simplex node is defined as a network element with different switching capabilities, but each interface has only one switching capability, while hybrid node means that one interface of a network element has multiple switching capabilities, and thus can adapt between different switching types.

Cross-layered search of path and the nature of the optical network add new constraints into the PCE. We need to consider horizontal interaction constraints such as wavelength continuity and also vertical integration constraints such as switching type

adaptation constraints between different layers. Those constraints add new challenges to the path computation algorithm and may need certain form of transformation of the network graph before path computation.

Scalability of the solution is also an important aspect we need to consider. Exhaustive search or non-polynomial algorithm cannot be accepted for dynamic requests.

In this dissertation, we consider multi-layer networks and focus on constraints in traffic engineering path computation and solutions to support them. In chapter 2, we discuss the taxonomy of constraints in multi-layer networks for divide-and-conquer purpose and review the existing solutions to these constraints. In chapter 3, we discuss the solution to Generalized Label Continuity Constraint (GLCC) such as wavelength continuity and VLAN tag continuity, together with analytical model of common vector solutions and simulation results. In chapter 4, we discuss the Interface Specific Adaptation Constraints (ISAC), such as Switching Capability, encoding type and bandwidth granularity constraints, together with the Channel Graph solution. In chapter 5, we provide an analytical model for link blocking probability estimation in a ring network with homogeneous traffic matrix, which can provide better estimation of network performance for the common vector solution. Finally, we provide some concluding remarks in chapter 6.

Chapter 2: Review of Constraints and Solutions in Multi-layer Networks

Many constraints, such as priority attributes, preemption attributes, policing attributes, bandwidth requirements, exist in the traditional IP networks[19]. Multi-layer networks such as Hybrid Optical/IP Networks introduce additional constraints that cannot be satisfied in a straightforward approach[20].

As shown in Figure 2.1, we summarize the constraints into two areas, i.e., prunable and non-prunable constraints.

Prunable metric means that the solution to such constraints is a simple filtering. All the network elements that do not have required features will be pruned before a path searching process. Bandwidth requirement is an example of prunable constraints.

Non-prunable constraints mean that certain network resources, such as a TE-link, should not be pruned, but whether it is useable depends on the parameters of the whole determined path. These constraints include the following two categories: additive constraints and non-additive constraints. Examples of additive constraints are attenuation and dispersion request of an optical signal. Wavelength continuity and its more general form, referred to as label continuity, is an example of non-additive constraints. Switching type adaptation is another example of non-prunable and non-additive constraint.

Switching type adaptation constraint is different from the label continuity constraint in nature. In MLN, some network elements are multi-switching-type-capable

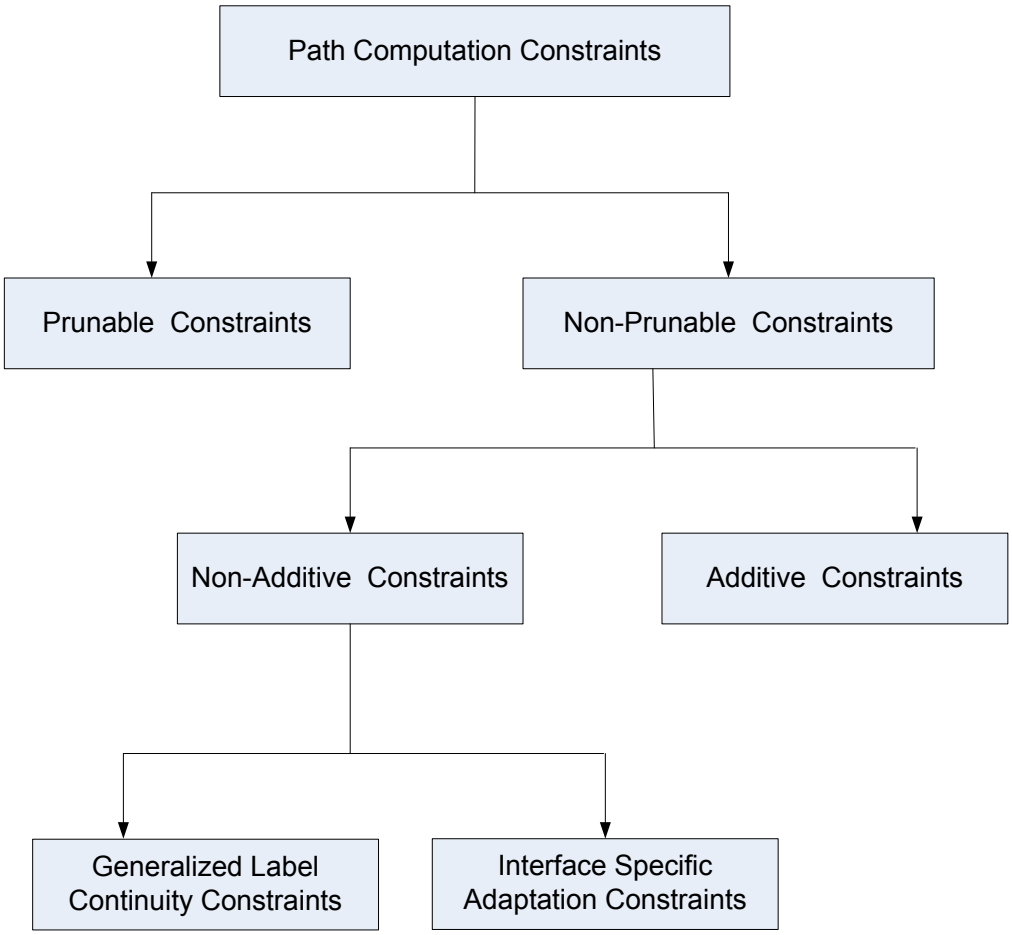

Figure 2.1: Taxonomy of Constraints in Multi-layer Networks

and they can adapt from one switching type to another. Unlike the wavelength conversion that can generally translate any incoming wavelength on any incoming fiber to any outgoing wavelength on any other fiber, adaptation function of multi-switching-type-capable LSRs is interface specific and generally, the adaptation cannot be done from any particular switching type to any other particular type on an interface. Therefore, this type of constraint is named as Interface Specific Adaptation Constraint (ISAC).

We will discuss all the above mentioned constraints in the following sections.

2.1 Prunable Constraints and Constrained Shortest Path First Solution

Prunable constraints include but are not limited to bandwidth requirement, policy constraint or protection requirement.

Some administrative policy may require that certain network resources should not be used for path setup. Shared Risk Link Group (SRLG) can also be a prunable constraints in such a way that the nodes and links of the same SRLG should not be used when searching disjoint paths.

Reference [21] discusses a disjoint path selection scheme with SRLG, in which link attributes associated with all layers must be taken into account. For example, two optical paths, which are treated as different IP links in IP layer, may share the same fiber using different wavelengths. We say that the two IP links belong to the same SRLG because a failure at the fiber will cause both IP links to fail at the same time. Therefore, when we compute node/link disjoint paths, SRLG, which includes the lower-layer information, must also be considered.

A solution to prunable constraints is known as CSFP (Constrained Shortest Path First), which is an extension of Dijkstra's algorithm. Link state protocol such as OSPF and IS-IS can advertise the resource availability, such as residual bandwidth information of a link, based on which PCE can build the TED. When an LSP setup request with the bandwidth requirement is made, the PCE can prune the links without sufficient bandwidth before running Dijkstra algorithm. CSPF can also be used to solve policy constraints or SRLG constraints, i.e. to pruned the restricted resource before path searching.

2.2 Non-prunable Constraints and the Solutions

Not every network element in an optical network provides O-E-O conversion along the lightpath. In such a network, Routing and Wavelength Assignment (RWA) is subject to many constraints categorized either as additive, or as non-additive. Certain constraints, such as wavelength continuity, switching capability and bandwidth requirements, are non-additive, while constraints such as attenuation and dispersion are additive. These constraints make the lightpath setup in an optical network very different from circuit setup in a traditional circuit-switched network.

2.2.1 Additive Constraints and the Solutions

In this subsection, we discuss the additive constraints such as attenuation and dispersion, followed by the discussion on K Shortest Paths (KSP) solutions.

Additive Constraints

In DWDM networks, with the increase of bit-rate and the number of wavelength, and with the reduction of the channel spacing, optical impairments have a more significant impact on the routing scheme. Reference [22] and [23] discuss the optical transmission technologies and the impairments systematically.

Attenuation is defined as decrease on signal strength caused by absorption and scattering. It is usually expressed in db/km. The impurity of fiber will convert optical signal to heat, which cause the absorption attenuation. Scattering means that the light ray may change its direction and diffuse out of the fiber. There are two types of scattering, namely Rayleigh scattering and Mie Scattering. Rayleigh Scattering is caused by miniscule changes in the core's refractive index and Mie scattering is resulted from the core that is not a perfect cylinder. Attenuation occurs not only during transmission, but also during switching because some switching fabric in OXC consists Micro-Electro-Mechanical Systems (MEMS), which is a passive device. Every deflection will cause loss of signal strength.

Solution to attenuation is amplification. However, signal-to-noise ratio (SNR) decreases because Erbium-Doped Fiber Amplifier (EDFA) adds Amplified Spontaneous Emission (ASE) to noise.

Dispersion is caused by different speeds of different wavelength during light ray propagation. The main reason is either by modal effect (PMD) or chromatic dispersion. PMD (polarization mode dispersion) is an inherent property of all optical media. Dispersion is measured in $ps/nm/km$.

With the reduction of signal spacing, inter-channel cross-talk will also happen.

Other non-linear effects such as Self-phase Modulation (SPM), Cross-phase Modulation (XPM) caused by non-linear index of refraction of glass, and Four Wave Mixing (FWM) which generates a new wavelength by mixing multiple wavelengths.

Constraints on attenuation, dispersion and delay are additive. Optimization of all the additive constraints at the same time is an NP-hard problem.

K Shortest Paths solution

K Shortest Paths (KSP) is a solution to additive constraints, such as attenuation, dispersion and delay. We cannot delete any intermediate results even if a threshold of a constraint is exceeded because we can hardly determine which segment of the path caused that problem and meanwhile that segment could appear in another path that satisfies all the constraints. We need to find K shortest paths, add corresponding link parameters and choose the qualified one.

Numerous publications have discussed KSP, computing K shortest paths in the order of increasing length. Reference [24–26] are three among many KSP algorithms. The algorithms provided in [24] and [25] allow cycles. Reference [25] provides a straightforward solution, *Recursive Enumeration Algorithm (REA)*, which does not delete any link or node to avoid cycles.

Recursive Enumeration Algorithm Define a digraph $G = \langle V, E \rangle$, where V is the node set and E is the arc set. It is easy to find the shortest path from node s to t and any other node. We define $\Gamma^{-1}(v)$ as the node set, such that for every element $u \in \Gamma^{-1}(v)$, arc $\langle u, v \rangle \in E$.

Candidate path set for k^{th} shortest path to node v is denoted as $C^k(v)$. Clearly,

the second shortest path from s to t must be one of the shortest path from s to a node v in $\Gamma^{-1}(t)$, followed by arc $\langle v, t\rangle$. Therefore, all the shortest path from s to any node in $\Gamma^{-1}(t)$ will be in $C^2(t)$. The path of the smallest length and with $\langle u, t\rangle$ as the last hop is denoted as $\pi^2(t)$. Clearly, $\pi^2(u) + \langle u, t\rangle$ with the paths remaining in $C^2(t)$ will together consists $C^3(t)$. The procedure to find $\pi^2(u)$ is the same as that to find $\pi^2(t)$. This is why this algorithm is recursive.

If we continue the above procedure, we can either find K shortest path or exhaust all possible paths from s to t.

Simply put, the next shortest path $\pi^k(t)$ is found and deleted from the candidate path heap, and the next shortest path to the predecessor u of t in $\pi^k(t)$ should be found and concatenated by $\langle u, t\rangle$. The newly found path is added to the candidate path heap, from which the next shortest path $\pi^{k+1}(t)$ is found and deleted.

The recursion happened when we search the next shortest path to the predecessor u of t in $\pi^k(t)$. This algorithm is simple and efficient. Because it does not avoid loops by deleting nodes or links, its complexity is $O(m + Kn\log(m/n))$, where m and n are the number of arcs and nodes, respectively.

YEN's algorithm YEN's algorithm searches loopless K Shortest Paths. A new implementation of YEN's algorithm is presented in [26]. YEN's algorithm cannot be applied directly to routing in MLN because it will produce unnecessary loops. Before we discuss in details, we reiterate some terminologies of YEN's algorithm as follows.

YEN's algorithm is based on a deviation algorithm, which constructs a "pseudo"-tree. This tree is not a usually defined tree because it contains repeated nodes. The "pseudo"-tree, whose root is the source node, is initialized to be the shortest path.

This path, denoted as p_1, will be the parent path for the next shortest path p_2.

The basic idea is that any newly found path must share some nodes that are along the existing "pseudo"-tree from the root. The last node that newly found path shares with the existing tree is called "deviation node", denoted as $d(p_k)$ if we are searching for path $k+1$. When searching path $k+1$, we should not repeat the nodes that had appeared from the root to $d(p_k)$. Therefore, these nodes should be deleted. We should also avoid the arc from the deviation node to its next node, and delete those nodes and arcs that had been deleted when p_k was found.

We should regard the nodes from the $d(p_k)$ to the predecessor of the terminal node along path p_k, as potential deviation node for $d(p_{k+1})$. Delete the nodes from the source node to potential $d(p_{k+1})$ along path p_k, and the arc from $d(p_{k+1})$ to its next node along path p_k, and run SPF algorithm to find shortest path q from $d(p_{k+1})$ to the terminal node. Concatenate the path from the source to the potential $d(p_{k+1})$ along path p_k with q to form the next potential shortest path and add it to candidate path heap. After searching along the parent path p_k is finished, choose the path with minimal cost from the candidate path heap.

YEN's algorithm is more complicated than REA. Because it needs to delete nodes and arcs before the next shortest path is found, its efficiency is worse than REA. However, the paths we can find are loopless. The complexity of YEN's algorithm, according to [26], is $O(Kn(m+nlogn))$, where n is the number of nodes and m is the number of arcs in the network graph.

2.2.2 Non-additive Constraints

In this subsection we discuss the non-additive constraints. These include constraints such as label continuity (for example wavelength continuity in all-optical networks) and switching capability.

Generalized Label Continuity Constraint and Auxiliary Graph Solution

Label continuity is important at least in two cases. First, wavelength continuity which is a non-additive constraint in all-optical networks where wavelength translation is not present at every or certain nodes across a path. For nodes without wavelength translation capability, the incoming wavelength and outgoing wavelength along the same path should be the same. Another case is the problem of global VLAN tag continuity, i.e., a path should maintain its VLAN tag at the nodes across a path. In such networks, the label swapping is not available at every or certain nodes.

This constraint is considered to be non-prunable because we cannot delete those links without the required wavelength. Such links can also be part of the lightpath after wavelength translation occurs in some intermediate nodes. We cannot prune the nodes or links according to VLAN tag availability either because we cannot determine which VLAN tag could be selected at the originating node.

An approach to solve the wavelength continuity constraint by converting a Network Graph to a Wavelength Graph has been presented in [27]. In wavelength graph, one plane is generated for each wavelength, and each node in the network graph is duplicated at each wavelength plane. For those nodes that have wavelength conversion capability, additional links are created to connect the replications of each node on the corresponding wavelength planes. Virtual nodes are generated as a dummy

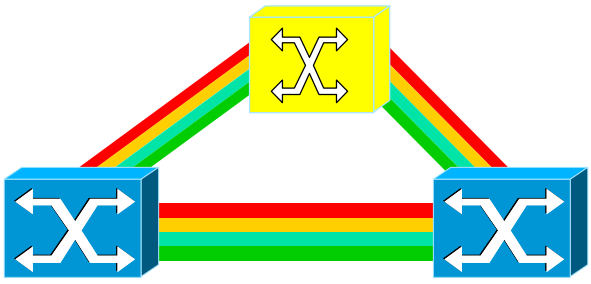

Figure 2.2: A Network Graph with Three Nodes and Four Wavelengths in Each Fiber

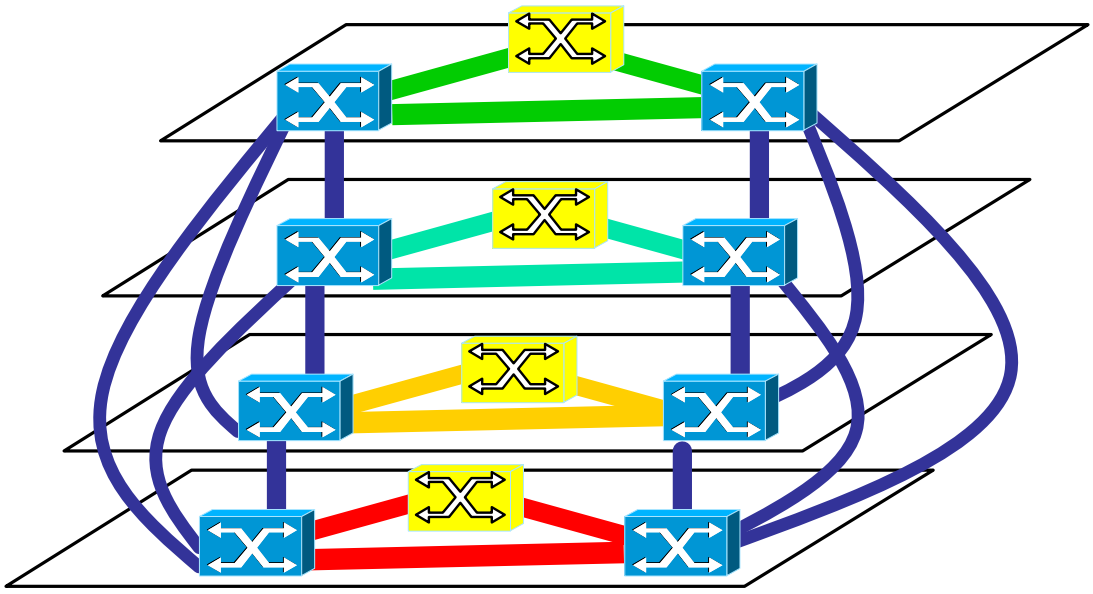

Figure 2.3: A Wavelength Graph with Three Nodes and Four Wavelength Planes

originating node and a dummy destination node. These virtual nodes are connected to the replications of the true originating node and destination node respectively and metric on each virtual link is assigned as zero.

Figure 2.2 and figure 2.3 illustrate the network graph and wavelength graph, respectively. We assume that the yellow switch in the middle does not have the wavelength translation capability.

Wavelength Graph solution could result in non-elementary path, which is a path

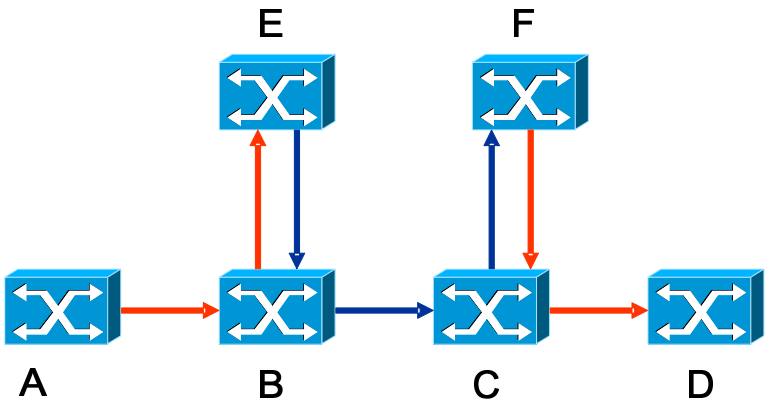

Figure 2.4: Non-elementary Path

with cycles. As shown in Figure 2.4, suppose B and C are $WIXC$ (Wavelength-Interchanging Crossconnect) which can switch a wavelength on incoming fiber to another wavelength on outgoing fiber. The lightpath we can setup from A to D is $A-B-E-B-C-F-C-D$. Therefore, running a shortest path algorithm on the network graph is not enough.

Although wavelength graph can find an optimal path for an LSP setup request, and can likewise be used to address VLAN continuity constraints, it has inherent scalability problems. The computational complexity of wavelength graph is $kn(n+k)$, where k is the number of wavelength in a fiber and n is the number of nodes in the network graph, which makes it impractical for large label space regardless of the network size.

Reference [28] provides a graph model for traffic grooming in multi-layer mesh networks, which addresses both the wavelength continuity problem and the grooming policies at the same time. Two layers are modeled and each node is replicated into $2(W+2)$ virtual nodes, where W is the number of wavelengths in each fiber. The

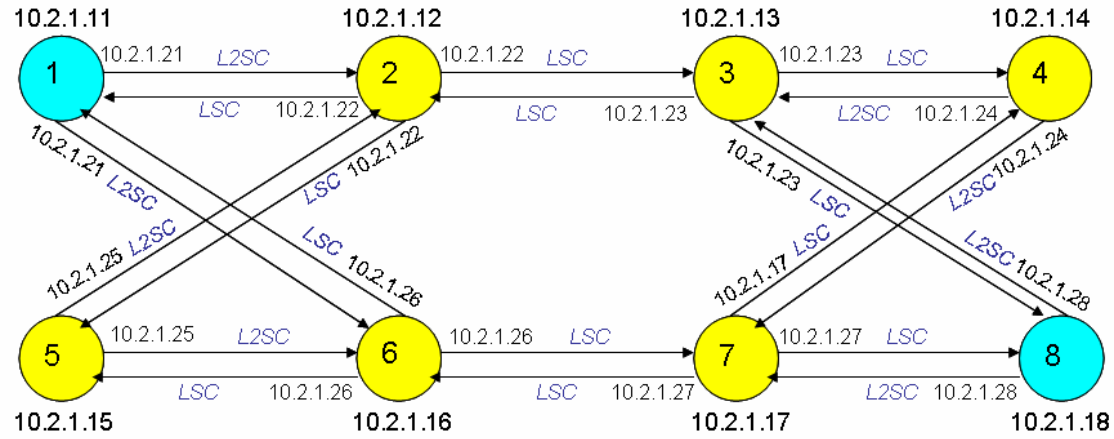

Figure 2.5: Sample Multi-Region Network Topology

lower W layers are similar to the wavelength graph defined in [27]. By adding a light-path layer and an access layer, various constraints such as transceivers and grooming capabilities can be modeled. This reference, as [27], will also have scalability problem for a large W.

Reference [29] tries to address the scalability problem by creating a link bundled auxiliary graph. This graph bundle all the wavelengths in a TE-link, which can significantly reduce the number of nodes in the auxiliary graph, but it did not provide an analytical model on the impact of wavelength continuity constraint to the blocking probability.

Switching Capability Constraint

Figure 2.5 is a sample network topology which consists of two regions, i.e., L2SC and LSC. All nodes are hybrid nodes.

An LSP request can specify 10 Gbps end-to-end connection with L2SC from node 1 to node 8. In this case, we cannot simply prune those links without L2SC because

we can adapt the switching type in the intermediate nodes to L2SC or LSC. We must also guarantee that when the flow reaches destination node, it must have been adapted back to L2SC. Because the adaptation is an interface specific functionality, the switching capability constraint is a typical example of ISAC.

An optimal path in MRN/MLN can be non-elementary, i.e., a simplex node may need to be visited more than once in a path, but each visit will take a different switching type on a different interface.

Breadth-First Search could be a potential solution to search a path, but it is non-optimal and cannot guarantee to find a path even if there is a feasible one.

Exhaustive search is not a scalable solution. It cannot handle dynamic LSP setup request even in a small network.

2.3 Summary

This chapter provides a taxonomy of constraints in multi-layer networks, which is the basis our divide-and-conquer solutions. It also reviews existing solutions to some constraints that also exist in conventional IP or optical networks, and discusses their limitation when applied to the multi-layer networks. For the GLCC and ISAC constraints, efficient and feasible solutions are to be explored in the subsequent chapters.

Chapter 3: Common Vector Solution to Generalized Label Continuity Constraint

An example of the Generalized Label Continuity Constraints is the wavelength continuity constraint in All-Optical Networks (AONs) or network based on Reconfigurable Optical Add-Drop Multiplexer (ROADM) and/or Dense Wavelength Division Multiplexing (DWDM), in which an end-to-end LSP needs to be setup. End-to-end LSP is mandatory for deterministic network services to high-end eScience applications that require high speed and high capacity lightpath, which is subject to the wavelength continuity constraint.

Another example is the VLAN tag continuity, where an end-to-end path across geographically dispersed Ethernet switches is needed. This Ethernet switching technology is deployed not only in local area networks, but also in backbones such as in the hybrid optical and packet network infrastructure, e.g., HOPI project. Ethernet switches in the network may not necessarily support VLAN tag swapping and therefore, VLAN tag continuity constraint also needs to be satisfied.

