A NON-BLOCKING DESIGN PARADIGM FOR WDM MESH BACKBONE NETWORKS AND ITS PERFORMANCE ANALYSIS

by

Xiao MA

Master in Telecommunications, School of Computing and Information, University of Pittsburgh, 2012

Submitted to the Graduate Faculty of
the School of Computing and Information in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2018

UNIVERSITY OF PITTSBURGH

SCHOOL OF COMPUTING AND INFORMATION

This dissertation was presented

by

Xiao MA

It was defended on

July 21, 2017

and approved by

Richard Thompson, University of Pittsburgh, School of Computing and Information, Department of Informatics and Networked Systems

Jeffery Wheeler, University of Pittsburgh, Department of Mathematics

David Tipper, University of Pittsburgh, School of Computing and Information, Department of Informatics and Networked Systems

Balaji Palanisamy, University of Pittsburgh, School of Computing and Information, Department of Informatics and Networked Systems

Dissertation Director: Richard Thompson, University of Pittsburgh, School of Computing and Information, Department of Informatics and Networked Systems

A NON-BLOCKING DESIGN PARADIGM FOR WDM MESH BACKBONE NETWORKS AND ITS PERFORMANCE ANALYSIS

Xiao MA, PhD

University of Pittsburgh, 2018

Current network design problems can be solved by offline or online methods. Offline methods are criticized for their complexity and inflexibility, whereas online methods lack guaranteed optimality. Non-blocking properties, which are typically studied in switching structures, could be used to evaluate the capability of a switching structure to handle dynamic traffic. This dissertation extends the study of non-blocking networks to general-connected mesh WDM backbone networks. This study begins by finding that a set of special graphs have certain non-blocking properties, and that non-blocking routing algorithms can be designed to be implemented in a distributable manner without a central decision system. The in-depth philosophy of this research is to investigate the relationship among these non-blocking properties, the topography of the network, and the power of these distributed routing algorithms. This design paradigm is illustrated by applying it to a potential implementation for NSFNet. After confirming that NSFNet is **NOT** non-blocking, we propose a virtual topography that makes NSFNet virtually non-blocking, along with system diagrams for the node structures and the discussion of the implementation framework. To evaluate the performance of the non-blocking algorithms, we compare the performance of our proposed online algorithm with other algorithms in a general traffic scenario.

Keywords: Non-blocking mesh network, graph theory, routing algorithm, performance analysis, queue theory, simulation, cost efficiency.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

This dissertation is the result of interdisciplinary research in telecom and math, which is an ideal research topic I was hoping for since my undergraduate time. During this long journey I have my special appreciation for my academic advisor, Dr. Thompson, for his long time support and encouragement in academic studies. I admire his patience in modifying my drafts and manuscripts as well as his rigorous academic attitude. Without his support, this dissertation couldn't be made.

I have to say thanks to the committee members who provided me a vast amount of suggestions to improve this paper. Especially, I appreciate Dr. Wheeler, who insisted I focus on the writing style of my proof for a long time. During this time, from the tedious work, I spotted the logic flaws in my proof more than once, which elevated the quality of the math parts significantly.

Last, but not the least, I have to mention the support from my classmates and residence community. I am very grateful to Maggie Yu, from whom I benefited a lot in both study methods and life style. Special thanks go to Yun Wang from ECON at PITT, Yuxiang Wang from SHRS at PITT, and Zhipeng Yang from CMU for inspiring thoughts and encouragement.

1.0 INTRODUCTION

With the development of optical communication technology, optical networking has attracted many scholars for its significant potential in high speed transmission because of its low error rate and easy scalability. In *wavelength division multiplexed* (WDM) systems, data streams can be multiplexed into different wavelengths in the same fiber. Currently, WDM supports channel capacity for each wavelength at 2.5Gbps, 10Gbps, 40Gbps, and even 100Gbps. WDM channels are deployed mainly in the C-band (1530-1565nm) and L-band (1565-1625nm), in order to accommodate EDFA operating frequencies. The channel spacing for a 50Gbps channel is 0.4nm, and 0.8mm for 100Gbps. In the band between 1300nm and 1600nm, approximately, a total number of 375 100Gbps channels might potentially be deployed in a single fiber, the throughput of which sums up to be more than 30Tbps.