Both wavelength and VLAN tags, as mentioned above, are essentially labels. These as well as other labels can be classified under Generalized Label Continuity Constraints (GLCC)[30]. GMPLS defines a set of protocols as the standardized control plane to instantiate the LSP setup, tear-down and manipulation.

In this chapter, we discuss the applicability of a common vector solution to address the GLCC problem. We develop an analytical model to obtain the estimation on

blocking probability of LSP setup requests. We also develop a simulation model. Subsequently, numerical results are presented and compared with the analytical model. Finally, concluding remarks are made.

3.1 VLAN Tag Constraints and the Common Vector Solution

VLAN tag is a field of 12 bits, which can identify $k = 4096$ VLANs. If we create VLAN plane as we do on wavelength plane and replicate nodes and links, a network will generate hundreds of millions of nodes and links. Clearly, wavelength graph approach cannot provide a scalable solution to VLAN tag continuity constraint.

To address this problem, we utilize the Common Vector solution, where the elements of the vector represent the availability or the lack of the labels at a node across the path. The set of available labels can simply be determined by taking the logical AND across the vectors at each node on the path. A method such as Extended Indexing Reference [31] can be used to facilitate the distribution of the labels.

A vector comprised of 4096 bits, with each bit indicating the status of one of the 4096 VLAN tags used at each node is an attractive solution for large Ethernet networks in practice.

3.2 Modeling of Common Vector Solution to Wavelength Continuity Constraint

In this section, we show that the common vector approach is a reasonably good solution to wavelength continuity constraint by the following models.

We assume that the arrival process of LSP setup requests to each node is Poisson with mean of λ and the departure process is exponential with mean service time $1/\mu$.

We use the following notations in our model:

$G = \langle V, E \rangle$: Network graph G with node set V and link set E.

N: Total number of nodes in V.

M: Total number of links in E.

L: Total number of labels in each link. In this context, label is wavelength λ.

v_i: Node i in node set V, where $i = 1..N$.

e_i: Link i in link set E, where $i = 1..M$.

h_{ij}: The number of hops of the shortest path between v_i and v_j in G.

γ_{ij}: The traffic intensity between v_i and v_j in G.

f_{ij}: The traffic flow generated by γ_{ij} in the network.

f_{e_k}: The total traffic flow on link e_k.

$I(v_i, v_j, e_k)$: An indication function on whether the shortest path between v_i and v_j on the network graph traverses link e_k.

A_{e_i}: The number of available labels on link e_i.

P_{ij}: Path between v_i and v_j.

p_{ij}^m: The link that the m^{th} hop along P_{ij} traverses.

$A_{p_{ij}^{1,..,m}}$: The number of labels that are commonly available on the first m hops along P_{ij}.

We also assume that the traffic intensity in the network is not high. When an LSP request arrives at node v_i, v_i will check its Traffic Engineering Database (TEDB), prune those links without enough bandwidth and run CSPF without considering the label availability. After a path is found, v_i will perform the logical AND operation

on the vector of label availability along the path. If no commonly available label is found, the LSP request will be rejected.

To simplify the analysis, we assume that the label occupancies of the links are independent. This may not be an accurate assumption in that all the labels along an LSP are selected to be the same, which indicates the dependency of the label occupancy on different links. However, this dependency favors the common vector solution, and the actual performance of the network will be better than the analytical model we provide in the next subsections.

3.2.1 Estimation of the Number of Available Labels on a Link

Blocking probability in various network scenarios have been extensively studies, such as in [32–34]. The blocking probability in all-optical networks with and without wavelength changers has been considered in [32] and has been modeled with the assumption that the traffic load is light, an initial estimate of link blocking probability is known, and usage of a wavelength on a hop is statistically independent of other hops as well as other wavelengths. This paper concluded that the blocking probability will increase along with the number of hops with or without wavelength changers, yet did not estimate the blocking probability that caused by lack of common labels along a path which was computed without consideration of label continuity.

Reference [33] focuses on the optical network with wavelength converters. The traffic is assumed to be bounded by the number of ports in a node, which hides the dynamic nature of the traffic. The wavelength converters presented in this paper cannot model the networks with label continuity constraints.

Reference [34] discusses the blocking probability in a ring network with homogeneous traffic demand matrix. It assumes that a lightpath request will randomly pick a wavelength plane and try to search path only on that plane. Though it provides a relatively tight blocking probability bounds under this assumption, it bypasses the wavelength continuity constraints.

Computing the number of available labels on a link is a $M/M/m/m$ queueing problem. We first need to know the traffic flow on each link.

The traffic flow generated by any pair of nodes v_i and v_j is:

$$f_{ij} = \gamma_{ij} h_{ij}. \tag{3.1}$$

We first run the shortest path first algorithm between each pair of nodes on a network graph without traffic and assign:

$$I(v_i, v_j, e_k) = \begin{cases} 1 & \text{if } P_{ij} \text{ traverses } e_k \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

Due to the assumption that the traffic intensity is not high, only a small proportion of LSPs will take a path other than the shortest one. Therefore, f_{e_k} can be estimated as:

$$f_{e_k} = \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_{ij} I(v_i, v_j, e_k). \tag{3.3}$$

Since the arrival of LSP request is a Poisson process to each node, the arrival of traffic flow to each link is also nearly a Poisson process. Therefore, we can apply

results form $M/M/m/m$ queueing system to find the number of available labels.

$$P(A_{e_i} = k) = \frac{f_{e_i}^k / k!}{\sum_{j=0}^{L} (f_{e_i}^j / j!)}. \tag{3.4}$$

This estimation in equation (3.4) is very rough especially when traffic intensity is moderate. It assumes static routing and overestimate the traffic flow on each link. Reference [34] provides a more accurate estimation on the upper bounds of link blocking probability in homogeneous optical ring networks.

If we assume that the routing algorithm is dynamic, the link blocking probability is nearly the same in the whole network, and the network is complicated enough that the alternative path is of the equal length as the shortest path, we can further simplified the equation (3.3) as follows:

$$f_{e_k} = \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} f_{ij}}{M}. \tag{3.5}$$

Because we use offered load instead of carried load to calculate the traffic flow on each link, the equation (3.4) gives us the upper bound of the actual blocking ratio.

3.2.2 Estimation of Blocking Probability of LSP Setup Requests

Equation (3.4) gives the probability of $P(A_{e_i} = k)$. For an LSP with multi-hops, a successful connection needs at least one particular label that is commonly available along the path.

Given a two-hop LSP P_{ij} between v_i and v_j, we denote that there are k common

wavelength available on P_{ij}^1 and P_{ij}^2 as $P(A_{P_{ij}^{1,2}} = k)$.

Given $A_{P_{ij}^1} = k_1$ and $A_{P_{ij}^2} = k_2$, we have:

$$P(A_{P_{ij}^{1,2}} = k) = \sum_{k_1=0}^{L} \sum_{k_2=0}^{L} \Big(P(A_{P_{ij}^{1,2}} = k | A_{P_{ij}^1} = k_1, A_{P_{ij}^2} = k_2) \\ P(A_{P_{ij}^1} = k_1) P(A_{P_{ij}^2} = k_2) \Big). \quad (3.6)$$

Given L labels in a link, the number of possibilities are: $A_{P_{ij}^1} = k_1, A_{P_{ij}^2} = k_2$ is

$$\Big| \{ A_{P_{ij}^1} = k_1, A_{P_{ij}^2} = k_2 \} \Big| = \binom{L}{k_1} \binom{L}{k_2} \quad (3.7)$$

Given L labels in a link, $A_{P_{ij}^1} = k_1$ and $A_{P_{ij}^2} = k_2$, $A_{P_{ij}^{1,2}} = k$ means that that we choose k_1 out of L on P_{ij}^1 first. Because there are k common labels on P_{ij}^1 and P_{ij}^2, it means we choose k_1 labels from the first fiber randomly, and choose k labels out of k_1 labels and $(k_2 - k)$ labels out of $(L - k_1)$ labels on P_{ij}^2.

The number of possibilities are:

$$\Big| \{ A_{P_{ij}^1} = k_1, A_{P_{ij}^2} = k_2, and A_{P_{ij}^{1,2}} = k \} \Big| = \binom{L}{k_1} \binom{k_1}{k} \binom{L - k_1}{k_2 - k} \quad (3.8)$$

Dividing equation (3.8) by equation (3.7), we have:

$$P(A_{P_{ij}^{1,2}} = k | A_{P_{ij}^1} = k_1, A_{P_{ij}^2} = k_2) = \frac{\binom{k_1}{k} \binom{L - k_1}{k_2 - k}}{\binom{L}{k_2}}, \quad (3.9)$$

where $max(0, k_1 + k_2 - L) \leq k \leq min(k_1, k_2)$.

Therefore, we have:

$$P(A_{P_{ij}^{1,2}}=k)=\sum_{k_1=1}^{L}\sum_{k_2=1}^{L}\frac{\binom{k_1}{k}\binom{L-k_1}{k_2-k}}{\binom{L}{k_2}}P(A_{p_{ij}^1}=k_1,A_{p_{ij}^2}=k_2), \tag{3.10}$$

where $max(0,k_1+k_2-L)\leq k\leq min(k_1,k_2)$.

The estimation of the case that there are k common labels along a $n-$hop LSP is an iterative process. We have:

$$P(A_{P_{ij}^{1,2,\ldots,n}}=k)=\sum_{k_1=0}^{L}\sum_{k_2=0}^{L}\Big(P(A_{P_{ij}^{1,2,\ldots,n}}=k|A_{P_{ij}^n}=k_2,A_{P_{ij}^{1,\ldots,n-1}}=k_1)$$
$$P(A_{P_{ij}^{1\ldots n-1}}=k_1)P(A_{P_{ij}^n}=k_2)\Big) \tag{3.11}$$

Therefore, the probability of blocking is $P(A_{P_{ij}^{1,2,\ldots,n}}=0)$.

We assume that the alternative path between any pair of nodes v_i and v_j is of the same length as the shortest path. The expected blocking probability is as follows:

$$P_b=\frac{\sum_{i=1}^{N}\sum_{j=1}^{N}\gamma_{ij}P(A_{P_{ij}^{1,2,\ldots,h_{ij}}}=0)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\gamma_{ij}} \tag{3.12}$$

If we use (3.3) to estimate the traffic flow on a link, we called it an estimation based on static routing. If we use (3.5) to estimate the traffic flow on a link, we called it an estimation based on homogeneous link load assumption.

Given arrival and departure ratio $\rho=\gamma/\mu$, we denote the probability of blocking in $M/M/m/m$ as $B(\rho,m)$.

Theorem 1. *In Erlang B model, given the number of servers is m, and the expected number of idle servers is $E[a_m]$, when the number of servers is increased to $m+1$ and the ρ is small enough such that $B(\rho, m) \leq 1/(m+1)$, we have:*

$$\frac{E[a_{m+1}]}{E[a_m]} > \frac{m+1}{m}. \tag{3.13}$$

Proof. Given the $M/M/m/m$ formula of n servers busy as:

$$P[n] = \frac{\rho^n/n!}{\sum_{i=0}^{m}(\frac{\rho^i}{i!})}, \tag{3.14}$$

We have the expected number of busy servers $E[b_m]$ as:

$$\begin{aligned} E[b_m] = \sum_{n=0}^{m}(n \cdot P[n]) &= \frac{\sum_{n=1}^{m}\rho^n/(n-1)!}{\sum_{i=0}^{m}(\rho^i/i!)} \\ &= \rho\frac{\sum_{n=0}^{m-1}(\rho^n/n!)}{\sum_{i=0}^{m}(\rho^i/i!)} \\ &= \rho\frac{\sum_{n=0}^{m}(\rho^n/n!) - \rho^m/m!}{\sum_{i=0}^{m}(\rho^i/i!)} \\ &= \rho(1 - P[n=m]). \end{aligned} \tag{3.15}$$

Therefore, we have:

$$E[a_m] = m - E[b_m]. \tag{3.16}$$

Given that:

$$\frac{E[a_{m+1}]}{E[a_m]} = \frac{(m+1) - \rho(1 - B(\rho, m+1))}{(m - \rho(1 - B(\rho, m))}, \tag{3.17}$$

it can be easily shown that (3.13) is equivalent with

$$1 - (m+1)B(\rho, m) + mB(\rho, m+1) > 0. \tag{3.18}$$

Because we assume that $B(\rho, m) \leq 1/(m+1)$ and $mB(\rho, m+1) > 0$, the inequality (3.18) holds. □

Theorem 1 shows that the proportion of available labels increases faster than that of the label increase. From this conclusion, we can prove the following theorem.

Theorem 2. *For any integer $m > 1$, we have:*

$$\frac{\binom{mL - mk_1}{mk_2}}{\binom{mL}{mk_2}} < \frac{\binom{L - k_1}{k_2}}{\binom{L}{k_2}} \tag{3.19}$$

Proof. The Left Hand Side (LHS) can be expanded as follows:

$$\begin{aligned} \frac{\binom{mL - mk_1}{mk_2}}{\binom{mL}{mk_2}} &= \frac{\frac{(mL-mk_1)!}{(mk_2)!(mL-mk_1-mk_2)!}}{\frac{(mL)!}{(mk_2)!(mL-mk_2)!}} \\ &= \frac{(mL - mk_1)(mL - mk_1 - 1)\cdots(mL - mk_1 - mk_2 + 1)}{(mL)(mL-1)\cdots(mL - mk_2 + 1)}. \end{aligned} \tag{3.20}$$

The Right Hand Side (RHS) can be expanded as follows:

$$\frac{\binom{L-k_1}{k_2}}{\binom{L}{k_2}} = \frac{\frac{(L-k_1)!}{k_2!(L-k_1-k_2)!}}{\frac{L!}{k_2!(L-k_2)!}} = \frac{(L-k_1)(L-k_1-1)\cdots(L-k_1-k_2+1)}{L(L-1)\cdots(L-k_2+1)}. \tag{3.21}$$

Because $\frac{mL}{mL-mk_1} = \frac{L}{L-k_1}$, $\frac{mL-mk_1-m}{mL-m} = \frac{L-k_1-1}{L-1}$, $\cdots$, and $\frac{mL-mk_1-mk_2+m}{mL-mk_2+m} = \frac{L-k_1-k_2+1}{L-k_2+1}$, we divide (3.21) by (3.20) and obtain:

$$\frac{LHS}{RHS} = \prod_{i=1}^{m-1}(\frac{mL-mk_1-i}{mL-i})\bullet\prod_{i=m+1}^{2m-1}(\frac{mL-mk_1-i}{mL-i})\bullet\cdots\bullet\prod_{i=(k_2-1)m+1}^{k_2m-1}(\frac{mL-mk_1-i}{mL-i})$$

$$= \prod_{j=0}^{k_2-1}\prod_{i=jm+1}^{(j+1)m-1}(\frac{mL-mk_1-i}{mL-i}) \tag{3.22}$$

Because each element in equation 3.22 is smaller than 1, therefore, we have:

$$\frac{\binom{mL-mk_1}{mk_2}}{\binom{mL}{mk_2}} < \frac{\binom{L-k_1}{k_2}}{\binom{L}{k_2}} \tag{3.23}$$

□

Theorem 1 shows that the expected number of idle labels increase faster than the increase of the total number of labels. Theorem 2 shows that when the number available labels increase, the probability of blocking caused by the lack of common

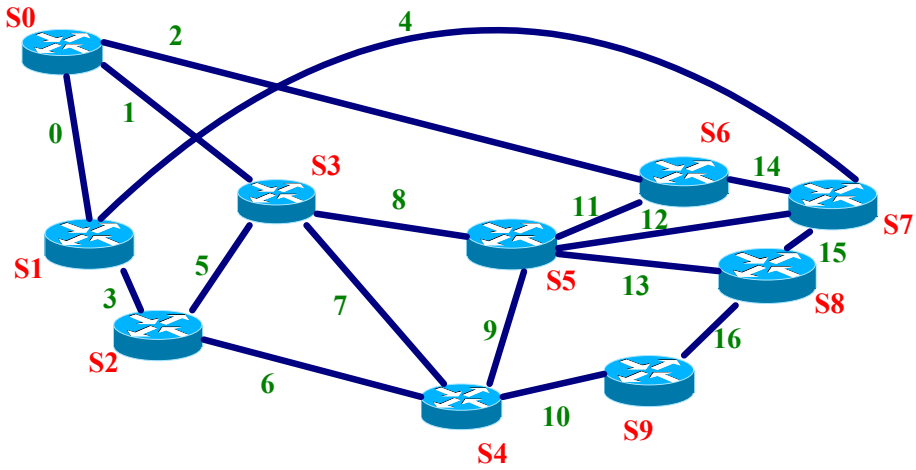

Figure 3.1: Network Topology

labels on a two-hop path will be reduced.

This conclusion implies that the common vector solution for the label continuity constraints will have less negative impact on performance from blocking probability perspective as the label space grows larger.

3.2.3 Numerical Results

Numerical Results on Homogeneous Traffic Matrix

We consider a network topology as depicted in figure 3.1, and assume that the traffic matrix is homogeneous. We consider that each link has 8 wavelengths.

Figure 3.2 shows the estimation of blocking probability based on both static route and the homogeneous link load assumption respectively. It also showed the actual blocking probability by simulating 10 million LSP setup requests. We can clearly see that the estimation based on homogeneous link load assumption is closer to the actual blocking rate. Both estimations are upper bounds of the actual blocking rate.

Figure 3.3 compares the blocking ratio of the common vector solution with the

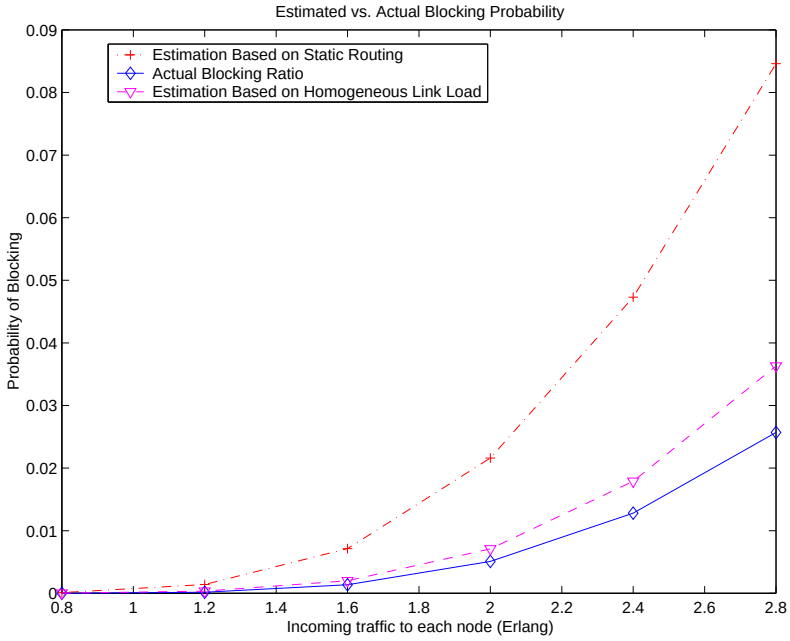

Figure 3.2: Comparison of Estimated and Actual Blocking Probability

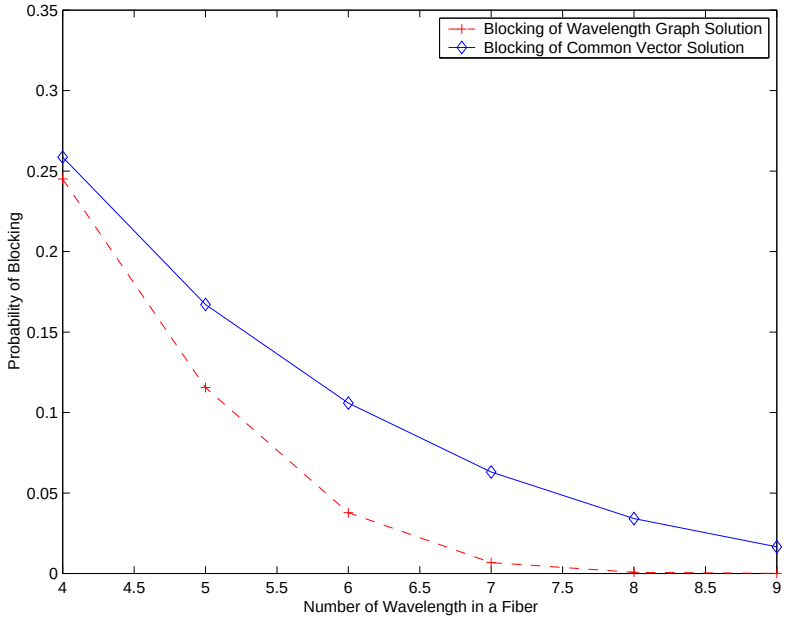

Figure 3.3: Comparison of Actual Blocking in Wavelength Graph Solution and Common Vector Solution

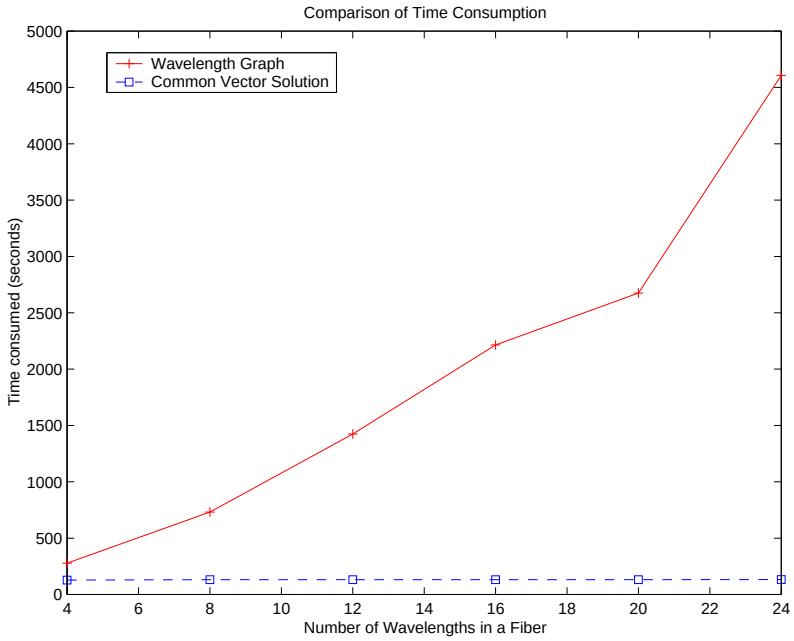

Figure 3.4: Comparison of Time Consumption

wavelength graph solution. Given incoming traffic of 3 Erlangs at each node, we simulated the cases that there are 4 to 8 wavelengths in a fiber. None of the node has wavelength translation capability. The wavelength graph solution can accept more calls than the common vector solution. However, when the network load is very high or very low, the improvement on the call acceptance ratio is not significant. When the network load is moderate, wavelength graph solution demonstrates a much better performance with regard to call acceptance.

The computational complexity of the common vector solution is equivalent to Dijkstra algorithm and then find the availability of labels on each link traversed by the LSP. Therefore, it is $O(M+Nlog(N))+O(N+L)$. The computational complexity of the wavelength graph solution is given in [27], which is $O(LN(N+L))$.

Figure 3.4 shows that the computational complexity of common vector solution is almost irrelevant to the number of wavelengths on a link, while the wavelength graph

Table 3.1: Matrix of Mean Holding Time between Nodes

node	0	1	2	3	4	5	6	7	8	9
0	0.0	0.6	0.7	0.8	1.2	1.3	1.4	1.5	0.5	1.0
1	0.6	0.0	0.7	0.9	0.8	1.0	1.1	1.2	1.4	1.3
2	0.8	0.7	0.0	1.6	0.4	1.2	1.3	0.5	1.0	1.5
3	0.8	0.7	1.6	0.0	1.2	0.4	1.2	0.8	1.3	1.0
4	0.5	1.6	1.7	0.3	0.0	0.4	1.5	1.8	0.2	1.0
5	1.8	0.8	0.2	1.2	0.9	0.0	1.1	1.8	0.2	1.0
6	1.8	1.8	1.7	1.7	0.2	0.2	0.0	0.3	1.0	0.3
7	1.8	1.5	1.1	0.5	0.9	0.5	0.2	0.0	1.5	1.0
8	1.4	1.2	0.8	0.6	1.2	0.8	0.7	1.3	0.0	1.0
9	1.8	1.8	1.7	0.3	1.0	0.2	0.8	0.2	1.2	0.0

solution shows that even if the wavelength number is small, it still takes much more time than the common vector solution does.

Numerical Results on Nonhomogeneous Traffic Matrix

Homogeneous and non-homogeneous traffic matrix may result in different network blocking scenarios. To make the comparison, we consider the following three designed cases:

1. Traffic generated at each node is the same and the traffic matrix is homogenous;

2. The inter-arrival of call requests at each node is the same and the mean holding time between node v_i and v_j is in proportion to the value in i^{th} row and j^{th} column given in table 3.1; and

3. The inter-arrival of call requests at each node is in proportion to the values given in table 3.2 and the mean holding time between node v_i and v_j is in proportion to the value in i^{th} row and j^{th} column given in table 3.1.

Table 3.2: The Inter-arrival of Call Requests at Each Node

node	0	1	2	3	4	5	6	7	8	9
Incoming Traffic Intensity	1.0	1.4	0.6	0.8	1.2	1.3	0.7	1.5	0.5	1.0

Table 3.3: Comparison of Blocking Probability of Common Vector Solution between Case 1 and 2

Wavelength	4	5	6	7	8	9
Common Vector Solution for Case 1	0.259	0.167	0.106	0.063	0.034	0.0166
Common Vector Solution for Case 2	0.262	0.173	0.110	0.068	0.039	0.0216

Table 3.4: Comparison of Blocking Probability of Wavelength Graph Solution between Case 1 and 2

Wavelength	4	5	6	7	8	9
Wavelength Graph Solution for Case 1	0.245	0.116	0.0378	0.0068	0.000747	0.000118
Wavelength Graph Solution for Case 2	0.251	0.127	0.0436	0.0091	0.001083	0.000085

Table 3.5: Estimated vs. Actual Blocking Probability of Common Vector Solution for Case 1 and 2

Wavelength	4	5	6	7	8	9
Common Vector Solution for Homogeneous Traffic Matrix	0.262	0.173	0.110	0.0683	0.0394	0.0216
Estimation of Blocking Probability Based on equation 3.3	0.507	0.392	0.286	0.196	0.125	0.0735
Estimation of Blocking Probability Based on equation 3.5	0.472	0.332	0.211	0.119	0.0595	0.0265

Table 3.3 and 3.4 show the blocking probability of common vector solution and wavelength graph solution with homogeneous and non-homogeneous traffic matrix, respectively.

When the incoming traffic intensity from each node is the same, table 3.5 shows the estimated and actual blocking probability of common vector solution for non-homogeneous traffic matrix. We can see that as the blocking probability decreases, the estimation based on both equation (3.3) and (3.5) is improving. The estimation based on equation (3.5) is much better than equation (3.3) because dynamic path computation will make the traffic flow in the network nearly homogenous.

However, when the traffic intensity generated at each node is different, the network will demonstrate a different blocking scenario.

Figure 3.5 and 3.6 show the blocking probability of all the three cases of the common vector and wavelength graph solutions, respectively. Figure 3.7 shows the comparison between estimated and actual blocking probability of the common vector solution. The estimation based on both equation (3.3) and (3.5) are showed. Figure 3.5 and 3.6 clearly show the performance degradation, which means that unbalanced traffic from each node can increase the network blocking probability. Figure 3.7 shows

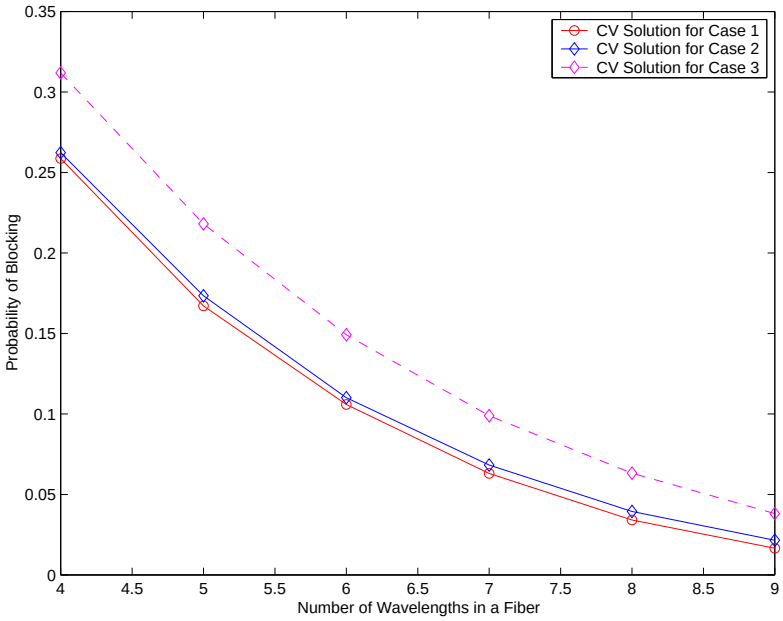

Figure 3.5: Comparison of Blocking Probability of Common Vector Solution between Case 1, 2 and 3

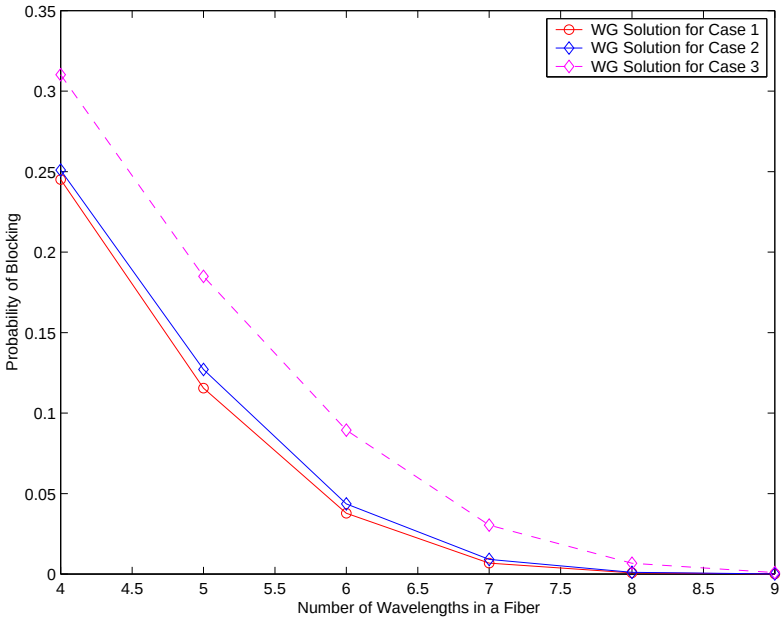

Figure 3.6: Comparison of Blocking Probability of Wavelength Graph Solution between Case 1, 2 and 3

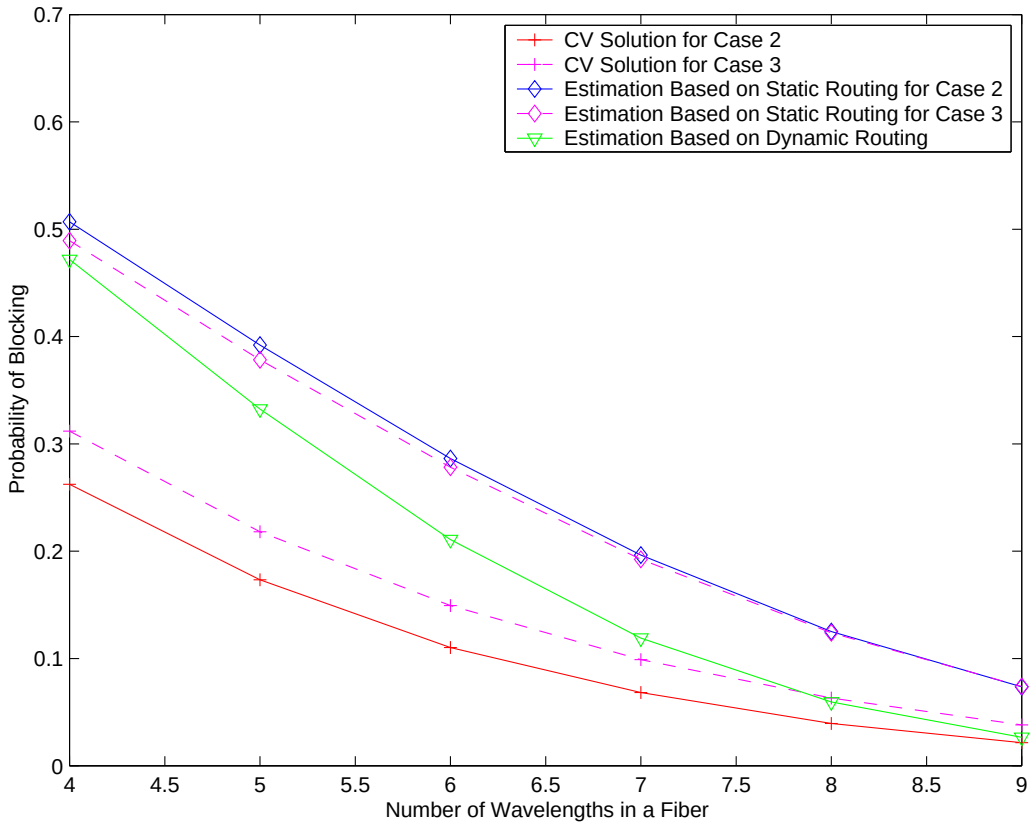

Figure 3.7: Estimated vs. Actual Blocking Probability of Common Vector Solution for Case 1, 2 and 3

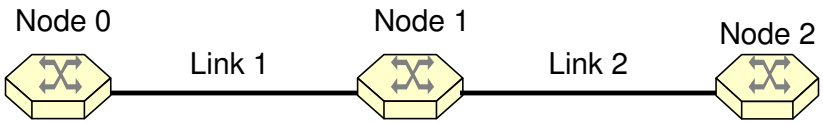

Figure 3.8: Linear Topology with Two Nodes

that when the traffic load is light, the estimation of blocking ratio based on dynamic routing is even less than the actual blocking ratio. This is because the dynamic routing assumption hides the non-homogeneous traffic demand from each node.

This implies that given the same traffic intensity, a network with nearly homogeneous traffic demand has better performance from blocking probability perspective.

Numerical Results on Label Space Increase

Given the same proportion of expected idle wavelengths as the number of wavelengths increases, we want to verify whether the common vector solution will demonstrate a better performance from blocking perspective in a network with homogeneous traffic intensity on all the links. Figure 3.8 is a network with only two nodes. Table 3.6 lists the incoming traffic per node and blocking probability on each link obtained by (3.4). According to (3.15), the incoming traffic to each node is designed in such a way that the number of expected idle wavelengths on each link is increased proportionally to the number of wavelengths.

Figure 3.9 clearly shows that the expected and actual blocking probabilities decrease fast when the label space grows larger. The actual blocking ratio for the two hop traffic also decreases fast. This result corroborates the conclusion of section 3.2.2.

Figure 3.10 illustrates the expected and actual blocking probabilities when we keep the same link blocking probability as the number of wavelengths in a link increases.

Table 3.6: Traffic Intensity and Link Blocking Probability Given the Same Proportion of Idle Wavelengths as the Label Space Increases

Wavelength	4	8	12	16	20
Incoming Traffic per Node	1	1.8499	2.7314	3.6262	4.52625
Traffic on Each Link	2	1.8499	2.7314	3.6262	4.52625
P_b obtained by (3.4)	0.095238	0.02183	0.006281	0.001986	0.000658
Expected Number of Idle Wavelengths	2.1905	4.381	6.5715	8.762	10.9525

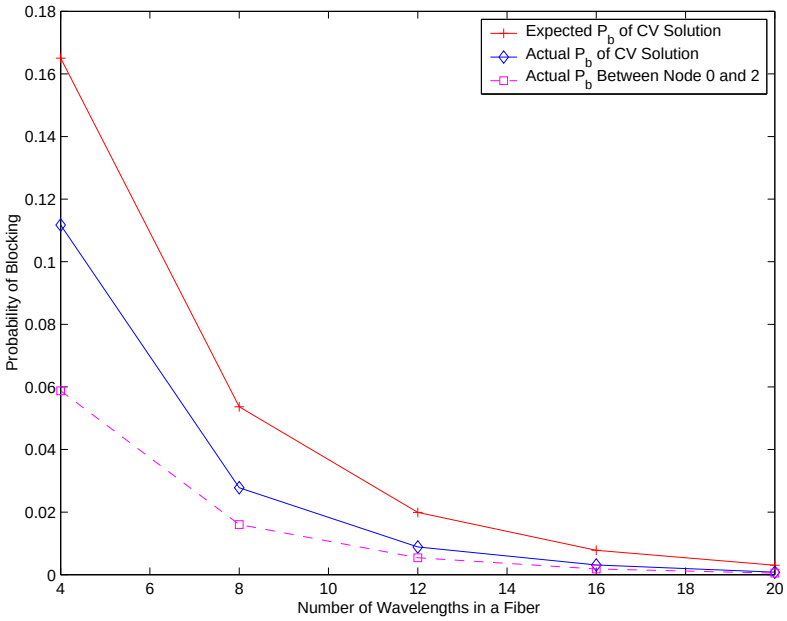

Figure 3.9: Decrease of Blocking Probabilities as the Label Space Increases

Table 3.7: Expected Number of Idle Wavelengths Given the Same Link Blocking Probability as the Label Space Increases

Wavelength	4	8	12	16	20
Incoming Traffic per Node	1	2.7533	4.6716	6.6660	8.7048
Traffic on Each Link	2	5.5065	9.3433	13.332	17.4096
P_b obtained by (3.4)	0.09524	0.09524	0.09524	0.09524	0.09524
Expected Number of Idle Wavelengths	2.1905	3.0179	3.5466	3.9377	4.2484

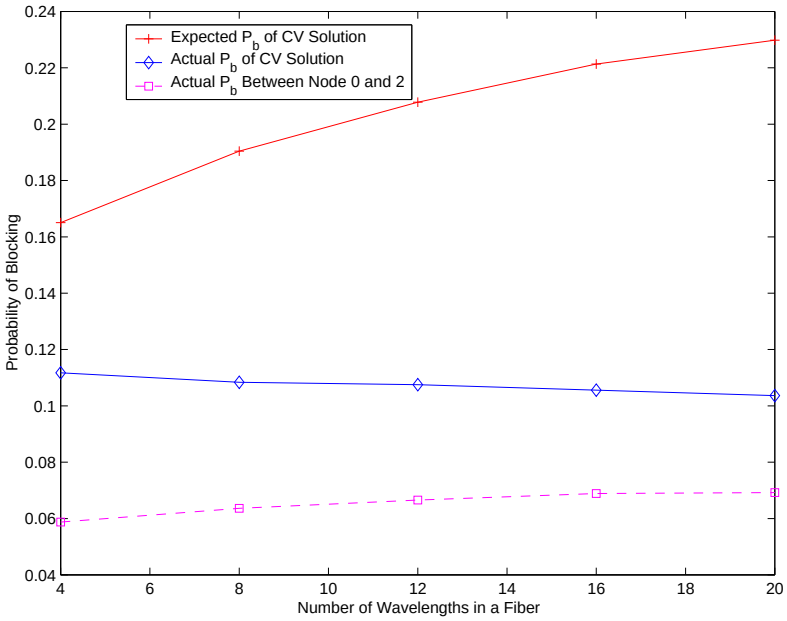

Figure 3.10: Given the Same Link Blocking Probability, the Blocking Probabilities of the Network along the Increase of Label Space

As shown in table 3.7, the expected number of idle wavelengths in a link increases slower than the total number of wavelength does. Therefore, the expected blocking probability increases fast. Because the label selection on different links is correlated, the actual blocking ratio between node 0 and node 2 only increases slightly. We also note the overall blocking ratio reduces slightly because when we reject more call setup requests between node 0 and node 2, the number of accepted one-hop calls is increased.

Table 3.8 gives the incoming traffic per node and blocking probability on each link without wavelength obtained by (3.4). According to (3.15), the incoming traffic to each node is designed in such a way that the number of expected idle wavelengths on each link is increased proportionally to the number of wavelengths.

Table 3.8: Traffic Intensity and Link Blocking Probability Given the Same Proportion of Idle Wavelengths as the Label Space Increases in a Network Depicted in Fig. 3.1

Wavelength	4	8	12	16	20
Incoming Traffic per Node	1	1.9608	3.2635	4.3508	5.4385
Average Traffic on Each Link	1.1111	2.1786	3.6261	4.8324	6.0427
P_b obtained by (3.4)	0.0210	$3.45 * 10^{-6}$	$4.24 * 10^{-8}$	$2.64 * 10^{-10}$	$1.70 * 10^{-12}$
Expected Number of Idle Wavelengths	2.9123	5.8245	8.7369	11.6492	14.5615

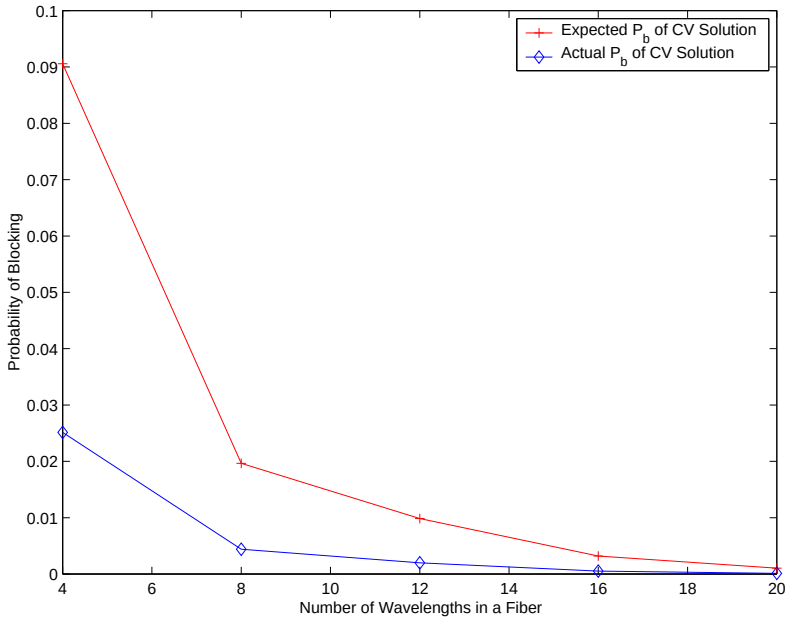

Figure 3.11: Decrease of Blocking Probabilities as the Label Space Increases in the Network Depicted in Figure 3.1

Figure 3.11 clearly shows that the expected and actual blocking probabilities decrease fast when the label space grows larger. This figure corroborates the intuition that path with any number of hops will be benefited from the increase of label space.

3.3 Summary

Generalized label continuity constraints becomes important constraints in label switched networks. This chapter provides a simple and efficient solution to this type of constraints. Here both analytical model and numeric results show that common vector solutions can address the scalability problem when the label space is large with reasonably good performance from blocking probability point of view.

Common vector solution has two steps. The first step it to compute the path without consideration on label continuity and the second step is to find a common label along the path. Since wavelength graph solution combine these two steps together, it has a better view on the network resources and can find an optimal solution. Hence, wavelength graph solution is recommended for small label space and common vector solution is good for large label space.

Chapter 4: Solutions to Switching Type Adaptation Constraints in Multi-Region and Multi-layer Networks

From the perspective of the control plane, a set of network elements that share the same switching technology is defined as a region. Currently, five regions are defined and they are PSC, L2SC, TDM, LSC and FSC region. A network of elements with multiple switching technology is called a multi-region network. A layer describes a data plane switching granularity level, such as VC4 and VC-12 in TDM region.

OSPF extension defined in [5] provides the control plane capability of obtaining all the traffic engineering data from different layers to build the TEDB. Interface Switching Capability Descriptor (ISCD) [35] describe the attributes of a TE link in its sub-TLV and a hybrid node can advertise multiple ISCDs for the same TE-link.

Figure 4.1 is an example of the architecture of a hybrid node with two switching elements. One is PSC switching element and the other is photonic switching element. At least one interface must advertise multiple switching capabilities for a hybrid node. Besides the external links, the two switching elements are also connected by internal links. Because each internal link has finite capacity, the TE-attributes associated with these internal links must also be advertised for the path computation purpose.

The nature of multi-layer networks implies that the collaborative mechanism across layers needs to be defined. Reference [36] defines the Interface adaptation

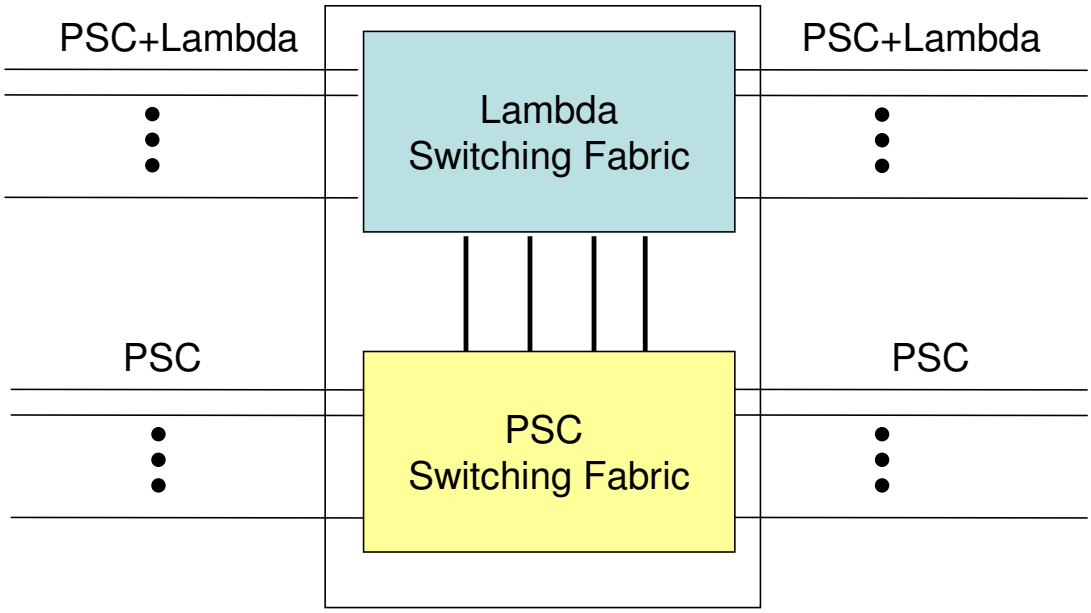

Figure 4.1: Switching Elements in a Node

capability descriptor (IACD) that contains necessary information for the path computation algorithm to determine the forwarding and switching capability of the internal links. Reference [37] proposed a more efficient approach called Node Adaptation Capability Descriptor (NACD) that can also reduce the LSP blocking probability. In [37], OSPF is extended with metric of dynamic values based on the availability of both external and internal available resources.

Once an LSP is found, it is set up through GMPLS RSVP-TE. LSRs on both ends of the LSP can form Forwarding Adjacency (FA) by advertising this LSP into the instance of OSPF/ISIS [38] and this LSP can be used for further path computation. The signaling capability of GMPLS makes it possible to optimize the network resource utilization and provision the service rapidly.

Because a path can be setup across layers, there may be a large gap between the required bandwidth on an upper layer and the large available bandwidth on a lower

layer. For example, a request of a 10 Gigabit Ethernet with switching type L2SC may instantiate through an OC-768 WDM channel. A large portion of the OC-768 channel will be wasted if other traffic flows are not multiplexed onto this lightpath. Therefore, traffic-grooming in multi-layer networks become an important research problem because it has significant impact on the loss performance [39].

In a network without a unified control plane, path computation may be divided into sub-problems in different layers. Connections in a lower layer are setup to create a virtual topology for the neighboring upper layer. Path computation on each layer is based on the virtual topology created by the neighboring lower layer with certain objective functions for optimality. This divide-and-conquer approach will definitely result in sub-optimal solutions because optimality of each layer is correlated and optimality in optical layer has been proved to be NP-hard [16].

Optimization of static and dynamic traffic requests are very different in terms of objective function and methodology. Given static traffic request, we can assume that the network has enough resource to accommodate all the requests. The optimization is from resource perspective, i.e. the paths are setup in such a way that minimal resources are used. For dynamic traffic requests, minimization of blocking probability or maximization network throughput is the objective function. For static traffic requests, integer linear programming could search an optimal path in a small network, but it has scalability problem. This chapter is to provide solutions for dynamic traffic requests in multi-layer networks.

4.1 Challenges to Path Computation Elements

With all the possibilities of network resource optimization and rapid service provisioning, MRN/MLN also introduces new challenges to path computation if we consider constraints such as the switching type adaptation, encoding type, bandwidth granularity, and etc. Cross-layered connection need both horizontal interaction and integration, which means the collaboration between network elements on the same layer, and vertical interaction and integration, which means the collaboration between different layers or regions.