For its high data rate, WDM technology is expected to be the best candidate for next generation backbone networks. A node in the backbone network forwards the data streams, as well as processes access traffic. Local traffic formats, such as IP and SONET, are first handled by the node and transformed into WDM signaling, before entering the backbone network. In addition, the WDM channel capacity can be improved by setting more wavelengths or using different modulation techniques, without deploying more fibers. Only the updates of terminal devices are necessary. Thus WDM is welcomed by telecom carriers for its promising low cost and convenient capacity upgrade.

As a link layer transmission method, WDM has been widely put into practice in many networks. However, optical switching and routing becomes a shortcoming because a limited power to process optical bits. In addition, the limited performance of optical buffers and wavelength converters further hampers the flexibility of switching systems. Neverthe-

less, electrical switching is convicted of being complicated in hardware and costs significant amount of energy.

Given the above limitations, the networking design problem in the optical layer is even more difficult due to a large number of factors to consider. A networking design problems aims to carry out an optimal network, which usually comes with a routing table, that could handle the desired traffic (sometimes to reach a given QoS) at the lowest cost or bearing some best quality in some aspect.

However, an optimal solution can be achieved at the cost of significant computation complexity. What's more, the decision process to reach the optimal solution may depend on global knowledge of the traffic in the network. Given the nature of bursty traffic of the internet and dynamic traffic in the network, the process of either message collection and computation would hamper the promptness and correctness of the optimal solution, which in turn, will hamper the performance of the network, not to even mention the orchestration of the switching/routing/wavelength conversion functions of the optical devices to implement the solutions.

Instead of state-of-art optimization to route the traffic at a network, this paper presents a generalized traffic study in different networks for their non-blocking properties. Non-blocking properties depicts the ability of a switching structure/network to handle certain type of traffic at zero loss. With this concept, we seek once-for-all general results for different networks and traffic. We proposes a non-blocking paradigm consisted of non-blocking theorems, non-blocking routing methods, and its system infrastructures. Next, a simulation framework is developed to evaluate and the performance and efficiency of this non-blocking paradigm.

The contents of this paper is organized as following. Chapter 2 introduces the background for traffic analysis, WDM systems, and non-blocking networks. Chapter 3 introduces the graph theory knowledge that is related to a connected network and paths in a connected network between nodes. Chapter 4 formally defines the non-blocking property for a general mesh network and proposes several algorithms and proves that a set of general connected network is non-blocking with certain algorithms. Chapter 5 discusses the implementation framework of a non-blocking network proposed in Chapter 4. Chapter 6 propose a simulation framework that is used to evaluate the traffic potential of the non-blocking networks after

loosening a constraint. Chapter 7 concludes this paper.

1.1 MOTIVATION

In networking design, we need to determine a network that can handle the desired amount of traffic while keeping the cost low or maximizing some desired features. A general network has many parameters. For example, how it is connected, what is the number of links, or the capacity of each link, and the hardware devices at each node. An optimal solution for a network design problem is essentially the best compromise between the design objective and the cost. For example, in PSTN network designing, the design objective is the blocking rate and cost is related to the number of lines, where increasing the number of lines will reduce the blocking rate however increases cost, which becomes a state-of-art optimization over tradeoffs.

Benes suggests[53] that a satisfactory network design and traffic analysis study should have certain generality, which enables it to be applied to any system. It should put three factors into consideration: random traffic, control unit, and the connecting network. He also lists the comparison between different networks, control systems and routing algorithms as a necessary task.

However, due to the large scale of the variations of random traffic and connected networks, this study was long kept in the scope of mathematical modeling and approximations. With the increase of computation power, we are able to compute optimal solutions given a network and the traffic status by optimization methods. The optimal solution is essentially a routing table for the traffic in the network that could achieve some design objective.

Optimization methods are typically implemented as offline routing algorithms, because they need to know the network and the traffic (might include some traffic not yet arrived in some cases) in order to compute optimal routing paths for the traffic. Online algorithms, in contrast, compute the optimal decision without knowing global traffic status. Offline algorithms are criticized for their centralized infrastructure, in which message collection and synchronization is a pitfall in their implementations as the input of the computation might

be wrong or outdated due to delays and errors in message propagating process. In addition, it is also challenging to orchestrate the commands to reach the optimal result throughout the network.