LSP setup in multi-region and multi-Layer networks (MRN/MLN) is more complicated in that different layers have different switching technologies. Some network elements are multi-switching capable and they can adapt from one switching type to another. Unlike the wavelength conversion that can generally translate any incoming wavelength on any incoming fiber to any outgoing wavelength on any other fiber, adaptation function of hybrid LSRs is interface specific and generally, the adaptation cannot be done from any particular switching type to any other particular type on an interface. This type of constraints is named as Interface Specific Adaptation Constraints (ISAC) in this dissertation.

Figure 4.2 illustrate an example of the interface specific adaptation constraint. Suppose that all the links with switching type L2SC have 200Mb/s unreserved bandwidth and all the links with switching type TDM have two Virtual Container level 4 (VC-4) available. Node A is a hybrid node that can adapt between switching type L2SC and TDM and all the other nodes are single-switching type capable nodes. Since the payload rate in each VC-4 is smaller than 155.52Mb/s, we need at least two

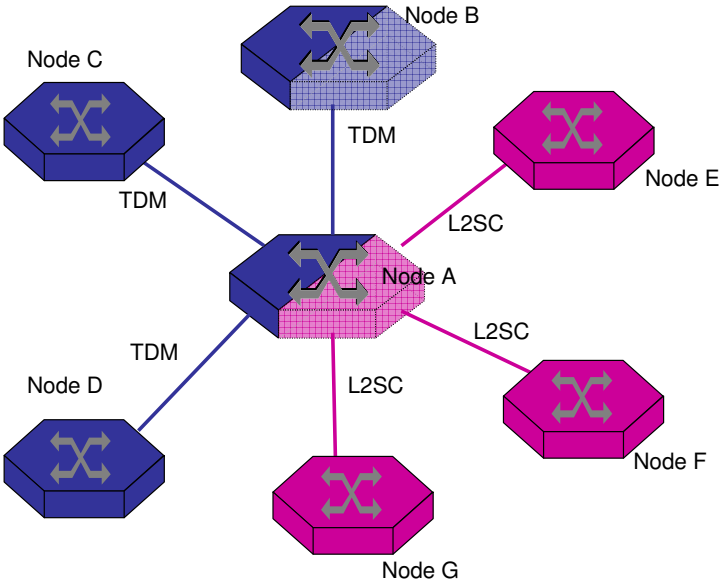

Figure 4.2: An Example of Interface Specific Constraint

VC-4 to transport 200Mb/s data.

Now we want to setup an Enterprise System Connection (ESCON) of 200Mb/s through this network. We also assume that only the interface between node A and B support Virtual concatenation (VCAT) as defined in [40]. Interface card between node A and C only support standard contiguous concatenation (CCAT). Reference [41] shows that VCAT has significantly better resource efficiency than CCAT, but we cannot preclude equipment with CCAT capability only. As we know, CCAT can only adapt the 200Mb/s ESCON service to VC-4-4c, which interface between node A and C cannot provide due to bandwidth shortage. However, if this 200Mb/s ESCON service has already been encapsulated into TDM protocol and come in through a TDM interface of node A, node A can switch it to the TDM interface between node A and C. Therefore, we cannot simply prune the interface between node A and node C because it may still be a feasible segment along the path.

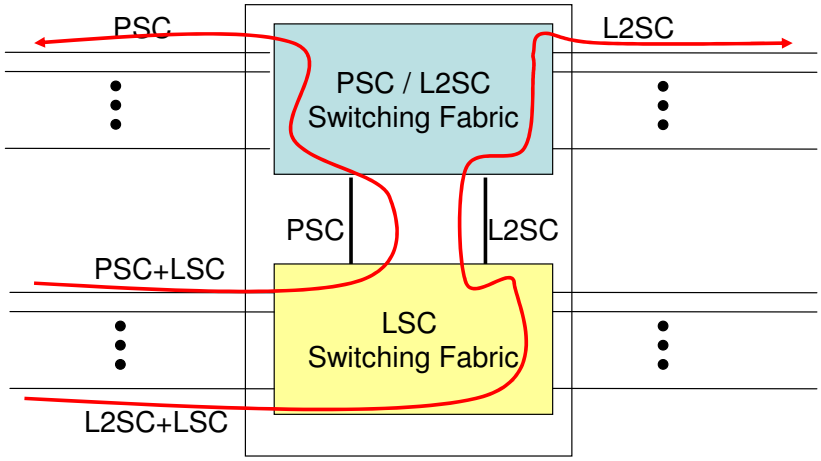

Figure 4.3: Another Example of Interface Specific Constraint

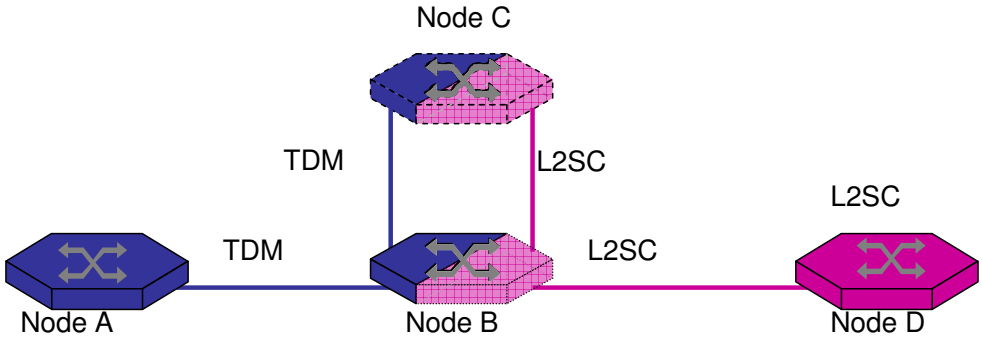

Figure 4.4: An Example of Non-Elementary Path in Multi-Region Networks

Figure 4.3 illustrate an example given in section 3.1 of [36]. A switching fabric that supports both PSC and L2SC functionalities is assembled with LSC interfaces enabling "lambda" encoding. In the switching fabric, some interfaces can terminate Lambda LSPs and perform frame (or cell) switching whilst other interfaces can terminate Lambda LSPs and perform packet switching. In this case, traffic can only flow along the red arrows shown in figure 4.3.

Similar to the wavelength continuity constraint, switching type adaptation may also generate non-elementary path as illustrated in Figure 4.4. Node A and D are

single-switching type capable nodes, node B is a simplex node with L2SC-only interface and TDM-only interface, and node C is a hybrid node which can adapt between L2SC and TDM. Because node B is visited twice, such path cannot be found through Dijkstra algorithm on a network graph.

Optimality of the path cost is another challenging problem. The cost of setting up an end-to-end LSP in MRN/MLN is a combination of three components as follow:

1. the cost for traversing a link on certain wavelength and switching type;
2. the cost for switching type adaptation when the path has to take a different switching technology at some intermediate nodes; and
3. the cost for wavelength translation when the path has to switch to a different wavelength at some intermediate nodes.

In this chapter, we assume that we can use the common vector solution provided in chapter 3 to address the wavelength continuity constraint.

Note that if only the first cost factor is present, then the problem is simplified to looking for a shortest path. If only the first and third cost components are present, then the problem is simplified to looking for a semi-lightpath in a single region network.

References [27–29] solve the wavelength continuity constraints by auxiliary graph. In these graph models, nodes are replicated and edges are created based on the connectivity information. However, these models cannot capture the connectivity information of the interface switching and adaptation capability constraints. Therefore, a new graph modeled needs to be proposed.

This chapter will provide a solution, called Channel Graph Solution for Interface Switching Adaptation Constraints (CSISAC) to address the adaptation and cost-optimization challenge.

4.2 Transformation between Network Graph and Channel Graph

In this chapter, we introduce the concept of the channel graph. It can be viewed as the dual of the network graph in terms of node and link definition[20]. The Path Computation Element (PCE) will run CSPF on the channel graph to compute the path.

4.2.1 Notations

We use the following notations in our model:

$G = \langle V, E \rangle$: Network graph G with node set V and link set E.

$H = \langle N, A \rangle$: Channel graph H with node set N and link set A.

$|V|$: Total number of nodes in V.

$|E|$: Total number of links in E.

$|N|$: Total number of nodes in N.

$|A|$: Total number of links in A.

v_i: Node i in node set V, where $i = 1..|V|$.

e_i: Link i in link set E, where $i = 1..|E|$.

L: Total number of switching types on links.

$I_{e_i}^{s_j}$: An indication function whether switching type s_j is available on link e_i, where $s_j = 1, 2, ..., L$.

$d_G^-(v_i)$: The in degree of node v_i in graph G.

$d_G^+(v_i)$: The out degree of node v_i in graph G.

$\varepsilon_G^-(v_i)$: The incoming link set of node v_i. Clearly, $|\varepsilon_G^-(v_i)| = d_G^-(v_i)$.

$\varepsilon_G^+(v_i)$: The outgoing link set of node v_i. Clearly, $|\varepsilon_G^+(v_i)| = d_G^+(v_i)$.

$\varepsilon_G^-(v_i, s_k)$: The set of incoming links with switching type s_k to node v_i.

$\varepsilon_G^+(v_i, s_k)$: The set of outgoing links with switching type s_k from node v_i.

$\langle e_i, s_j \rangle$: Switching type s_j on link e_i in G.

$H(e_i)$ or $H(\langle e_i, s_j \rangle)$: The head node of directed link e_i.

$T(e_i)$ or $T(\langle e_i, s_j \rangle)$: The tail node of directed link e_i.

$n_{i,j}^k$: Node $n_{i,j}^k$ in node set N, where $k = 1..|N|$ and can be mapped to $\langle e_i, s_j \rangle$ in G.

a_i^k: Link a_i^k in link set A, where $i = 1..|A|$ and can be mapped to node v_k in G.

$\Gamma_G^-(v_k)$: The set that enumerates $\langle e_i, s_j \rangle$ such that $e_i \in \varepsilon_G^-(v_k)$ and $I_{e_i}^{s_j} = 1$.

$\Gamma_G^+(v_k)$: The set that enumerates $\langle e_i, s_j \rangle$ such that $e_i \in \varepsilon_G^+(v_k)$ and $I_{e_i}^{s_j} = 1$.

$\Theta_G(v_k)$: Indication matrix of adaptation capability of node v_k. $\Theta_G(v_k)$ is a $|\Gamma_G^-(v_k)|$ by $|\Gamma_G^+(v_k)|$ matrix, where an element in i^{th} row and j^{th} column is denoted as $\theta_{i,j}(v_k)$.

h_{ij}: The number of hops of the cost optimal path between v_i and v_j in G.

Π_{ij}: Path between v_i and v_j.

π_{ij}^k: π_{ij}^k is a tuple$\langle e_x, s_y \rangle$, which indicates that the k^{th} hop along Π_{ij} traverses link e_x and switching type s_y.

$\varphi(\pi_{ij}^k)$: A function to extract link index from π_{ij}^k. If $\pi_{ij}^k = \langle e_x, s_y \rangle$, then $\varphi(\pi_{ij}^k) = e_x$;

$\psi(\pi_{ij}^k)$: A function to extract switching type from π_{ij}^k. If $\pi_{ij}^k = \langle e_x, s_y \rangle$, then $\psi(\pi_{ij}^k) = s_y$;

From the above definition, we have,

$$I_{e_i}^{s_j} = \begin{cases} 1 & \text{if } e_i \text{ has switching type } s_j; \\ 0 & \text{otherwise;} \end{cases} \tag{4.1}$$

and,

$$\theta_{i,j}(v_k) = \begin{cases} 1 & \text{if } v_k \text{ can switch element } i \text{ in } \Gamma_G^-(v_k) \text{ to element } j \text{ in } \Gamma_G^+(v_k); \\ 0 & \text{otherwise.} \end{cases} \tag{4.2}$$

4.2.2 Construction of a Channel Graph

A Channel graph can be viewed as dual of a network graph. A tuple $\langle e_i, s_j \rangle$ in a network graph will be translated into a node $n_{i,j}^k$ in the channel graph. A node v_i in the network graph will be translated into several arcs in the channel graph. The construction of the channel graph yields a straight-forward $1:1$ mapping of an LSP in network graph into a path in the channel graph.

For each node v_i in network graph G, we generate a node in channel graph H for each element in $\Gamma_G^-(v_i)$ and $\Gamma_G^+(v_i)$, which means that each $\langle e_i, s_j \rangle$ in a network graph can be one-to-one mapped to a node in the channel graph H. Therefore, we have:

$$|N| = \sum_{i=1}^{|V|} (|\Gamma_G^-(v_i)| + |\Gamma_G^+(v_i)|) \tag{4.3}$$

For each node v_i in network graph G, we scan each element in $\Gamma_G^-(v_i)$ and $\Gamma_G^+(v_i)$. If an element $\langle e_m, s_j \rangle$ in $\Gamma_G^-(v_i)$ can be switched to an element $\langle e_p, s_q \rangle$ in $\Gamma_G^+(v_i)$, and $\langle e_m, s_j \rangle$ and $\langle e_p, s_q \rangle$ are mapped to $n_{m,j}^k$ and $n_{p,q}^l$, respectively, a link a_x^i is created in

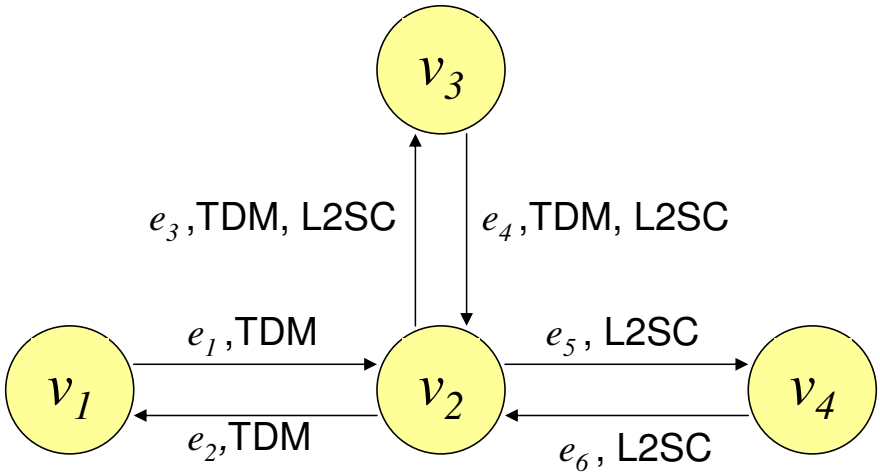

Figure 4.5: A Network Graph Abstracted from the Figure 4.4

channel graph H between $n_{i,j}^k$ and $n_{p,q}^l$. Therefore, we have:

$$|A| = \sum_{i=1}^{|V|} \sum_{j=1}^{|\Gamma_G^-(v_i)|} \sum_{k=1}^{|\Gamma_G^+(v_i)|} (\theta_{j,k}(v_i)) \tag{4.4}$$

We take figure 4.4 as an example to illustrate the whole transformation process. Suppose each connection between nodes in figure 4.4 is bidirectional and node C will advertise both switching types of TDM and L2SC on its links. Therefore, we abstract figure 4.4 as a network graph illustrated in figure 4.5.

In this sample network graph $G = \langle V, E \rangle$, $V = \{v_1, v_2, v_3, v_4\}$ and $E = \{e_1, e_2, e_3, e_4, e_5, e_6\}$. We define s_1 as L2SC and s_2 as TDM, and then we have $\Gamma_G^-(v_2)$=$\{\langle e_1, s_2\rangle, \langle e_4, s_1\rangle, \langle e_4, s_2\rangle, \langle e_6, s_1\rangle\}$, $\Gamma_G^+(v_2)$=$\{\langle e_2, s_2\rangle, \langle e_3, s_1\rangle, \langle e_3, s_2\rangle, \langle e_5, s_1\rangle\}$, and $\Theta_G(v_2) =$

$$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}.$$

Based on figure 4.5, we can generate eight nodes in H, enumerated as N =

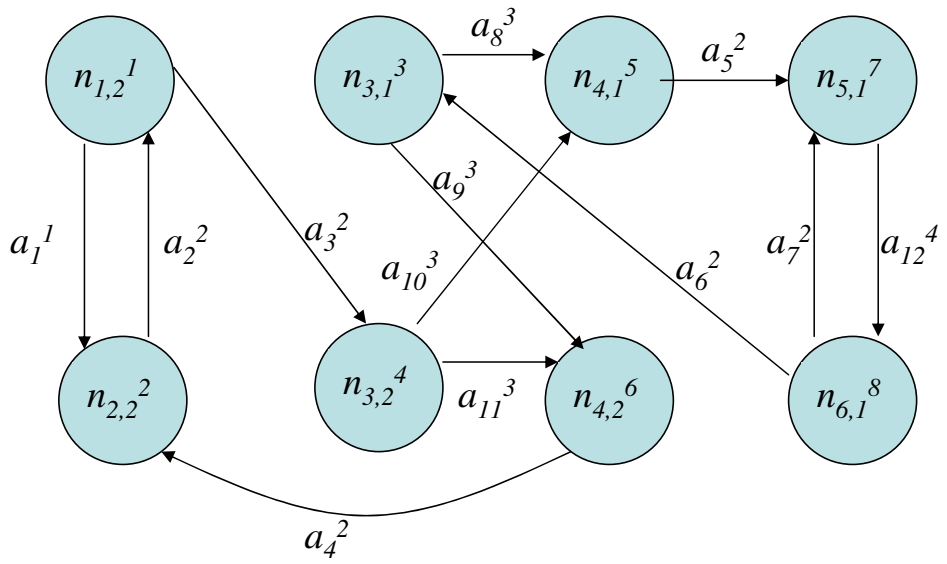

Figure 4.6: A Channel Graph Constructed from the Network Graph as Shown in Figure 4.5

$\{n_{1,2}^1, n_{2,2}^2, n_{3,1}^3, n_{3,2}^4, n_{4,1}^5, n_{4,2}^6, n_{5,1}^7, n_{6,1}^8\}$.

We create links in H based on $\Theta_G(v_i)$ for each $v_i \in V$. The channel graph H transformed from G is illustrated in figure 4.6.

4.3 Cost Modeling and Mapping between a Network Graph and a Channel Graph

The cost structure of using the resources is presented as follows. For each link e_j, variable $w(\langle e_j, s_k \rangle, \lambda_i)$ is given as the "cost" of using wavelength λ_i, switching capability s_k on link e_j.

The cost of wavelength conversion from λ_p to λ_q on node v_i is modeled as $c(v_i, \lambda_p, \lambda_q)$. The cost of interface adaptation from $\langle e_x, s_j \rangle$ to $\langle e_y, s_k \rangle$ at $T(e_x)$ is modeled as $c(T(e_x), \langle e_x, s_j \rangle, \langle e_y, s_k \rangle)$. Here we assume that the cost of wavelength conversion

and interface adaptation are independent.

An LSP between node v_i and v_j is defined as $\Pi_{ij} = \{\pi_{ij}^1, \pi_{ij}^2, \cdots, \pi_{ij}^{h_{ij}}\}$, where π_{ij}^k is a tuple $\langle e_x, s_y \rangle$ and $T(\pi_{ij}^n) = H(\pi_{ij}^{n+1})$ for $n = 1, \cdots, (h_{ij} - 1)$.

The cost of an LSP Π_{ij} is defined as $C(\Pi_{ij})$. We have

$$C(\Pi_{ij}) = \sum_{k=1}^{h_{ij}} w(\pi_{ij}^k, \lambda_k) + \sum_{k=1}^{h_{ij}-1} c(T(\pi_{ij}^k), \lambda_k, \lambda_{k+1}) + \sum_{k=1}^{h_{ij-1}} c(T(\pi_{ij}^k), \pi_{ij}^k, \pi_{ij}^{k+1}), \quad (4.5)$$

where h_{ij} is the number of hops that the LSP traverses; $w(\pi_{ij}^k, \lambda_k)$ is the cost that a LSP traverses π_{ij}^k on λ_i; $c(T(\pi_{ij}^k), \lambda_k, \lambda_{k+1})$ is the cost of wavelength translation from λ_k to λ_{k+1} at node $T(\pi_{ij}^k)$; and $c(T(\pi_{ij}^k), \pi_{ij}^k, \pi_{ij}^{k+1})$ is the cost of switching type adaptation from π_{ij}^k to π_{ij}^{k+1} at node $T(\pi_{ij}^k)$.

If we relax the wavelength continuity constraint, (4.5) will be as follows:

$$C(\Pi_{ij}) = \sum_{k=1}^{h_{ij}} w(\pi_{ij}^k) + \sum_{k=1}^{h_{ij-1}} c(T(\pi_{ij}^k), \pi_{ij}^k, \pi_{ij}^{k+1}), \quad (4.6)$$

When we search a minimum cost path in a network graph, we need to transform the problem to be a minimum cost path in a channel graph. Therefore, we need to map the cost $w(\pi_{ij}^k)$ defined in a network graph to $c(n_{x,y}^z)$ in channel graph, where $\pi_{ij}^k = \langle e_x, s_y \rangle$ and z is the index of node $n_{x,y}^z$ in N. The cost $c(T(\pi_{ij}^k), \pi_{ij}^k, \pi_{ij}^{k+1})$ in the network graph is mapped to $c(a_l^{T(\pi_{ij}^k)})$ in the channel graph, where arc $a_l^{T(\pi_{ij}^k)}$ is mapped to node $T(\pi_{ij}^k)$ in the network graph.

4.4 Searching Optimal Path in a Channel Graph

4.4.1 Path Searching in a Channel Graph

The source node ID and destination node ID in a LSP request are node IDs in G. Therefore, to find a path in H, we need to convert node IDs in G into node IDs in H.

Suppose we want to find a path from v_i to v_j with switching capability s_k. The source nodes in H can include any node that can be mapped to an element in $\varepsilon_G^+(v_i, s_k)$.The destination nodes in H can include any node that can be mapped to an element in $\varepsilon_G^-(v_j, s_k)$.

To simplify the path computation, we need to add two virtual nodes $N_s^{'}$ and $N_t^{'}$. We also add virtual links from $N_s^{'}$ to each element in $\varepsilon_G^+(v_i, s_k)$ and from each element in $\varepsilon_G^-(v_j, s_k)$ to $N_t^{'}$ in H. Each virtual link is assigned with metric 0.

We just need to run CSPF to find a path from $N_s^{'}$ to $N_t^{'}$ on H.

Suppose we are seeking a path from v_2 to v_3 in the network graph G in Figure 4.5 with switching capability L2SC, the final channel graph H is the channel graph as shown in Figure 4.7.

Theorem 3. *For any network graph, if there is a path that satisfies the switching type constraint from the source to the destination, it can be found through the channel graph.*

Proof. If there is a path in the network graph, the path can be described as a sequence of directed links $\pi_{ij}^1, \pi_{ij}^2, \cdots, \pi_{ij}^{h_{ij}}$, such that the $T(\pi_{ij}^k) = H(\pi_{ij}^{k+1})$ for $k = 1, 2, \cdots, h_{ij} - 1$. From the above discussion, we can see that all the switching capability parameters and adaptation functionalities which were hidden or aggregated

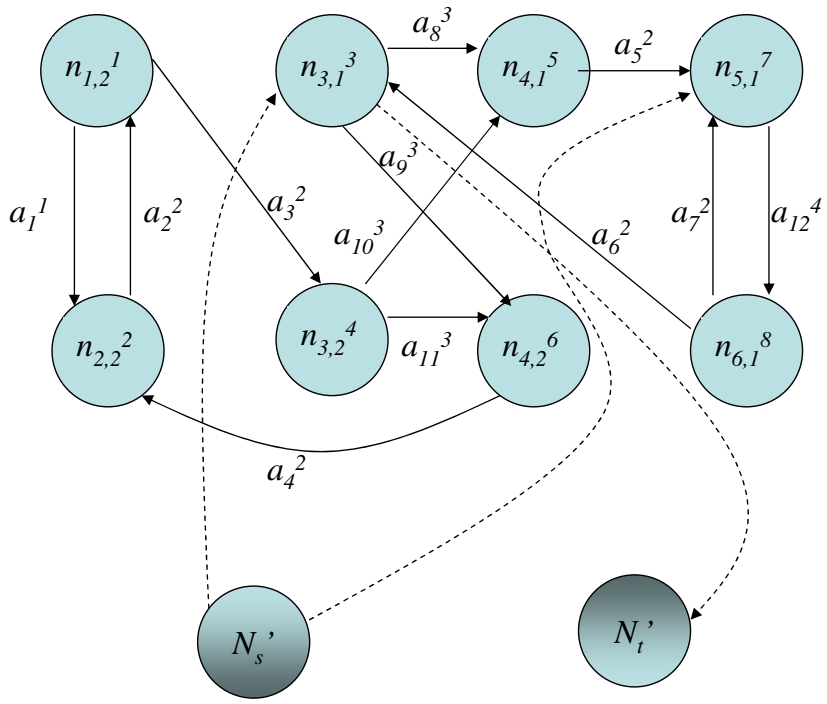

Figure 4.7: Adding Virtual Nodes and Links to Channel Graph

together in the network graph now become explicit and separate nodes or links in the channel graph. Running CSPF on channel graph will easily traverse all these resources to find an optimal path. Because mapping between the network graph and channel grpah is 1 : 1, a path in the channel graph can be mapped back to a path in the network graph due to 1 : 1 mapping. □

4.4.2 Optimality of LSP Searching in MRN

Given the cost modeling equation in (4.6), we have the following theorem.

Theorem 4. *Suppose an intermediate node v_x is on the optimal path Π_{ij}. Π_{ij} does not necessarily contain Π_{ix}, the optimal path between v_i and v_x.*

Proof. Equation (4.6) shows that the cost of an LSP contains not only the cost $w(\pi_{ij}^k)$

to traverse a link, but also the cost $c(T(\pi_{ij}^k), \pi_{ij}^k, \pi_{ij}^{k+1})$ of switching type adaptation.

By definition, we have:

$$C(\Pi_{ix}) = \sum_{k=1}^{h_{ix}} w(\pi_{ix}^k) + \sum_{k=1}^{h_{ix-1}} c(T(\pi_{ix}^k), \pi_{ix}^k, \pi_{ix}^{k+1}), \tag{4.7}$$

, and

$$C(\Pi_{xj}) = \sum_{k=1}^{h_{xj}} w(\pi_{xj}^k) + \sum_{k=1}^{h_{xj-1}} c(T(\pi_{xj}^k), \pi_{xj}^k, \pi_{xj}^{k+1}), \tag{4.8}$$

, and $C(\Pi_{ij})$ can be written as:

$$C(\Pi_{ij}) = \sum_{k=1}^{h_{ix}} w(\pi_{ij}^k) + \sum_{k=1}^{h_{ix-1}} c(T(\pi_{ij}^k), \pi_{ij}^k, \pi_{ij}^{k+1}) + c(v_x, \pi_{ij}^{h_{ix}}, \pi_{ij}^{h_{ix}+1}) + \sum_{k=h_{ix+1}}^{h_{ij}} w(\pi_{ij}^k) + \sum_{k=h_{ix+1}}^{h_{ij-1}} c(T(\pi_{ij}^k), \pi_{ij}^k, \pi_{ij}^{k+1}). \tag{4.9}$$

We define the cost between node v_i and v_j that contains Π_{ix} as $C(\Pi'_{ij})$. Then we have:

$$C(\Pi'_{ij}) = C(\Pi_{ix}) + c(v_x, \pi_{ix}^{h_{ix}}, \pi_{xj}^1) + C(\Pi_{xj}). \tag{4.10}$$

It is clear by definition of Π_{ix} and Π_{xj} that:

$$C(\Pi_{ij}) - c(v_x, \pi_{ij}^{h_{ix}}, \pi_{ij}^{h_{ix}+1}) \leq C(\Pi'_{ij}) - c(v_x, \pi_{ix}^{h_{ix}}, \pi_{xj}^1). \tag{4.11}$$

However, the switching type on $\pi_{ix}^{h_{ix}}$ may be selected a few hops before, and without considering π_{xj}^1. Therefore, cost $c(v_x, \pi_{ix}^{h_{ix}}, \pi_{xj}^1) \geq 0$ or even in an extreme case that $c(v_x, \pi_{ix}^{h_{ix}}, \pi_{xj}^1)$ can be infinity if node v_x does not have the switching type

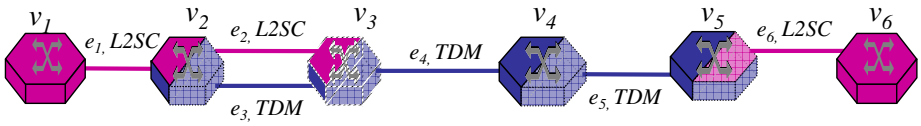

Figure 4.8: An Illustration of Theorem 4

adaptation capability between $\pi_{ix}^{h_{ix}}$ and π_{xj}^{1}. On the other hand, $c(v_x, \pi_{ij}^{h_{ix}}, \pi_{ij}^{h_{ix}+1})$ is selected by taking all the subsequent hops into account. Therefore, we have $c(v_x, \pi_{ij}^{h_{ix}}, \pi_{ij}^{h_{ix}+1}) \leq c(v_x, \pi_{ix}^{h_{ix}}, \pi_{xj}^{1})$. And therefore, the relationship between $C(\Pi_{ij})$ and $C(\Pi'_{ij})$ cannot be determined. □

Figure 4.8 illustrated an example of theorem 4. Suppose node v_1, v_4 and v_6 are single-switching-capable nodes. Node v_2 and v_5 are hybrid nodes that can adapt between L2SC and TDM. Node v_3 is a simplex node. Suppose the metric on each link is 1 and the adaptation cost between L2SC and TDM is 10.

Clearly the optimal cost path between node v_1 and v_3 is $\{\langle e_1, L2SC\rangle, \langle e_2, L2SC\rangle\}$. But the optimal path between node v_1 and v_6 is $\{\langle e_1, L2SC\rangle, \langle e_3, TDM\rangle, \langle e_4, TDM\rangle, \langle e_5, TDM\rangle, \langle e_6, L2SC\rangle\}$. Though v_3 is on the optimal path between v_1 and v_6, but $\Pi_{1,6}^2$ take a different path between v_2 and v_3 from $\Pi_{1,3}^2$.

Theorem 5. *The $C(\Pi_{ij})$ can be determined by knowing the minimal cost from v_i to all the elements in $\Gamma_G^-(v_j)$.*

Proof. The proof of this theorem is straightforward. Theorem 4 clearly shows that we cannot determine the optimal path to the destination node v_j by checking all the optimal path to the adjacent nodes of v_j.

We denote the k^{th} element in $\Gamma_G^-(v_j)$ as $\gamma_{G,k}^-(v_j)$ and the minimal cost from v_i to $\gamma_{G,k}^-(v_j)$ is $C(v_i, \gamma_{G,k}^-(v_j))$. Because the last hop of the optimal path must be

an element in $\Gamma_G^-(v_j)$, when we know the minimal cost from v_i to $\gamma_{G,k}^-(v_j)$, the minimal cost between v_i and v_j will be $\min\{C(v_i, \gamma_{G,k}^-(v_j)) + w(\gamma_{G,k}^-(v_j))\}$ for $k = 1, \cdots, |\Gamma_G^-(v_j)|$. □

We denote the cost to vertex n_j as $d[n_j]$. We define a queue as Q. Any vertex that is reachable from n_i, but the optimal path to which has not yet been determined will be added to Q. We denote $P[n_j]$ to store the cost optimal path from n_i to n_j. Each node has a flag $f[n_j]$ to mark optimal path to n_j has been found or not. We denote $w[H(a_j)]$ as the link metric in G that maps to a_j in H, and $w[a_j]$ as the adaptation cost.

Based on Theorem 5, we need to modify the Dijkstra algorithm as in table 4.1.

Theorem 6. *The modified Dijkstra algorithm given in table 4.1 will find the cost optimal path from $N_s^{'}$ to $N_t^{'}$.*

Proof. This is equivalent to prove that when we make any change in Q, this change cannot reduce the cost to any node n_k such that $f[n_k]$="found".

The modified Dijkstra is different from the conventional Dijkstra at line 9 and 20 in table 4.1. Though the calculation of path cost is added, we still guarantee that the minimal cost path to each n_k is found, which can be proved in the same way as the conventional Dijkstra. Because the cost of all the a_j such that $a_j \in \Gamma_H^-(N_t^{'})$ is 0, by Theorem 5, we know that the path cost from $N_s^{'}$ to $N_t^{'}$ is optimal. □

Table 4.1: Modified Dijkstra Algorithm for Channel Graph Solution

1. BEGIN
2. For each node n_j in H
3. $d[n_j] := \infty$
4. $P[n_j] := null$
5. $f[n_j] :=$ "Not Found"
6. End For
7. $d[N_s^{'}] := 0$
8. For each link a_j in $\varepsilon_H^+(N_s^{'})$
9. $d[T(a_j)] := w(a_j) + w[H(a_j)]$
10. add $T(a_j)$ to Q
11. End For
12. sort Q according to $d[n_j]$ for $n_j \in Q$
13. $n_k :=$ head of Q
14. $f[n_k] :=$ "found"
15. $P[n_k] := N_s^{'}$
16. Remove head of Q
17. while Q is not empty
18. For each link a_j in $\varepsilon_H^+(n_k)$
19. if $d[n_k] + w(a_j) + w[H(a_j)] < d[T(a_j)]$
20. $d[T(a_j)] := d[n_k] + w(a_j) + w[H(a_j)]$
21. if $T(a_j)$ is not in Q
22. Add $T(a_j)$ into Q
23. $P[T(a_j)] := P[H(a_j)]$ concatenate a_j
24. End if
25. End if
26. End For
27. sort Q according to $d[n_j]$ for $n_j \in Q$
28. $n_k :=$ head of Q
29. $f[n_k] :=$ "found"
30. Remove head of Q
31. End while
32. END

4.5 Pseudo Code for Channel Graph Solution for Interface Switching Adaptation Constraints

The following algorithm, which we refer to as Channel Graph Solution for Interface Switching Adaptation Constraints (CSISAC), computes the Shortest Paths with switching capability *swcap* from source vertex s to destination vertex t on a channel graph transformed from a network graph.

The following variables are defined:

1. Each element in V has components { *VertexID, InDegree, OutDegree, IncomingArcList, OutgoingArcList* ...}

2. Each element in E has components { *EdgeID, HeadVertexID, TailVertexID, ISCD, IACD, Metric, Bandwidth, Attenuation...*}

3. Each element in N has components { *NodeID, OriginalEdgeID, OriginalSwitchingCap, InDegree, OutDegree, IncomingArcList, OutgoingArcList, Metric, Bandwidth,...*}

4. Each element in A has components {*ArcID, OriginalVertexID, HeadNodeID, TailNodeID, Metric...*}

4.5.1 Step 1 (Translation of a network graph to a channel graph)

Step 1.1 (Create nodes in H)

$p = 0$
For each node v_i in G
  For each element $\langle e_j, s_k \rangle \in \varepsilon^+_G(v_i, s_k)$ with enough bandwidth
    $p = p + 1$
    Generate node $n^p_{j,k}$ in H
    $n^p_{j,k}$.NodeID=p
    $n^p_{j,k}$.OriginalEdgeID= e_j.EdgeID
    $n^p_{j,k}$.OriginalSwitchingCap = s_k
    $n^p_{j,k}$.Metric = e_j.Metric
      ... ...
    End For
  End For
End For

Step 1.2 (Create arcs in H)

$q = 0$
For each node v_i in G
  For each tuple $\langle e_j, s_k \rangle \in \Gamma^-_G(v_i)$
    If v_i can adapt $\langle e_j, s_k \rangle$ to $\langle e_x, s_y \rangle \in \Gamma^+_G(v_i)$ THEN
      $q = q + 1$
      Generate arc a^i_q in H
      a^i_q.ArcID = q
      a^i_q.OriginalVertexID = v_i.VertexID
      $H(a^i_q) = n^p_{j,k}$ in H // p is the index of $n^p_{j,k}$ in H
      $T(a^i_q) = n^p_{x,y}$ in H // p is the index of $n^p_{x,y}$ in H
      Add a^i_q to $\Gamma^+_H(H(a^i_q)))$
      Add a^i_q to $\Gamma^-_H(T(a^i_q)))$
      a^i_q.Metric= Adaptation Cost of $C(v_i, \langle e_j, s_k \rangle, \langle e_x, s_y \rangle)$
      ... ...
    End If
  End For
End For

Step 1.3 (Generate virtual source node and destination node in H)

Add *newSource* to H
For each tuple $\langle e_j, s_k \rangle \in \Gamma_G^+(v_s)$
  Add new arc to H to connect *newSource* to $n_{j,k}^p$
  Assign 0 to the metric of the new arc in H
End For
Add *newDest* to H
For each tuple $\langle e_j, s_k \rangle \in \Gamma_G^+(v_t)$
  Add new arc to H to connect $n_{j,k}^p$ to *newDest*
  Assign 0 to the metric of the new arc in H
End For

4.5.2 Step 2 (Searching path on channel Graph H)

Running the modified Dijkstra algorithm given in table 4.1.

4.5.3 Step 3 (Converting the path found in H to a path in G)

Suppose a path has been found and stored in variable *Path*.

For each path element a_q^i in *Path*
  output $\left\langle H(a_q^i).OriginalEdgeID, H(a_q^i).OriginalSwitchingCap \right\rangle$
End For

4.6 Computational Complexity and Proof of Efficiency for Channel Graph Solution

Computational complexity of step 1 is $O(|N| + \sum_{i=1}^{|V|}(|\Gamma_G^-(v_i)||\Gamma_G^+(v_i)|))$. The Computational complexity of step 2 is $O(|A| + |N|\log(|N|))$. The computational complexity of step 3 is $O(|A|)$. Because we only need to transform network graph to channel

graph once, the step 1 can be done only once for all the LSP setup request. The overall complexity of the channel graph solution is $O(|A| + |N| \log(|N|))$.

Theorem 7. *Given the worst case that the cost of taking* $\langle e_x, s_j \rangle \in \Gamma^+_G(v_i)$ *is dependant on* $\langle e_y, s_k \rangle \in \Gamma^-_G(v_i)$ *for each combination of* v_i, e_x, s_j, e_y *and* s_k*, the channel graph solution is the most efficient solution.*

Proof. In equation (4.6), it is defined that the cost of connection between $\langle e_x, s_j \rangle \in \Gamma^+_G(v_i)$ and $\langle e_y, s_k \rangle \in \Gamma^-_G(v_i)$ is $c(v_i, \langle e_x, s_j \rangle, \langle e_y, s_k \rangle)$. From theorem 5, we know that we can find the cost optimal path by knowing the cost to each $\langle e_x, s_j \rangle$.

In the worst case, suppose $c(v_i, \langle e_x, s_j \rangle, \langle e_y, s_k \rangle)$ has a different value for each combination of v_i, e_x, s_j, e_y and s_k, the cost optimal path can only be determined by checking each combination of $c(v_i, \langle e_x, s_j \rangle, \langle e_y, s_k \rangle)$ such that $T(e_x) = H(e_y) = v_i$.

This requirement can be translated into a graph with $|N|$ nodes to describe all the $\langle e_x, s_j \rangle \in (\Gamma^+_G(v_i) \bigcup \Gamma^-_G(v_i))$ for any $v_i \in G$, and $|A|$ arcs to describe all the connectivity information between $\langle e_x, s_j \rangle$ and $\langle e_y, s_k \rangle$) such that $e_x \in E$, $e_y \in E$ and $T(e_x) = H(e_y)$.

Therefore, the computational complexity is $O(|A| + |N| \log(|N|)))$, which is the complexity of channel graph solution. □

In the real practice, the network is usually not the worst case. There may be only a few hybrid nodes that can do the adaptation. In this scenario, we prove channel graph solution has the same order of complexity as the network graph in a sparse network.

Theorem 8. *In a sparse network* $G = \langle V, E \rangle$ *with* k *hybrid nodes and* $k \ll |V|$*, suppose that* $d^-_G(v_i) = d^+_G(v_i) \doteq (|E|/|V|)$ *and there are* L *switching types in average*

on each incoming or outgoing link of a hybrid node, the channel graph solution has the same order of complexity as the network graph.

Proof. Without losing generality, we denote the fist $|V| - k$ nodes as the simplex or single-switching capable nodes and the last k nodes as the hybrid nodes. The dual of $G = \langle V, E \rangle$ is channel graph $H = \langle N, A \rangle$.

Given there are k hybrid nodes, we have,

$$|N| = k \times \left(\frac{L|E|}{|V|} \right) + (|V| - k) \times \frac{|E|}{|V|}. \tag{4.12}$$

From the $k \ll |V|$, we know that:

$$|N| \doteq |E|. \tag{4.13}$$

For the first $|V|-k$ nodes, $\frac{|E|^2}{|V|^2}$ links will be generated to describe the connectivity. For the k hybrid nodes, $k \times \left(\frac{L^2|E|^2}{|V|^2} \right)$ links will be generated in the channel graph. Therefore, we have,

$$|A| = k \times \left(\frac{L^2|E|^2}{|V|^2} \right) + (|V| - k) \times \left(\frac{|E|^2}{|V|^2} \right). \tag{4.14}$$

Given $k \ll |V|$ and L is usually small, we have,

$$|A| \doteq \frac{|E|^2}{|V|}. \tag{4.15}$$

Because the computational complexity of channel graph is $O(|A| + |N| \log(|N|)))$. Therefore the computational complexity $O(T)$ in our case is:

$$O(T) = \frac{|E|^2}{|V|} + |E| \log |E|. \tag{4.16}$$

Sparse network means that $|E| = O(|V|)$. Therefore,

$$O(T) = O(|E|) + |V| \log |V|, \tag{4.17}$$

which is the computational complexity of Dijkstra algorithm on a network graph. □

4.7 Breadth-first Search (BFS)

Breadth-first Search (BFS) is also a graph search algorithm that begins from a root node. We mark all the nodes as "unvisited "and mark the root node as "visited". We explore all of the root's neighboring nodes. If a neighboring node is "unvisited", then put it into a queue. The head of the queue is popped up and marked as "visited". We then apply BFS for the head node, and continue doing this until all the nodes are "visited".

BFS is an efficient algorithm. For a graph with n nodes and m links, the computational complexity is $O(n + m)$. However, it does not guarantee the optimality of the path. Actually, it finds the path with fewest hops from the root. By this nature, it is not possible to find KSP through BFS algorithm.

For the MRN/MLN, BFS cannot be applied directly because the set of reachable outgoing links of a node is dependent on the switching type of the incoming link. Path searching also depends on which switching type was chosen on a link with multiple switching types.

Therefore, the BFS is slightly modified as follows:

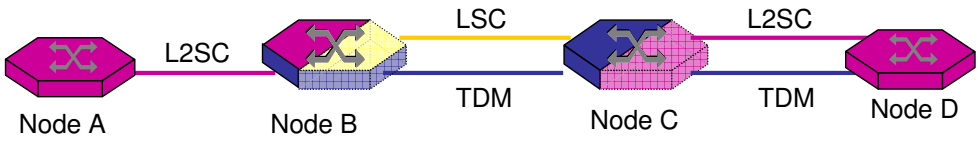

Figure 4.9: A Network on Which BFS May or May not Find a Path

1. For each node the BFS algorithm traverses, the incoming link and its switching type should also be stored;

2. Only those outgoing links that are reachable from the incoming link with the particular switching type should be traversed; and

3. If there are multiple switching types on an outgoing link that could be reached from the incoming link, choose the same switching type on the outgoing links as that on the incoming link if possible. If the switching type on the incoming link is not available on an outgoing link, randomly choose a switching type that can be adapted to from the incoming link.

The above modification clearly shows that the non-elementary path cannot be found by BFS.

Figure 4.9 showed a case that BFS may not find an existing path. Suppose node A is a single switching capable node, node B is a hybrid node that can adapt between L2SC, LSC and TDM, node C is a hybrid node that can adapt between TDM and L2SC, and node D is a simplex node. Now we want to setup a L2SC connection between node A and D. In BFS, if node B translates L2SC between A and B to LSC between B and C, the path searching will fail because node C cannot adapt LSC to TDM or L2SC. Since node B doesn't have any idea which will be the subsequent links and can only choose a switching type between node B and node C randomly, the path

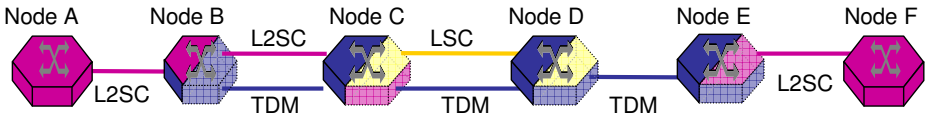

Figure 4.10: Cost-optimal Path Cannot Be Found by BFS

computation algorithm only have 50% chance to find the path. We can assume that in a large network, this probability is further reduced.

Figure 4.10 shows a network scenario where cost-optimal path may not be found by BFS. Suppose only node A and F are single-switch capable nodes and all other nodes are hybrid nodes. If we want to compute a path between node A and F with switching type L2SC, the optimal path could be $< A, B, L2SC > - < B, C, TDM > - < C, D, TDM > - < D, E, TDM > - < E, F, L2SC >$. However, BFS may find a path as $< A, B, L2SC > - < B, C, L2SC > - < C, D, LSC > - < D, E, TDM > - < E, F, L2SC >$.

4.8 Numerical Results

Our simulation is based on figure 4.11, a simplified abstraction of the HOPI network. In this section, we will compare the computational complexity, probability of blocking and number of non-elementary paths of channel graph solution and BFS.

We generate $5 * 10^5$ LSP setup requests and suppose the traffic matrix is homogeneous. Figure 4.12 shows the comparison of the blocking probability between channel graph and BFS solution.

Figure 4.12 shows that channel graph solution can accept more LSP requests than BFS solution. When the traffic intensity is light, the blocking caused by the lack of link capacity is negligible. The dominant reason of blocking in BFS is illustrated in

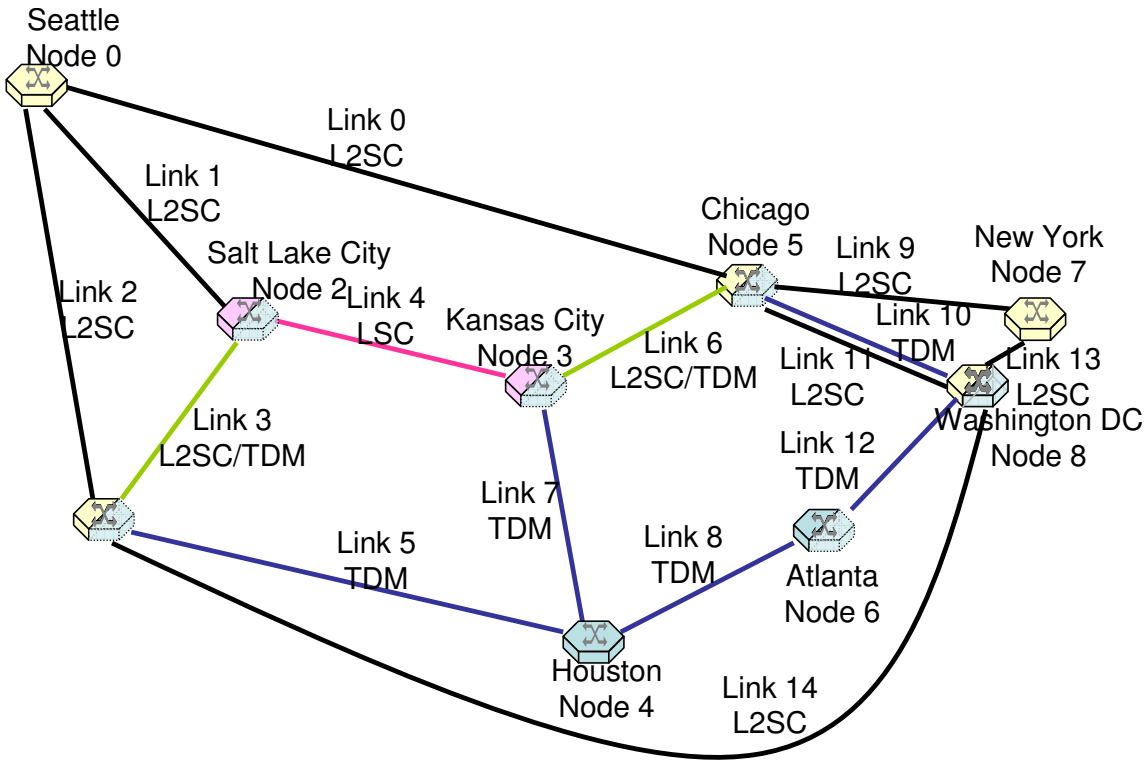

Figure 4.11: A Simplified Abstraction of HOPI Network Topology

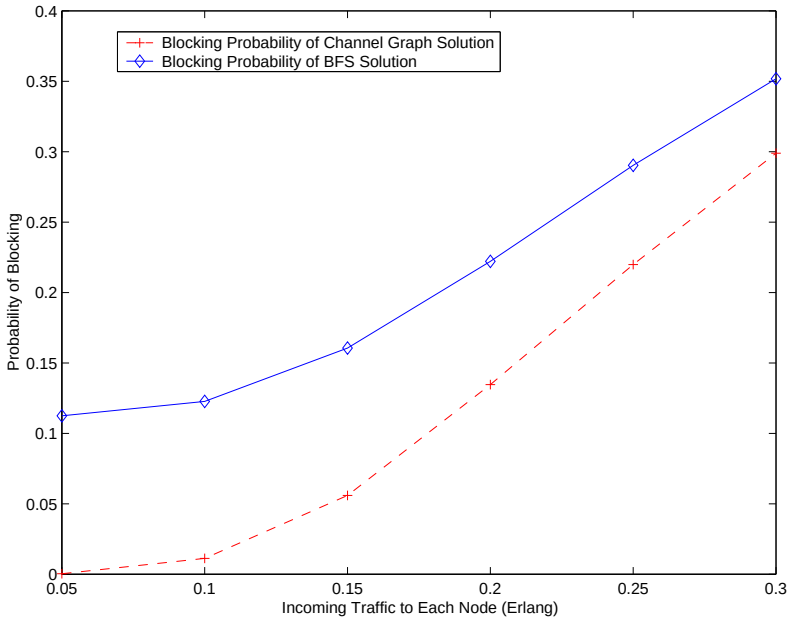

Figure 4.12: Comparison of Blocking Probability Between Channel Graph and BFS Solutions

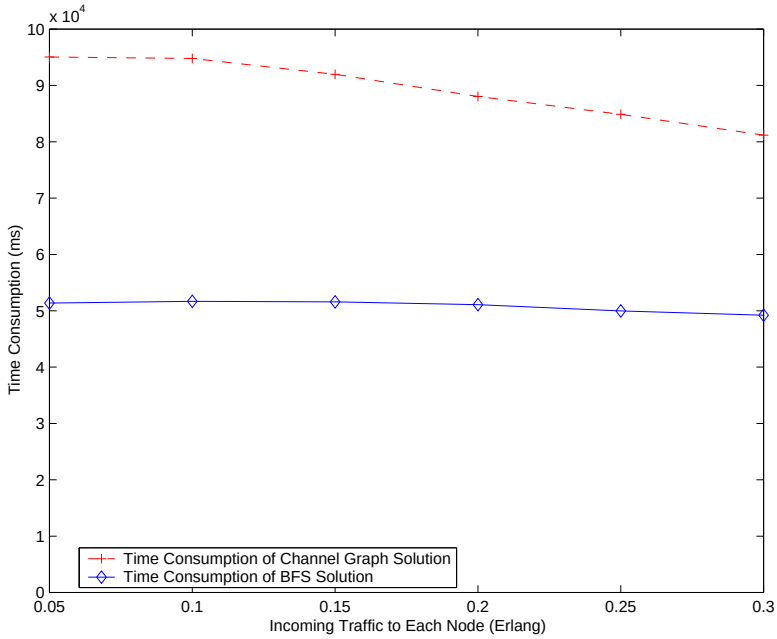

Figure 4.13: Comparison of Time Consumption Between Channel Graph and BFS Solutions

Figure 4.9. When the traffic intensity is moderate or high, the blocking caused by the lack of link capacity is dominant.