The optimization methods appear to have sufficient generality since they model traffic and topography integrally. However its computation cost is great and its implementation concerns are likely to hamper the performance. Moreover, it provides solution in a case to case manner and sheds no light on the relationships for different networks and traffic status.

On the other hand, online algorithms are developed, which are essentially routing algorithms at an instant based on heuristic preferences to improve the performance. They are usually efficient and prompt compared to offline algorithms however their optimality is hard to guarantee nor evaluated with granularity. The details of the discussions are provided in Section 2.3.5.

In order to study the capability of different networks to handle dynamic traffic, non-blocking properties are worthy of investigation. It applies to every connected network, and a network with a non-blocking property demonstrates better traffic handling ability than a network without it. It is also a once-for-all result, such that, if we prove that a network is non-blocking for a set of traffic scenarios, we know there is a solution to route the connections for each traffic scenario. In another words, a non-blocking property would confirm that the network is able to handle a set of traffic scenarios.

In addition to the generality and "once-for-all" mighty power it introduces, another reason to study the non-blocking property for WDM mesh network is that it has been widely studied in traditional switching structures; for example, the Clos structure. Standing on these corner stones, we are very confident to find out promising results in mesh networks. Furthermore, in the study of non-blocking mesh networks, we may shed light on the in-depth relationship between non-blocking property and the network graph structure along with the power of routing algorithms.

Paper [2] provides a novel model method to approach non-blocking properties. However, this paper does not develop a result for non-blocking mesh networks based on the model. This dissertation continues this method, and proves a set of non-blocking general mesh network structures and proposes several non-blocking online algorithms. We also discuss the

Table 1: Interdisciplinary terms in this paper

Mathematics	Networking
Graph	Topography
Vertex	Node
Edge	Link

implementation framework to apply the non-blocking design paradigm to NSFNet. We also evaluate performance of the online algorithms and their mutations in different networks by simulation in more general traffic conditions.

This research journey was partially funded by NSF grant #CNS-1307643. I do appreciate NSF for funding this interesting project to shed light on unknown grounds.

1.2 CONTENTS

This paper covers an interdisciplinary research between graph theory, optimization methods, and computer networking. Different communities have different preference of terms. For example, mathematicians use "vertex" while the network community uses "node" to refer to the same thing. However they have different implications: in the math community a vertex is usually associated with degrees, weights and colors; in the network community, a node may represent a set of hardware devices and functions that are equipped in the node. This paper uses "vertex" in the math context and "node" in network context. Table 1 distinguishes a set of similar words.

It is worth mentioning that the term "link" has different expressions throughout this paper. When mentioned as a part of a telecom system, the term "link" is in the same concept as an undirected edge which supports two way communication and is denoted as (v_1, v_2).

When mentioned in a routing algorithm or its analysis, it represents a one way link which is a part of a directed routing path, and is denoted as $v_1 \to v_2$.

Mathematicians call a connected network "graph" while telecom engineers uses the word "topography" to refer the way a network gets connected. It is worth highlighting that, the word "topology" in is used to refer to the underlying simple graph of a network while it should be used to describe the methods to investigate the topography, not to even mention that it has a totally different meaning in mathematic conventions–the topology study in real analysis, etc. In this dissertation, we use the term "graph structure", and "topography" in the math parts (Section 2.4 and Chapter 4). In telecom sections, we keep the term "virtual topology"[1], which is currently well known to the telecom community as a synonym for the underlying simple graph of a network and its interconnectedness. To mention the structure of a graph in other contents of the telecom studies, we continue to use the term "topography".

This investigation of non-blocking mesh networks has taken a long journey. It begins with enumerating small scale rearrangeably non-blocking (later referred as RNB) graphs, which are plotted in a semi-lattice (Section 3.3). By observing the structural similarities through the result, several preliminary results are found. After I compared this RNB property to other graph theory properties, I speculated that bipartite complete graphs may have structural advantages and might be non-blocking. After a through investigation by combinatoric and graph theory analysis, I confirmed that they have certain non-blocking properties and I found there exists two online algorithms that could achieve the non-blocking properties due to the structure advantages. The routing algorithm along with the non-blocking graph structure forms a design paradigm. Several other facts regarding to the optimality of this paradigm are discussed.