Figure 4.13 shows that channel graph solution needs more time than BFS to compute a path. It takes about twice the time as BFS needs. The simulation shows that channel graph is a scalable solution. We can see the decrease of time consumption as the traffic intensity grows. This is partially caused by pruning the link without sufficient bandwidth which makes the graph not connected.

Figure 4.14 compares the average hops that channel graph solution and BFS take. Because BFS is a min-hop algorithm, its average hops is fewer than channel graph solution's. Along with the increase of traffic intensity, the average hop is increased because links along the shortest path may be occupied and the algorithm can only find longer path.

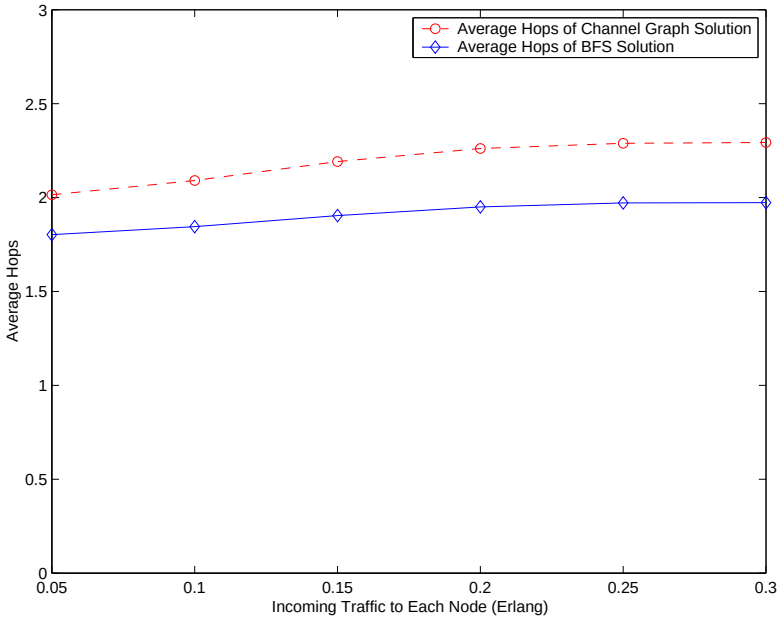

Figure 4.14: Comparison of Average Hops Between Channel Graph and BFS Solutions

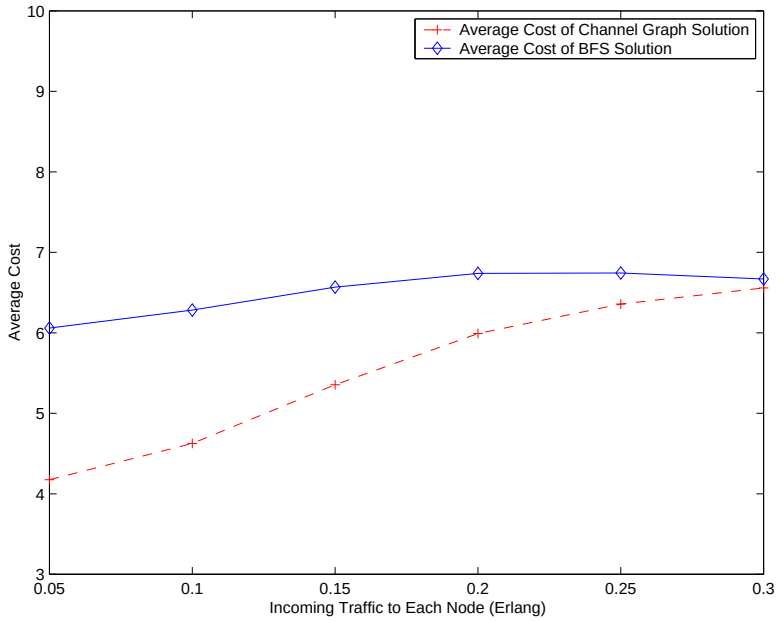

Figure 4.15: Comparison of Average Cost Between Channel Graph and BFS Solutions

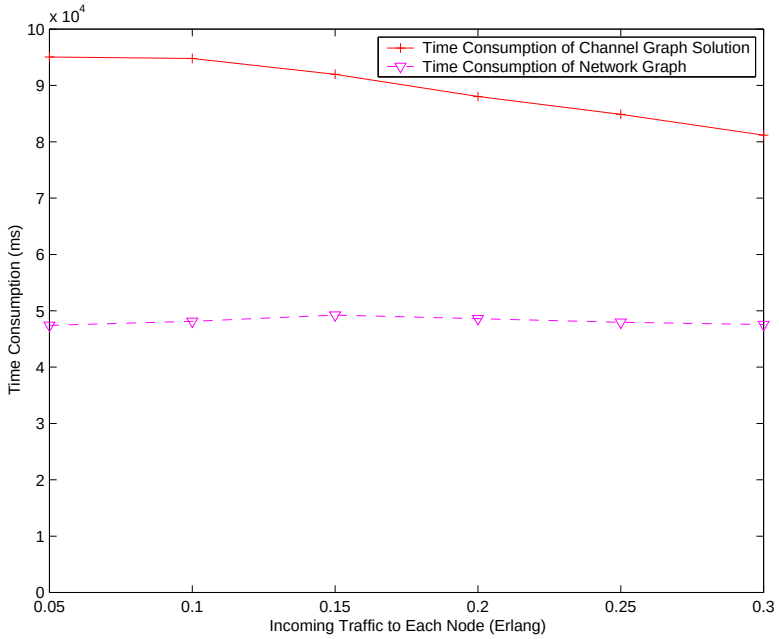

Figure 4.16: Efficiency of Channel Graph Solution

Figure 4.15 compares the average cost that channel graph solution and BFS takes. Because BFS is a min-hop algorithm, it is not a cost optimal solution. Along with the increase of traffic intensity, the average cost is increased because links along the minimal cost path may be occupied and the algorithm can only find a more expensive path.

Theorem 8 has proved that channel graph solution has the same order of efficiency as the network graph. Figure 4.16 verifies this theorem.

4.9 Modification and Complexity of KSP for Channel Graph

In MRN/MLN environment, running CSPF in a Channel Graph to find a path may not be enough to address the optical impairment constraints. Therefore, we need to

run KSP.

If we take Figure 4.7 to run KSP, we can find two paths. The first path will be $N_s^{'} - n_{3,1}^3 - N_t^{'}$. The second path will be $N_s^{'} - n_{5,1}^7 - n_{6,1}^8 - n_{3,1}^3 - N_t^{'}$.

In YEN's KSP algorithm, if $N_s^{'} - n_{3,1}^3 - N_t^{'}$ is the parent path, we need to find alternative path from $n_{3,1}^3$ to $N_t^{'}$. We may find one if the network topology gets complicated enough. However, the newly found path will contain an unnecessary loop. This is because when we reach node $n_{3,1}^3$, we have already reached the destination node. Therefore, we need to modify the KSP algorithm in such a way that if the predecessor of the destination node in channel graph is the incoming link of the destination node with the required switching type in the network graph, we will stop searching alternative path from this predecessor.

The second reason that could generate a loop when we run YEN's algorithm is that we may come back to a node in H whose head node and switching type in network graph G has appeared in the same path. The solution is that when a node in the Channel Graph is deleted, all the nodes (corresponding to a link in the network graph) in Channel Graph having the same head node and switching capability in the network graph must also be deleted.

KSP is used in [21] to find disjoint path with SRLG in GMPLS network. To ensure the survivability of the network, we can also apply Oki's algorithm to channel graph. When we prune links $L(i, j)$ on the network graph, we need to prune all the nodes in channel graph that can be mapped to $L(i, j)$ in the network graph.

The computational complexity is $O(K|N|(|A| + |N|log(|N|)))$ according to YEN's algorithm.

4.10 Summary

In this chapter we discussed the channel graph solution to interface specific adaptation constraints. Network graph needs to be transformed before path computation starts. It is proved that the cost modeling and mapping between the channel graph and the network graph provided in this chapter can guarantee to find the cost-optimal path efficiently. We compared the computational complexity and blocking probability between channel graph and BFS solutions. It is showed that channel graph solution only takes slightly more time than BFS, but it has much better performance from blocking probability aspect.

We prove that in a sparse network with a small number of adaptation nodes, the channel graph has the same order of complexity as the network graph. Numerical results verifies this proof.

KSP needs modification before applying to the channel graph to avoid unnecessary loop.

Chapter 5: Link Performance Bounds in Homogeneous Optically Switched Ring Networks

The estimation of the blocking probability of the common vector solution presented in chapter 3 is based on equation (3.3). To simplify the estimation, we intentionally overlooked the discrepancy between the offered load and carried load on each link, and consequently, the calculated blocking probability turned out to be always greater than the actual blocking ratio. Though precise calculation of the link blocking probability can demonstrate significant improvement on the network performance estimation, the computational complexity is non-polynomial. Therefore, bounds may serve as a useful measure for design, provisioning and performance evaluation purposes, especially if the upper and lower bounds are found to be very tight. In this chapter, we propose a more accurate estimation on the traffic flow intensity on a link in a homogeneous ring network.

We take ring topology as prototype because it is simple, yet provides insights to blocking probability of other complicated topologies. Meanwhile, variants of ring architecture are widely deployed by service providers. The mesh architecture may be developed by extending the existing ring topology to further simplify the interconnection of these networks at the core.

The model discussed in this chapter is based on the lower and upper bounds of link blocking probability. We analyze the performance for a homogeneous traffic case and present simulation results for representative ring networks. We demonstrate that the

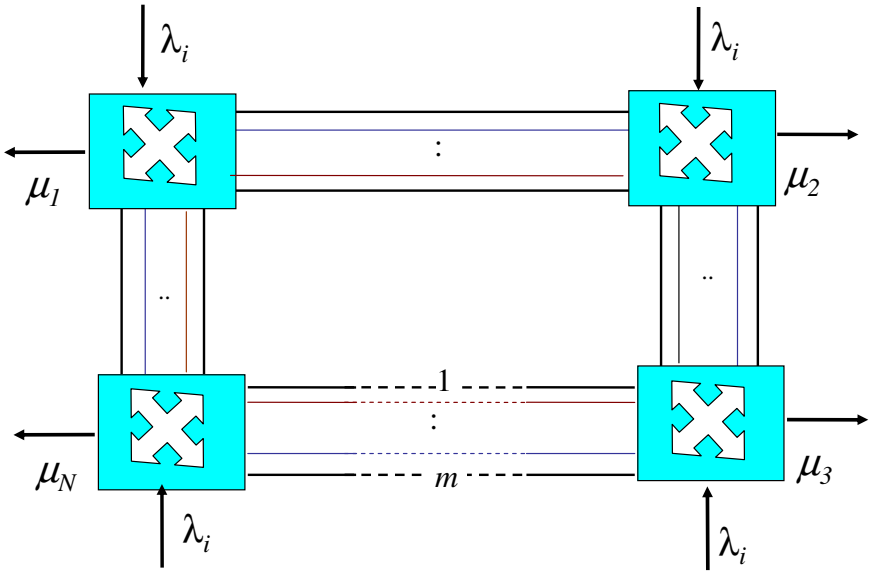

Figure 5.1: Optically Switched Ring Networks

bounds are very tight with an error of less than 2% when the traffic load is modest. The approach is based on partitioning the state space into subspaces and weighting the upper bound of blocking probability in each subspace with the occurrence of the states. The computational complexity of the approach is comparable to solving a degree of N polynomial equation.

This chapter is organized as follows. We consider a ring topology with dynamic optical links interconnecting the network nodes and develop upper and lower bounds to estimate the probability of link blocking. We then presents our model, followed by the derivation of the lower and upper bounds. Numerical results and concluding remarks are provided subsequently.

5.1 System Model

As shown in Figure 5.1, N access nodes are connected to a unidirectional optically switched ring. A fiber on the ring has m wavelengths, and each node j has the same incoming traffic λ_i. We assume that wavelength conversion is unavailable, the traffic demand matrix is homogeneous, and traffic only flows clockwise. Meanwhile, incoming traffic at each access node is a Poisson process, and the traffic is distributed among all the wavelengths randomly with equal probability. If a request cannot be satisfied on the selected wavelength plane, it is rejected. The service time of each request is assumed to follow an exponential distribution.

From the above assumptions, we know that the arrival to each wavelength plane is also a Poisson process. The problem is simplified to determining a blocking probability on each wavelength plane, given the Poisson arrival rate of incoming traffic of λ_i/m at each node.

Blocking probability in various network scenarios have been addressed in [32, 33, 42, 43]. The blocking probability in all-optical networks with and without wavelength changers has been considered in [32] and has been modeled with the assumption that the traffic load is light, an initial estimate of link blocking probability is known, and usage of a wavelength on a hop is statistically independent of other hops as well as other wavelengths. The estimation of fiber utilization ratio in [32] neglects the impact on call arrival rate caused by the blocking. This is reasonable when the link blocking probability is low. However, with high link blocking probability, the assumption will result in over-estimating the path blocking probability. Moreover, by assuming that wavelength seizure and release are independent of each other, the dynamic nature of

the traffic is hidden.

Reference [33] focuses on the optical network with wavelength converters. The traffic is assumed to be bounded by the number of ports in a node, which hides the dynamic nature of the traffic. Reference [42] provides both analytical model and simulation results on call blocking probability in a ring network for very light traffic. This model assumes a homogeneous traffic matrix, Markovian correlation of blocking at adjacent links, and certain regular topologies. The computational complexity is modest.

More recently, a computational model for estimating blocking probability in a multi-fiber WDM optical network has been presented in [43] where the entire wavelength channel is dedicated to a single connection. Our work deals with dynamically provisioned networks, where even if a wavelength is occupied on some segments of a network, it can still be reused wherever possible, hence considerably reducing the blocking probability.

With Fiber to the Home (FTTH), the arrival rate at an edge node will become high and blocking probability may be much greater than that in the core network. Estimation of blocking probability in a network with arbitrary traffic intensity is a hard problem in that the correlation of blocking probability on different links makes precise computation of carried load impossible.

We note that link blocking probability is usually the basis to compute call blocking. This will be the focus of this chapter and is equally applicable to [32] and [42]. It can further be extended to model waveband switching or fiber switching.

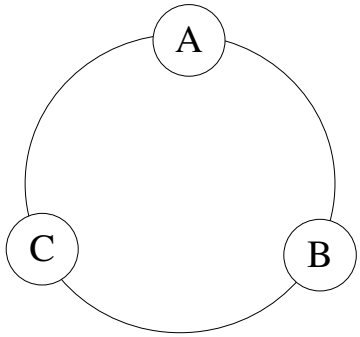

Figure 5.2: A Ring with Three Nodes

Table 5.1: List of Link States

Link		Occupied by path	Occupied by path	Occupied by path
AB	1. Idle	2. From A to B	3. From A to C	4. From C to B
BC	5. Idle	6. From B to C	3. From A to C	7. From B to A
CA	8. Idle	9. From C to A	4. From C to B	7. From B to A

5.2 State Transition Diagram of a Unidirectional Ring Topology

5.2.1 A Simple Example

Suppose we have 3 nodes in a unidirectional ring. Each link has only one wavelength. Figure 5.2 shows that both link AB and BC being busy can be caused by two scenarios, i.e., AB is occupied by a lightpath from A to B and BC is occupied by a lightpath from B to C or AB and BC is occupied by a lightpath from A to C, which means that we should not only note whether a link is busy, but also note the source-destination pair which occupies the link.

We can list the states as shown in Table 5.1. Therefore, we can find 14 different states for a three-node ring. We define S_1 as the scenario that all links are idle. S_1 can be described as $S_1 = (1, 5, 8)$, where the three indices in the parentheses are

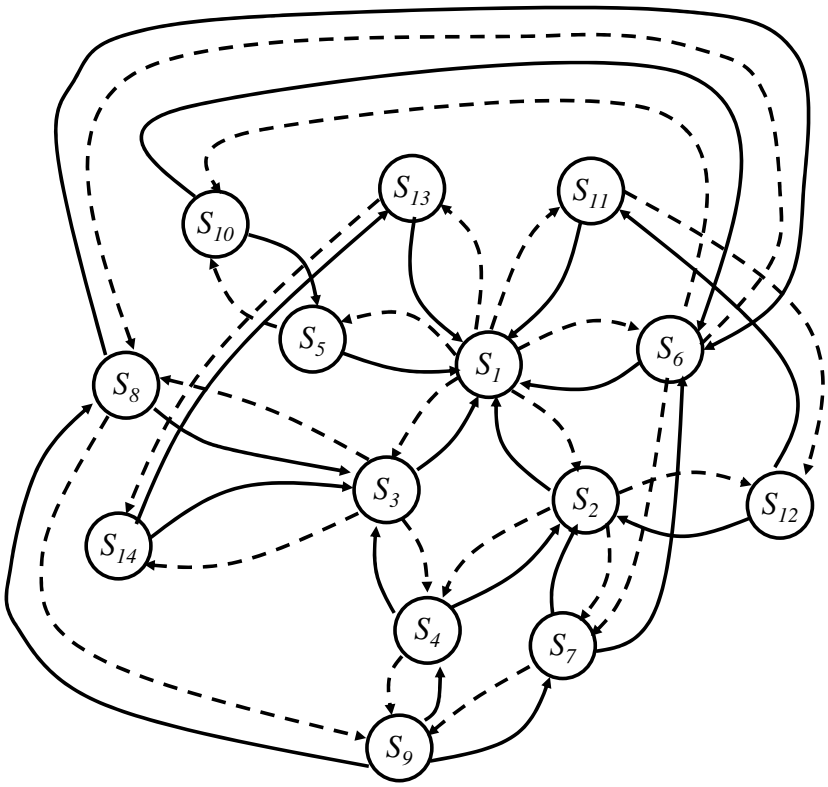

Figure 5.3: State Transition Diagram of a Three-node Ring. Each Arrow Represents Transition Rate of $\lambda/2$, and Each Arrow with Dashed Line Represents Transition Rate of μ.

defined in Table I. Similarly, we define $S_2 = (1, 5, 9)$, $S_3 = (1, 6, 8)$, $S_4 = (1, 6, 9)$, $S_5 = (1, 7, 7)$, $S_6 = (2, 5, 8)$, $S_7 = (2, 5, 9)$, $S_8 = (2, 6, 8)$, $S_9 = (2, 6, 9)$, $S_{10} = (2, 7, 7)$, $S_{11} = (3, 3, 8)$, $S_{12} = (3, 3, 9)$, $S_{13} = (4, 5, 4)$ and $S_{14} = (4, 6, 4)$.

We assume that the traffic demand matrix is homogeneous, the incoming traffic at each node is λ, the departure rate of each lightpath request is μ. The state transition diagram of Figure 5.2 can be depicted as in Figure 5.3.

	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}
S_1	-3λ	μ	μ	0	μ	μ	0	0	0	0	μ	0	μ	0
S_2	λ/2	-μ-1.5λ	0	μ	0	0	μ	0	0	0	0	μ	0	0
S_3	λ/2	0	-μ-1.5λ	μ	0	0	0	μ	0	0	0	0	0	μ
S_4	0	λ/2	λ/2	-2μ-0.5λ	0	0	0	0	μ	0	0	0	0	0
S_5	λ/2	0	0	0	-μ-0.5λ	0	0	0	0	μ	0	0	0	0
S_6	λ/2	0	0	0	0	-μ-1.5λ	μ	μ	0	μ	0	0	0	0
S_7	0	λ/2	0	0	0	λ/2	-2μ-0.5λ	0	μ	0	0	0	0	0
S_8	0	0	λ/2	0	0	λ/2	0	-2μ-0.5λ	μ	0	0	0	0	0
S_9	0	0	0	λ/2	0	0	λ/2	λ/2	-3μ	0	0	0	0	0
S_{10}	0	0	0	0	λ/2	λ/2	0	0	0	-2μ	0	0	0	0
S_{11}	λ/2	0	0	0	0	0	0	0	0	0	-μ-0.5λ	μ	0	0
S_{12}	0	λ/2	0	0	0	0	0	0	0	0	λ/2	-2μ	0	0
S_{13}	λ/2	0	0	0	0	0	0	0	0	0	0	0	-μ-0.5λ	μ
S_{14}	0	0	λ/2	0	0	0	0	0	0	0	0	0	λ/2	-2μ

Figure 5.4: State Transition Matrix in a Three-node Ring

Table 5.2: Comparison of Caculated and Simulated Probability of Blocking P_b

λ/μ	0.1	0.2	0.3	0.4	0.5	0.6
Caculated P_b	0.1237	0.2113	0.2776	0.3301	0.3730	0.4090
Simulated P_b	0.1250	0.2139	0.2784	0.3336	0.3723	0.4157

Based on Figure 5.3, we can obtain the state transition matrix as shown in Figure 5.4. We can obtain the probability of each state in the following vector:

$$S' = \left[\frac{8}{D}\ \frac{4r}{D}\ \frac{4r}{D}\ \frac{2r^2}{D}\ \frac{4r}{D}\ \frac{4r}{D}\ \frac{2r^2}{D}\ \frac{2r^2}{D}\ \frac{r^3}{D}\ \frac{2r^2}{D}\ \frac{4r}{D}\ \frac{2r^2}{D}\ \frac{4r}{D}\ \frac{2r^2}{D}\right], \tag{5.1}$$

where $D = r^3 + 12r^2 + 24r + 8$, and $r = \lambda/\mu$.

According to (5.1), we can easily calculate the probability that link AB is busy as:

$$P_b = \sum_{k=6}^{14} S_k. \tag{5.2}$$

5.2.2 Partitioning the state space

A three-node unidirectional ring with homogeneous traffic demand is the simplest scenario. However, calculating the precise link blocking probability is complicated. Generally, for a ring with N nodes, we need $O(N^N)$ network states, and since we need to compute the transition rate for any state to any other state, an $O(N^N \times N^N)$ matrix results. Even if N is small, the matrix becomes intractable due to its non-polynomial complexity. A precise state transition diagram to obtain the blocking probability is not possible when N is large.

When the traffic demand matrix is homogeneous, the blocking probability on each link is also the same. For a ring with N links denoted as $(l_1, l_2, \cdots, l_N)$, we focus on the blocking probability on l_1.

The basic idea of the proposed algorithm is straightforward. The arrival rate on l_1 is dependent on the current network states. We assume that there are K network states, and the flow rate on l_1 of state k is I_k. Without loss of generality, we assume the departure rate of all the states is one per time unit. We denote p_k as the probability that the network is in state k. The blocking probability of link l_1 is:

$$P_b = \sum_{k=1}^{K} \frac{I_k}{1 + I_k} p_k. \tag{5.3}$$

When all links are idle, the flow rate on l_1 is the maximum. When link l_N is busy, the flow rate on l_1 is reduced. If both l_N and l_2 are busy, the flow rate on l_1 is further reduced. We denote the network state space as Ω.

The proposed algorithm in this chapter simplifies the state transition diagram by

dividing Ω into 3 subspaces as follows:

$$S_0 = \Omega - S_1 - S_2. \tag{5.4}$$

$$S_1 = \{l_N \text{ busy and } l_2 \text{ idle}\}. \tag{5.5}$$

$$S_2 = \{l_N \text{ busy and } l_2 \text{ busy}\}. \tag{5.6}$$

We use the maximum flow rate I_{sk} among all the states $s_{km} \in S_k$ as the representative flow rate in S_k, and denote p_{sk} as the probability that the network is in state S_k, and define $P_{bu}^{Sk} = I_{sk}/(1 + I_{sk})$. Therefore, we have

$$\sum_{k=0}^{2} (P_{bu}^{Sk} p_{sk}) > P_b. \tag{5.7}$$

Equation (5.7) can not be used directly because it is of the same complexity to calculate the precise p_{sk}. Further approximation is required.

5.3 Lower Bound of Blocking Probability

Suppose the blocking probability on each link is P_b. We offer the following approach to compute the lower bound of blocking probability on a link. In all subsequent analysis, we assume the traffic matrix is homogeneous with newly generated traffic intensity of λ at each node. The ring is unidirectional and traffic flows clockwise.

Theorem 9. *In a ring with N nodes, if node A is k hops away from the head node of link δ, the traffic flow $I(k)$ from A on the link δ satisfies*

$$I(k) \geq \frac{\lambda((1 - P_b)^k - (1 - P_b)^{N-1})}{(N - 1)P_b}. \tag{5.8}$$

Proof. The prerequisite for a successful lightpath setup over link δ from node A is that all the links along the path other than δ to be idle. Only path that is longer than k will pass traffic through link δ. Because each link is busy with probability P_b, the probability of successful lightpath setup from A can be approximated as $(1 - P_b)^m$ where $m + 1$ is the number of hops from node A to the destination. The maximal hop is $(N - 1)$ in that a node will never setup a lightpath to loop back. The traffic flow β from a node to any other node is identical, and therefore

$$\beta = \lambda/(N - 1). \tag{5.9}$$

Thus, the overall traffic flow from node A on link δ is approximately

$$\sum_{m=k}^{N-2} \beta(1 - P_b)^m = \frac{\lambda((1 - P_b)^k - (1 - P_b)^{N-1})}{(N - 1)P_b}. \tag{5.10}$$

When a link is busy, the link after this busy link is likely to be busy because the traffic flows clockwise. Due to this dependency on link blockage, the probability that k links are idle will be greater than $(1 - P_b)^k$. □

Corollary 1: From Theorem 9, the overall traffic flow Λ on a link in the ring topology given homogeneous traffic demand satisfies

$$\Lambda = \sum_{k=0}^{N-2} I(k) \geq \frac{\lambda(1 - (1 - P_b)^{N-1}(1 + P_b(N - 1)))}{(N - 1)P_b^2}. \tag{5.11}$$

Theorem 10. *We assume the arrival process is Poisson and the departure process is exponential with mean service time* $1/\mu$ *. The lower bound of link blocking probability* P_b *can be computed by substituting* Λ *with the lower bound of equation* (5.11) *and by*

defining $\rho = \Lambda/\mu$ *to solve the equation*

$$\frac{\rho}{1+\rho} = P_b. \tag{5.12}$$

Proof. The blocking probability on a particular link δ can be approximately computed according to M/M/1/1 as

$$\frac{\rho}{1+\rho} = P_b^{'}. \tag{5.13}$$

Given P_b, $P_b^{'} \leq P_b$ because the left side of (5.13) is a monotonic increasing function, and we substituted ρ with its lower bound. However, P_b is also unknown. Because ρ is a monotonic decreasing function of P_b, (5.13) is also a monotonic decreasing function of P_b. Therefore, when $P_b^{'} \leq P_b$, the solution of P_b to (5.12) is strictly smaller than the precise value of P_b. □

We denote the calculated result from (5.12) as P_{bl}.

5.4 Upper Bound of Blocking Probability

When all links are idle, the flow I_{s0} on l_1 is the maximum.

Theorem 11. *In a ring with* N *nodes,*

$$I_{s0} = N\lambda/2. \tag{5.14}$$

Proof. If a node A is k hops away from the head node of l_1, the probability that A imposes traffic flow on l_1 will be the left hand side of (5.10) with $P_b = 0$. Therefore,

The total traffic flow on l_1 will be

$$\sum_{k=0}^{N-2}\sum_{m=k}^{N-2}\frac{\lambda}{N-1}=\frac{N\lambda}{2}. \tag{5.15}$$

□

The upper bound of link probability can never exceed

$$P_b^U=\frac{N\lambda/2}{1+N\lambda/2}=P_{bu}^{S0}. \tag{5.16}$$

Equation (5.16) gives the simplest approach to calculate the upper bound of link blocking probability. However, we can find tighter upper bound by the following steps.

5.4.1 Lower Bound of p_{S2}

According to (5.6), p_{s2}=P{l_N busy and l_2 busy}. Therefore, we have

$$p_{s2}\geq P\{l_N \text{ busy}\}P\{l_2 \text{ busy}\}=P_b^2\geq P_{bl}^2. \tag{5.17}$$

The lower bound of p_{S2} can be calculated as $p_{S2}^l=P_{bl}^2$.

5.4.2 Lower Bound of $P(S_1\cup S_2)$

We know that $S_1\cup S_2=\{l_N \text{ busy}\}$ and $S_1\cap S_2=\Phi$. Therefore, $P_{S1}+P_{S2}=P_b\geq P_{bl}$. The lower bound of $P(S1\cup S2)$ is P_{bl}.

5.4.3 Upper Bound of Link Blocking Probability

In S_2, the flow rate on l_1 can not exceed $\lambda/(N-1)$. This is because only the traffic flow from the head node of l_1 to the tail node of l_1 can be accepted. Therefore, we have

$$P_{bu}^{S2} = \frac{\lambda/(N-1)}{1+\lambda/(N-1)}. \tag{5.18}$$

In S_1, the flow rate on l_1 cannot exceed λ. Therefore, we have

$$P_{bu}^{S1} = \frac{\lambda}{1+\lambda}. \tag{5.19}$$

Because $p_{S0} + p_{S1} + p_{S2} = 1$, we have

$$p_{S0} \leq 1 - P_{bl}. \tag{5.20}$$

Theorem 12. *The upper bound of link blocking probability P_{bu} can be calculated as below:*

$$P_{bu} = (1-P_{bl})P_{bu}^{S0} + (P_{bl} - P_{bl}^2)P_{bu}^{S1} + P_{bl}^2 P_{bu}^{S2}. \tag{5.21}$$

Proof. S_0, S_1, and S_2 is a partition of the set of all network states. Therefore, from (5.3), we have

$$P_b < \sum_{k\in S_0} p_k P_{bu}^{S0} + \sum_{k\in S_1} p_k P_{bu}^{S1} + \sum_{k\in S_2} p_k P_{bu}^{S2} = \sum_{k=0}^{2} p_{sk} P_{bu}^{Sk}. \tag{5.22}$$

From (5.20) and $P_{bu}^{S0} > P_{bu}^{S1}$, we have

$$\sum_{k=0}^{2} p_{sk} P_{bu}^{Sk} < (1-P_{bl})P_{bu}^{S0} + (P_{bl} - p_{S2})P_{bu}^{S1} + p_{S2}P_{bu}^{S2}. \tag{5.23}$$

Because $p_{S2} \geq P_{bl}^2$, we have

$$P_b < (1 - P_{bl})P_{bu}^{S0} + (P_{bl} - P_{bl}^2)P_{bu}^{S1} + P_{bl}^2 P_{bu}^{S2} = P_{bu}, \tag{5.24}$$

where P_{bl} can be calculated according to (5.12). □

5.5 Extension to $M/M/m/m$ Model

If we relax the wavelength continuity constraint, and assume a call setup request can be accepted as long as there is enough bandwidth along the determined LSP, the $M/M/m/m$ model can be applied..

Theorem 9 and Corollary 1 will still hold. Theorem 10 can be extended as follows:

We assume that there are w wavelength in each fiber. The lower bound of link blocking probability P_b can be computed by substituting Λ with the lower bound of equation (5.11) and by defining $\rho = \Lambda/\mu$ to solve the equation

$$\frac{\rho^w/w!}{\sum_{k=0}^{w} \rho^k/k!} = P_b. \tag{5.25}$$

The left side of equation (5.25) is the Erlang B formula and it is also a monotonic increasing function. Equation (5.16) in $M/M/m/m$ becomes

$$P_b^U = \frac{I_{s0}^w/w!}{\sum_{k=0}^{w} I_{s0}^k/k!} = P_{bu}^{S0}. \tag{5.26}$$

5.6 Simulation versus Calculation Results

In our simulation, we assume that we are dealing with a unidirectional WDM network with 10 wavelengths in a fiber as shown in Figure 5.1. The arrival of incoming call

Table 5.3: Caculated Bounds in a Six-Node Ring

λ	0.001	0.0025	0.005	0.0075	0.01
P_{bl}	0.002967	0.007301	0.01423	0.02084	0.02714
P_{bu}	0.002985	0.007408	0.01464	0.02171	0.02861
P_{bu}^{S0}	0.002991	0.007444	0.01478	0.02200	0.02913

Table 5.4: Caculated versus Simulated Bounds in a Three-Node Ring

λ	0.1	0.2	0.3	0.4	0.5	0.6
P_{bl}	0.1212	0.2056	0.2696	0.3206	0.3625	0.3980
P_{bc}	0.1237	0.2113	0.2776	0.3301	0.3730	0.4090
P_{bu}	0.1250	0.2144	0.2816	0.3341	0.3765	0.4116

to each node is a Poisson process at a rate of 10λ per second. The distribution of the service rate is exponential with the mean of one per second. We also assume the path computation element will randomly pick up a wavelength plane and run CSPF on that plane. The traffic demand matrix is homogeneous.

This model is to calculate the upper bound and lower bound of blocking probability when the discrepancy between carried load and offered load is not negligible. When the traffic load is light, P_{bu}^{S0} can sever as a good upper bound. Table 5.3 provides the comparison of calculated lower bound P_{bl}, calculated upper bound P_{bu} and calculated P_{bu}^{S0} in a six-node ring.

Table 5.4 provides the comparison of calculated lower bounds P_{bl}, upper bounds P_{bu} and the actual blocking probability P_{bc} when λ is increased from 0.1 to 0.6 at a

Table 5.5: Caculated versus Simulated Bounds in a Four-Node Ring

λ	0.1	0.2	0.3	0.4	0.5	0.6
P_{bl}	0.1412	0.2249	0.2841	0.3297	0.3665	0.3973
P_{bs}	0.1543	0.2483	0.3108	0.3604	0.4001	0.4335
P_{bu}	0.1548	0.2537	0.3227	0.3738	0.4133	0.4448

Table 5.6: Caculated versus Simulated Bounds in a Five-Node Ring

λ	0.1	0.2	0.3	0.4	0.5	0.6
P_{bl}	0.1537	0.2336	0.2879	0.3289	0.3619	0.3894
P_{bs}	0.1806	0.2785	0.3412	0.3895	0.4255	0.4517
P_{bu}	0.1817	0.2879	0.3583	0.4084	0.4460	0.4753

Table 5.7: Caculated versus Simulated Bounds in a Six-Node Ring

λ	0.1	0.2	0.3	0.4	0.5	0.6
P_{bl}	0.1614	0.2367	0.2867	0.3243	0.3543	0.3794
P_{bs}	0.2002	0.3001	0.3599	0.4072	0.4420	0.4653
P_{bu}	0.2063	0.3185	0.3897	0.4390	0.4751	0.5027

step of 0.1. It is also depicted in Fig. 5.5.

Table 5.5 to Table 5.7 provides the comparison of calculated lower bounds P_{bl}, upper bounds P_{bu}, and the blocking probability P_{bs} from simulation when λ is increased from 0.1 to 0.6 at a step of 0.1. Table 5.5 is also depicted in Fig. 5.6.

From the simulation, we can observe that the calculated upper bound is very tight. When the traffic intensity is modest, the error is less than 2%.

Fig. 5.9 depicts the error of P_{bl} and P_{bu} when the number of node and traffic intensity increase. We can observe that the error increases along with the node number and traffic intensity.

If we relax the wavelength continuity constraint, the system model will be as described in 5.5.

Table 5.8: Error as the Number of Nodes and Traffic Intensity Increase

Number of Nodes	4	5	6
% Error of P_{bl} when $\lambda = 0.1$	8.49	14.89	19.38
% Error of P_{bl} when $\lambda = 0.2$	9.42	16.12	21.13
% Error of P_{bu} when $\lambda = 0.1$	0.32	0.61	3.05
% Error of P_{bu} when $\lambda = 0.2$	2.17	3.38	6.13

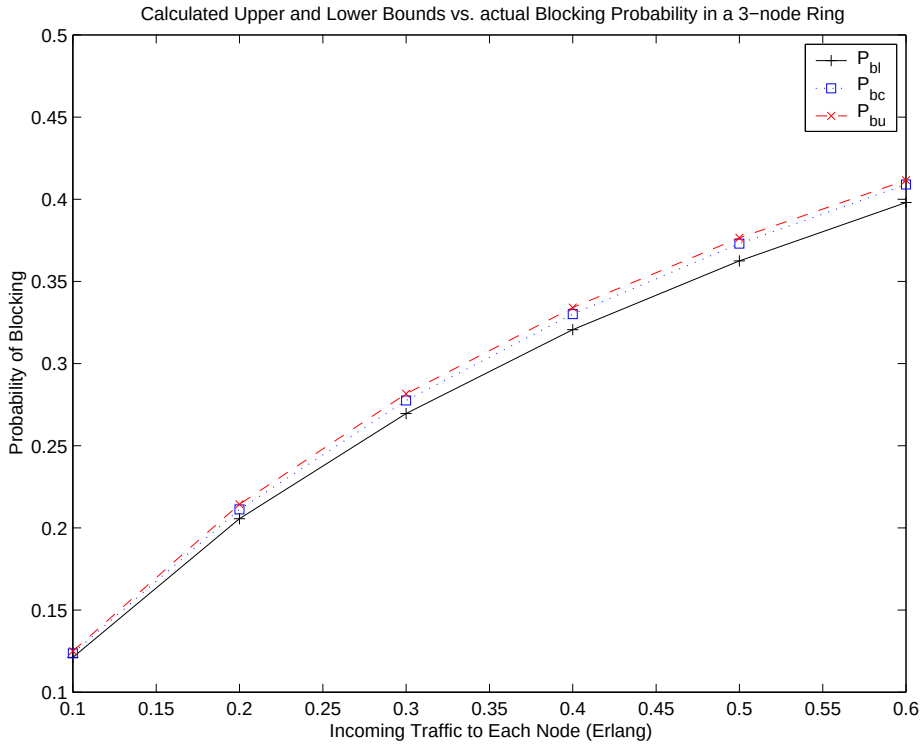

Figure 5.5: Calculated Upper and Lower Bounds vs. Actual Blocking Probability in a 3-node Ring

Table 5.9: Calculated versus Simulated Bounds in a Six-Node Ring with relaxation of wavelength continuity

λ	0.8	1.0	1.2	1.4	1.6	1.8
P_{bl}	0.08535	0.1154	0.1425	0.1670	0.1889	0.2088
P_{bs}	0.08713	0.1210	0.1520	0.1814	0.2065	0.2283
P_{bu}	0.12747	0.1839	0.2353	0.2802	0.3187	0.3515

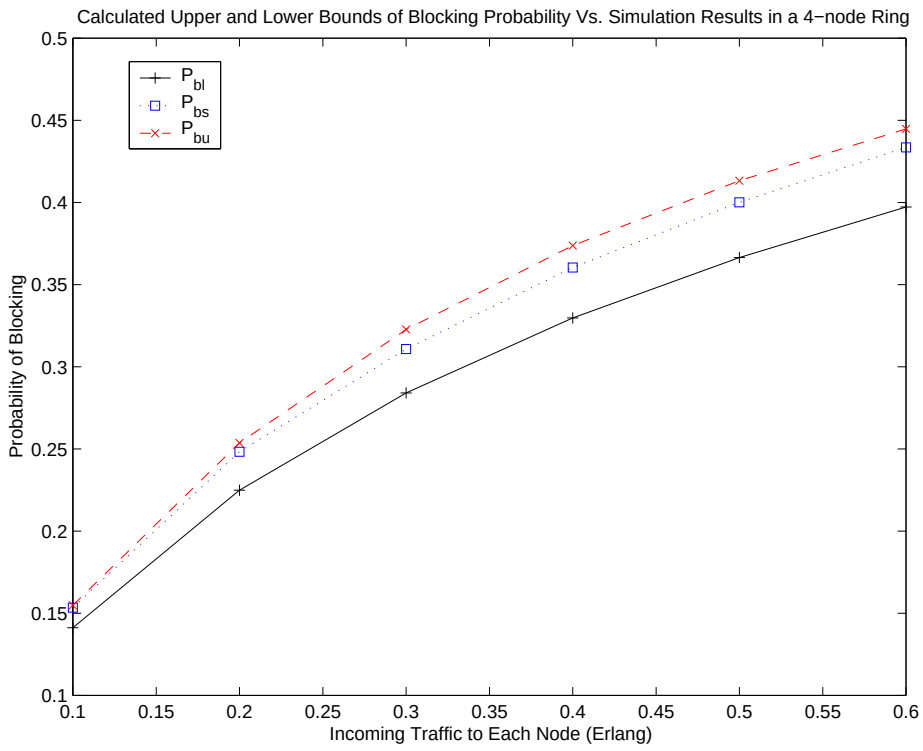

Figure 5.6: Calculated Upper and Lower Bounds of Blocking Probability Vs. Simulation Results in a 4-node Ring

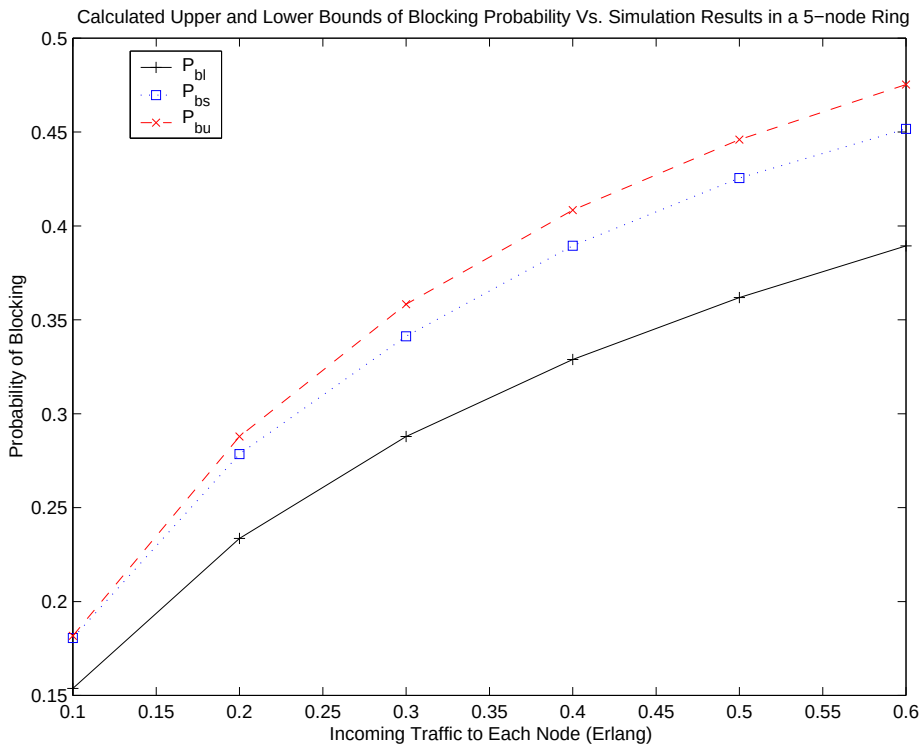

Figure 5.7: Calculated Upper and Lower Bounds of Blocking Probability Vs. Simulation Results in a 5-node Ring

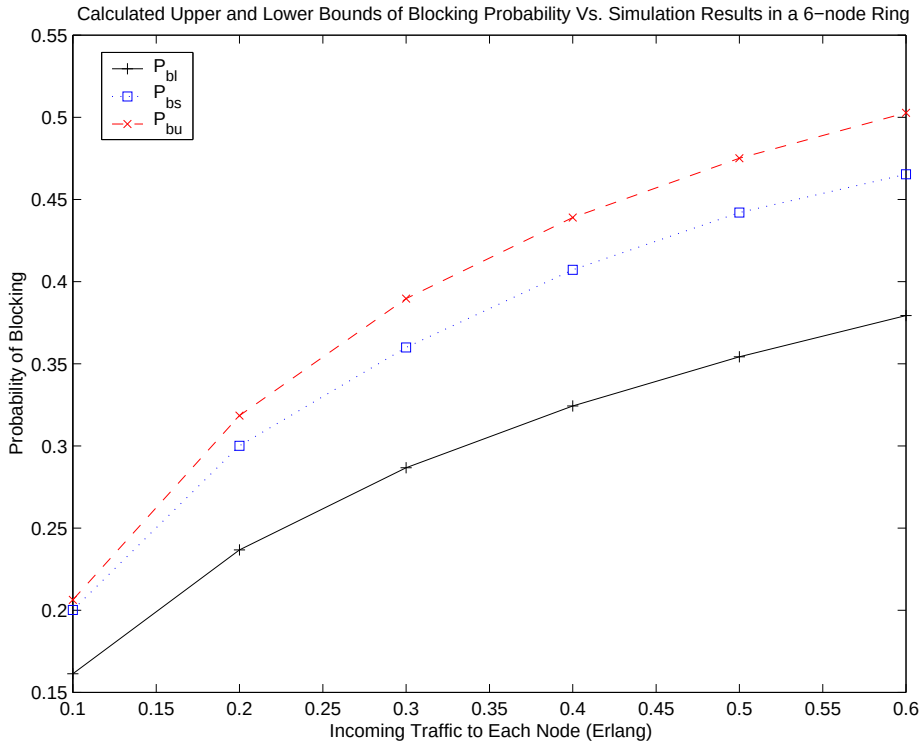

Figure 5.8: Calculated Upper and Lower Bounds of Blocking Probability Vs. Simulation Results in a 6-node Ring

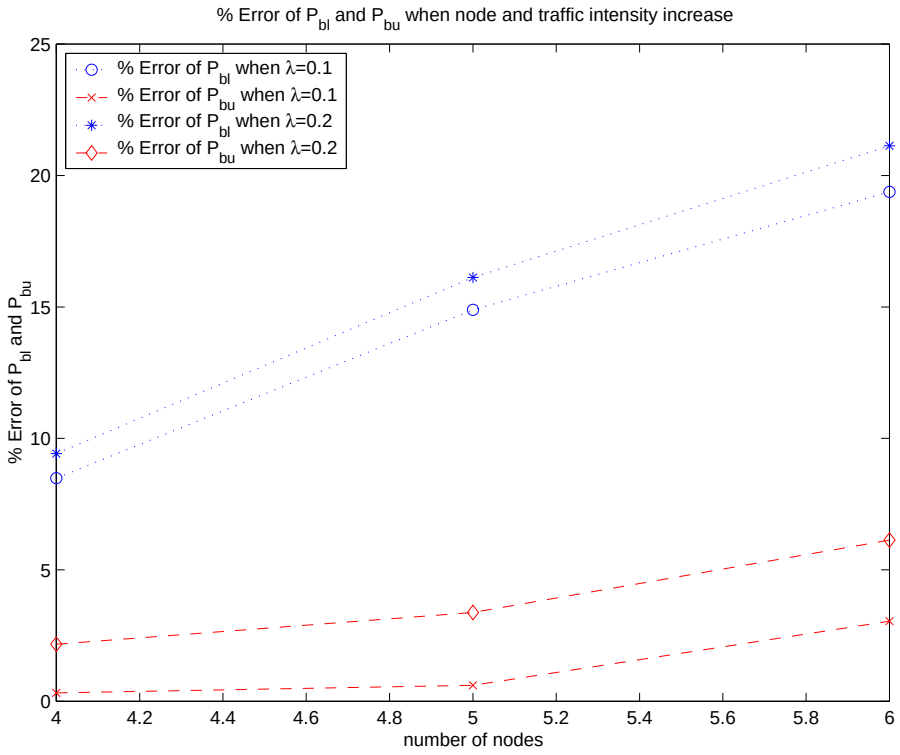

Figure 5.9: Error of P_{bl} and P_{bu} When Node and Traffic Intensity Increase

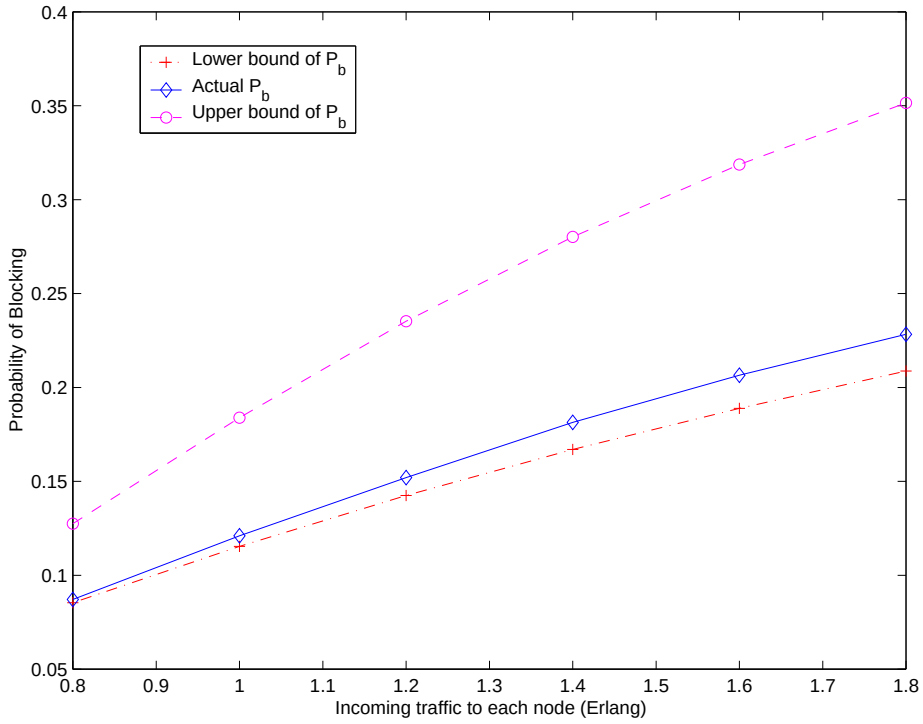

Figure 5.10: Calculated Upper and Lower Bounds of Blocking Probability Vs. Simulation Results in a 6-node Ring with 4 Wavelengths in each Fiber

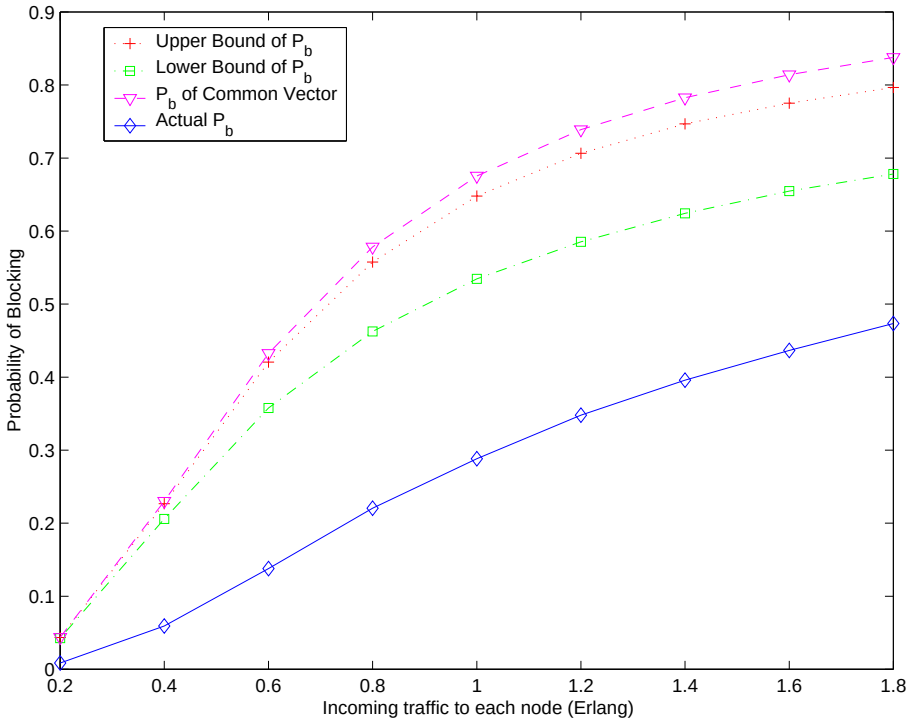

Figure 5.11: Comparison of Estimated and Actual Blocking Probability

We assume that there are 4 wavelengths in a fiber. Table 5.9 provides the comparison of calculated lower bounds P_{bl}, upper bounds P_{bu}, and the blocking probability P_{bs} from simulation when λ is increased from 0.8 to 1.8 at a step of 0.2. Table 5.9 is also depicted in Fig. 5.10.

If we use equation (5.11) in replacement of (3.3) to compute the traffic flow on a link, we can have a better estimation on the blocking probability of call setup requests.

5.7 Conclusion

This chapter presents a novel and efficient approach to calculate the lower and upper bounds of interconnecting link blocking probability in optically switched ring networks deploying optical network elements like ROADMs. The approach considers the impact of blocking on flow rate on a link by dividing the network state space into three subspaces. This novel approach has the potential to be applied to heterogeneous traffic demand matrix in ring or mesh networks. The efficiency of the approach provides the network designers a useful tool to control the QoS or adjust the traffic flow in their networks. It could also facilitates the estimation of the network performance in various scenarios, such as the common vector solution for generalized label continuity constraints.

Chapter 6: Conclusions and Future Work

6.1 Further Improvement on The Path Computation Efficiency in Multi-Layer Networks

6.1.1 A Mixture of Network Graph and Channel Graph

Theorem 8 has proved that the channel graph has the same order of complexity as the network graph for a sparse network with k hybrid node such that $k \ll |V|$. Equation (4.16) gives the approximate computational complexity. In a dense network with all other assumptions the same as in theorem 8, equation (4.16) can be written as:

$$O(T) = d_G^-(v_i)|E| + d_G^-(v_i)|V|(\log|V| + \log(d_G^-(v_i))). \tag{6.1}$$

In a dense network such that $d_G^-(v_i) = O(|V|)$, we have:

$$O(T) = |V||E| + 2|V|^2(\log|V|). \tag{6.2}$$

If the network G has limited number of boundary node, we can probably further simplify the path searching by a mixture of network graph and channel graph.

Figure 6.1 is a simple example of a network graph with clear region boundary. We call the nodes that are not connected to any region boundary node as internal node. We can convert any region boundary node v_i with all the links $\langle e_j, s_k\rangle$ such that $\langle e_j, s_k\rangle \in \varepsilon_G^-(v_i, s_k) \bigcup \varepsilon_G^+(v_i, s_k)$ into a channel graph H. All the other single-switching-type-capable nodes v_x are moved to H. For v_x which is connected through

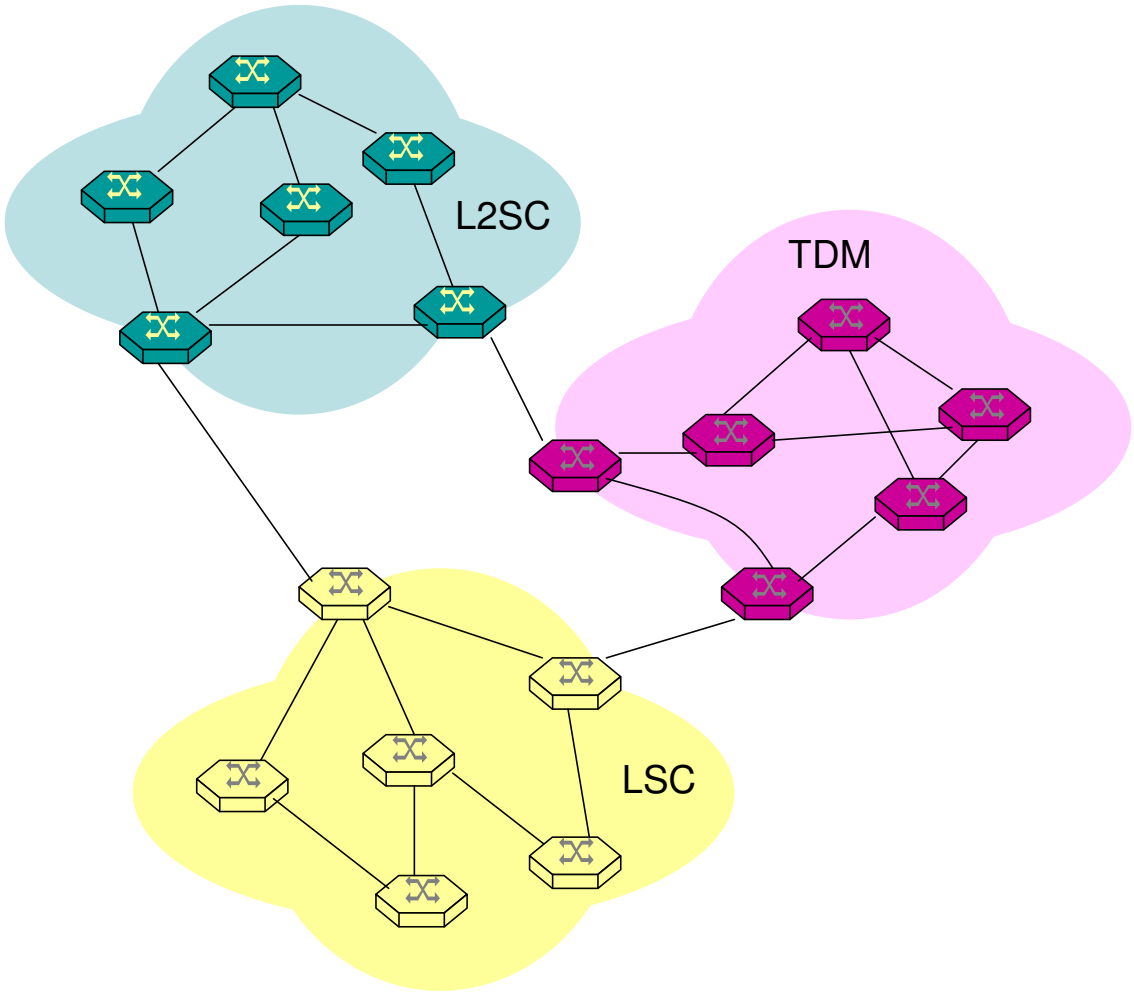

Figure 6.1: A Network with Limited Region Boundary Nodes

e_j to region boundary node v_i in G, because the link e_j has been converted into a node n_y in the channel graph, we create a virtual link between v_x and n_y in H. For links e_j that is not connected between an internal node, it is moved to H without any change.

This mixed graph solution with k region boundary nodes such that $k \ll |V|$ can reduce the computational complexity of the path computation to be approximately that in a network graph. The worst case is that every node is a region boundary node, which has the same computational complexity as the channel graph solution.

6.1.2 Further Improvement on Efficiency by Considering Switching Network Concept

Theorem 7 has proved that the channel graph is the most efficient solution for the worst case that the cost of taking $\langle e_x, s_j \rangle \in \Gamma^+_G(v_i)$ is dependant on $\langle e_y, s_k \rangle \in \Gamma^-_G(v_i)$ for each combination of v_i, e_x, s_j, e_y and s_k.

Figure 4.1 shows the architecture of a hybrid node with multiple switching elements. The adaptation cost between different switching capabilities can actually be modeled as the link cost between TDM switching element and PSC switching element. This is similar to the link between different wavelength planes in the wavelength graph. Though it does not capture the interface specific constraints as given in figure 4.2, it gives us further possibility to aggregate $\langle v_i, e_x, s_j, e_y, s_k \rangle$ according to switching elements and further reduce the computational complexity of path computation algorithm.

6.2 Future Work on Common Vector Solution

6.2.1 Proof or Disproof of Theorem 1 for any ρ

Theorem 1 is only proved for small ρ. According to simulation, this inequality should hold for any ρ. Rigorous proof or disproof of theorem 1 for any ρ is desired.

6.2.2 Common Vector Performance in case of Multiple Fibers between Two Nodes

When there are multiple fibers between two nodes, the evaluation of common vector performances is different from what is shown in chapter 3. I leave this part for future study.

6.3 More Accurate Estimation on Link Performance Bounds

The link performance bounds in chapter 5 is only on ring topology. For mesh networks, a tight estimation on link performance bounds can provide the network operator more confidence on network planning and path computation strategy.

6.4 Application of Common Vector and Channel Graph Solution in Multi-domain Networks

The introduction of path computation elements in [6] and [44] enables cooperation between PCE components in multi-domain environment. Reference [45] discussed cooperative inter-domain path computation, on which the performance of common vector and channel graph solution can be studied.

6.5 Stability of Virtual Topology

In multi-layer networks, the lower layer provides virtual topology to the neighboring upper layer. When searching a path, we should try best to use the residual capacity on existing virtual topology, i.e., try to avoid switching type adaptation. When a cross-layered path is setup, the virtual topology is changed from the perspective of the upper layer, which may make cause subsequent establishment and tearing down of existing paths if we want to guarantee the optimality of these paths.

6.6 Conclusion

This dissertation discusses various constraints on path computation in multi-layer networks in support of routing and traffic engineering. Solutions to different constraints are provided, feasibility of the common vector solution for generalized label continuity constraints and optimality of the channel graph solution to interface specific constraints are provided. Though we focuss here on multi-layer networks, the approach on multi-layer path computation described in this paper is also applicable to multi-area/multi-AS networks.

All the constraints, such as optical impairment, wavelength continuity and switching capability requirements, can be satisfied. Channel Graph may be considered as a general approach to address the non-additive interface specific adaptation constraints at a node. This method is in particular efficient for switching type adaptation for which the number of switching types is limited. However, for label continuity the common vector method is preferred.

The solutions to path computation as discussed here lend themselves as good

candidates for practical implementation. The proposed solutions for switching type adaptation and VLAN tag have been implemented as part of path computation in Dynamic Resource Allocation in GMPLS Optical Networks (*DRAGON*) project, an NSF sponsored project to create dynamic, deterministic, and manageable end-to-end network transport services for high-end e-Science applications.

Bibliography

Bibliography

[1] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol label switching architecture," Jan. 2001, RFC3031.

[2] E. Mannie and Ed., "Generalized multi-protocol label switching (GMPLS) architecture," *RFC3945*.

[3] D. Awduche and B. Jabbari, "Internet traffic engineering using multi-protocol label switching (MPLS)," *Journal of Computer Networks*, vol. 40, no. 1, pp. 111–129, Sept. 2002, invited Paper.

[4] M. Vigoureux, B. Berde, L. Anderson, T. Cinkler, L. Levrau, M. Ondata, D. Colle, J. Fdez-Palacios, and M. Jäger, "Multilayer traffic engineering for GMPLS-enabled networks," *IEEE Communications Magazine*, pp. 44–50, July 2005.

[5] K. Kompella and Y. Rekhter, "OSPF extensions in support of Generalized Multi-Protocol Label Switching (GMPLS)," Oct. 2005, RFC4203.

[6] A. Farrel, J.-P. Vasseur, and J. Ash, "A Path Computation Element (PCE)-based architecture," Aug. 2006, RFC4655.

[7] L. Berger, "Generalized Multi-Protocol Label Switching (GMPLS) signaling resource reservation protocol-traffic engineering (rsvp-te) extensions," Jan. 2003, RFC3473.

[8] K. Kompella and Y. Rekhter, "Link bundling in MPLS traffic engineering (TE)," Oct. 2005, RFC4201.

[9] A. Farrel, D. Papadimitriou, J.-P. Vasseur, and A. Ayyangar, "Encoding of attributes for multiprotocol label switching (MPLS) label switched path (LSP) establishment using resource reservation protocol-traffic engineering (RSVP-TE)," Feb. 2006, RFC4420.

[10] T. Lehman, J. Sobieski, and B. Jabbari, "DRAGON: a framework for service provisioning in heterogeneous grid networks," *Communications Magazine, IEEE*, vol. 44, no. 3, pp. 84– 90, Mar. 2006.

[11] M. Veeraraghavan, X. Fang, and X. Zheng, "On the suitability of applications for gmpls networks," *IEEE GLOBECOM, San Francisco*, Nov. 2006.

[12] R. Bhandari, *Survivable Networks: Algorithms for Diverse Routing*, 1st ed. Kluwer Academic Publishers, 1999.

[13] W. L.Winston, *Operations Research: Applications and Algorithms*, 4th ed. THOMPSON BROOKS/COLE, 2004.

[14] D. Banerjee and B. Mukherjee, "Wavelength-routed optical networks: Linear formulation, resource budgeting tradeoffs, and a reconfiguration study," *IEEE/ACM Transaction on Networking*, vol. 8, no. 5, pp. 598–607, Oct. 2000.

[15] M. Gondran and M. Minoux, *Graphs and Algorithms*, ser. Discrete Mathematics. A Wiley-Interscience Publication, June 1986.

[16] B. Mukherjee and D. Banerjee, "Some principles for designing a wide-area WDM optical network," *IEEE/ACM Transaction on Networking*, vol. 4, no. 5, pp. 684–695, Oct. 1996.

[17] K. Shiomoto, D. Papadimitriou, J.-L. L. Roux, M. Vigoureux, and D. Brungard, "Requirements for GMPLS-based multi-region and multi-layer networks (MRN/MLN)," *draft-ietf-ccamp-gmpls-mln-reqs-05.txt*, Aug. 2007, Internet Draft (work in progress).

[18] E. Dotaro, M. Vigoureux, and D. Papadimitriou, "Multi-Region networks: Generalized multi-protocol label switching (GMPLS) as enabler for vertical integration," *Globecom, IEEE*, pp. 374– 379, Nov. 2004.

[19] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, "Requirements for traffic engineering over MPLS," Sept. 1999, RFC2702.

[20] B. Jabbari, S. Gong, and E. Oki, "On constraints for path computation in multilayer switched networks," *IEICE Transactions on Communications*, vol. E90-B, no. 8, pp. 1922–1927, Aug. 2007.

[21] E. Oki, N. Matsuura, K. Shiomoto, and N. Yamanaka, "A disjoint path selection scheme with shared risk link group constraints in gmpls networks," *IEICE Transactions on Communications*, vol. E86-B, no. 8, pp. 2455–2462, Aug. 2003.

[22] R. Elsenpeter and T. J. Velte, *Optical Networking: A Beginner's Guide*, 1st ed. McGraw-Hill/OSBORNE, 2002.

[23] P. Tomsu and C. Schmutzer, *Next Generation Optical Networks: The Convergence of IP Intelligence and Optical Technologies*, 1st ed. Prentice Hall, 2002.

[24] D. Eppstein, "Finding the K shortest paths," *SIAM Journal on Computing*, vol. 28, no. 2, pp. 652–677, Apr. 1999.

[25] V. M. J. Pelayo and A. M. Varó, "Computing the k shortest paths: A new algorithm and an experimental comparison," in *Proc. 3rd Int. Worksh. Algorithm Engineering (WAE 1999)*, ser. Lecture Notes in Computer Science, J. S. Vitter and C. D. Zaroliagis, Eds., no. 1668. Springer-Verlag, 1999, pp. 15–29.

[26] E. Q. V. Martins and M. M. B. Pascoal, "A new implementation of YEN's ranking loopless paths algorithm." *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, vol. 1, pp. 121–134, Jan. 2003.

[27] I. Chlamtac, A. Faragoe, and T. Zhang, "Lightpath (Wavelength) routing in large WDM networks," *IEEE JSAC*, vol. 14, no. 5, June 1996.

[28] H. Zhu, H. Zhang, K. Zhu, and B. Mukherjee, "A novel generic graph model for traffic grooming in heterogeneous WDM mesh networks," *IEEE/ACM Transactions on Networking*, vol. 11, no. 2, pp. 285–299, 2003.

[29] W. Yao and B. Ramamurthy, "A link bundled auxiliary graph model for constrained dynamic traffic grooming in WDM mesh networks," *IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS*, vol. 23, no. 8, pp. 1542–1555, 2005.

[30] S. Gong and B. Jabbari, "The scalability and performance of common vector solution to generalized label continuity constraint in hybrid optical/packet networks," *IEEE GLOBECOM, Washington DC*, Nov. 2007.

[31] B. Jabbari, "Extended indexing method for resource assignment," *Internal Report*, Aug. 2006.

[32] R. A. Barry and P. A. Humblet, "Models of blocking probability in all-optical networks with and without wavelength changers," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 5, pp. 858–867, June 1996.

[33] L. Wei Chen and E. Modiano, "Efficient routing and wavelength assignment for reconfigurable wdm ring networks with wavelength converters," *IEEE/ACM Transactions on Networking*, vol. 13, no. 1, pp. 173–186, Feb. 2005.

[34] S. Gong and B. Jabbari, "Link performance bounds in homogeneous optically switched ring networks," *IEEE GLOBECOM, San Francisco*, Nov. 2006.

[35] K. Kompella and Y. Rekhter, "Routing extensions in support of generalized multi-protocol label switching (GMPLS)," Oct. 2005, RFC4202.

[36] D. Papadimitriou, M. Vigoureux, K. Shiomoto, D. Brungard, and J. L. Roux, "Generalized multi-protocol label switching (GMPLS) protocol extensions for multi-layer and multi-region networks (MLN/MRN)," *draft-papadimitriou-ccamp-gmpls-mrn-extensions-04.txt*, July 2007, Internet Draft.

[37] A. Giorgetti, N. Andriolli, L. Valcarenghi, P. Castoldi, and F. Cugini, "Introducing node adaptation capability descriptor in GMPLS multi-region networks," *High Performance Switching and Routing Workshop*, pp. 29–33, May 2005.

[38] K. Kompella and Y. Rekhter, "Label Switched Paths (LSP) hierarchy with generalized multi-protocol label switching (GMPLS) traffic engineering (TE)," Oct. 2005, RFC4206.

[39] S. Thiagarajan and A. K. Somani, "Capacity fairness of wdm networks with grooming capabilities," *Proceedings of SPIE*, vol. 4233, pp. 191–201, Sept. 2000.

[40] "Network node interface for the synchronous digital hierarchy (SDH)," *ITU Recommendation G.707*, Oct. 2000.

[41] J. Kuri, M. Gagnaire, and N. Puech, "On the resource efficiency of virtual concatenation in SDH/SONET mesh transport networks bearing protected scheduled connections," *Journal of Lightwave Technology*, vol. 23, no. 10, pp. 3012–3023, Oct. 2005.

[42] S. Subramaniam, M. Azizogelu, and A. K. Somani, "All-optical networks with sparse wavelength conversion," *IEEE/ACM Transactions on Networking*, vol. 4, no. 4, pp. 544–557, Aug. 1996.

[43] Y. Luo and N. Ansari, "A computational model for estimating blocking probabilities of multifiber wdm optical networks," *IEEE Communications Letters*, vol. 8, no. 1, pp. 60–62, Jan. 2004.

[44] J. Ash and J. L. Roux, "Path Computation Element (PCE) communication protocol generic requirements," Sept. 2006, RFC4657.

[45] P. Torab, B. Jabbari, Q. Xu, S. Gong, X. Yang, T. Lehman, C. Tracy, and J. Sobieski, "On cooperative inter-domain path computation," *IEEE Symposium on Computers and Communications Proceedings*, pp. 511– 518, June 2006.

Curriculum Vitae

Shujia Gong received B.S. degree on Computer Engineering and Telecommunications from Beijing University of Posts and Telecommunications in 1993. He worked for Siemens Communication Network LTD. in Beijing from 1993 to 2000 as a technical consultant. He received his M.S. degree in Electrical Engineering from George Mason University in 2003. His research includes traffic engineering and path computation algorithm in hybrid optical/IP networks.