Next, I moved to implement a bipartite complete graph topography in NSFNet by adding virtual links. In the curiosity to know the extreme power of our online routing algorithms in the virtual topology, I loosened the assumption of traffic patterns (non-blocking is no longer guaranteed) and conducted a series of simulations evaluating utility rates, blocking rates, and cost efficiency of virtual bipartite complete graphs.

[1]The "virtual topology" is used to describe the structure of the network in the virtual layer. It is indeed an improper use of word "topology". However, due to its popularity and the matureness of the concept I reluctantly use this term in related context.

2.0 BACKGROUND

This chapter introduces the background for the studyings of the non-blocking properties of a network. As classic non-blocking studies are based on switching structures, we show that optical non-blocking networks could be enabled in its hardware structure by reviewing optical switching technologies. Next, we introduce the background of WDM networks including network infrastructure, optical switching techniques, and network design methods. We highlight the drawbacks and limitations for current network designing methods and propose that a non-blocking network paradigm may be very promising to tackle these drawbacks.

Section 2.2 introduces circuit switching, OBS, and OPS based on the above framework. Section 2.3 discusses the network design problems in optical networks. Section 2.4 discusses the non-blocking studies and proposes the main topic for this dissertation. Section 2.5 summarizes this chapter.

2.1 WDM INFRASTRUCTURE

In WDM networks, optical signals can be multiplexed into different wavelengths in the same fiber. This is very similar to FDM signals in electric systems. However, they are vastly different because of the difference in hardware functionalities.

Optical WDM networks have several specific devices. The reshaping devices keep the signal quality from chromatic dispersion and non-linear effects. These impacts increase with the fiber's length and thus reshaping devices need to be positioned over a distance to guarantee signal quality. Wavelength converters are another type of important hardware devices, which are capable of moving the signal from one wavelength to another, similar to a remod-

ulation function, which can improve the flexibility of the network. Nevertheless, wavelength converters are criticized of being costly and have performance problems. Another device is optical buffers. which are awkward compared to electrical buffers because of poor functionality and scarcity. Due to these differences in hardware devices, the infrastructure and network design problems of optical networks are different from electrical networks.

The infrastructure of a node in a WDM backbone network is briefly illustrated in Figure 1. It provides access for local network traffic to the backbone network as well as switching/routing traffic. In this example node, a switching structure is connected to three identical fibers which are connected to other nodes. The optical line terminal (OLT) is a set of devices including transponders, multiplexers, demultiplexers, and protocol processing devices. The OLT is used to handle incoming signals including IP, SONET, etc. It then demultiplexes the signal into different wavelengths (3 for each fiber in this example) and passes them to separate ports of the optical cross connect (OXC).

An OXC performs optical switching functions. The flexibility of the OXC depends on whether there are wavelength converters equipped at the ports of the OXC. If this OXC is wavelength converter free, the port handling a wavelength can only be switched to the ports assigned the same wavelength. Otherwise, the OXC can connect a pair of ports assigned different wavelengths as long as the wavelength converter equipped in the OXC is capable of interchanging the wavelengths.

The traffic initiated or destined to the local network enters the node through via optical add & drop multiplexers (OADM), which are connected to the OXC by dedicated ports shown in Figure 1. The OXC then forwards it to the network before reaching its destination.

It should be mentioned that a wavelength converter is a double-edged sword. It provides flexibility to the network which are expected to reduce the blocking rate and save wavelengths. However, it has two critical drawbacks. First, the cost is high and is not likely to drop within a few years. Second, when the number of wavelengths are huge, the performance of wavelength converters does not satisfy commercial uses.

It is a tricky strategy to deploy wavelength converters at a portion of nodes in the network, or similarly, a portion of the ports at the OXC to compromise the tradeoff between cost the flexibility. This problem is not the main topic of this dissertation and thus omitted.

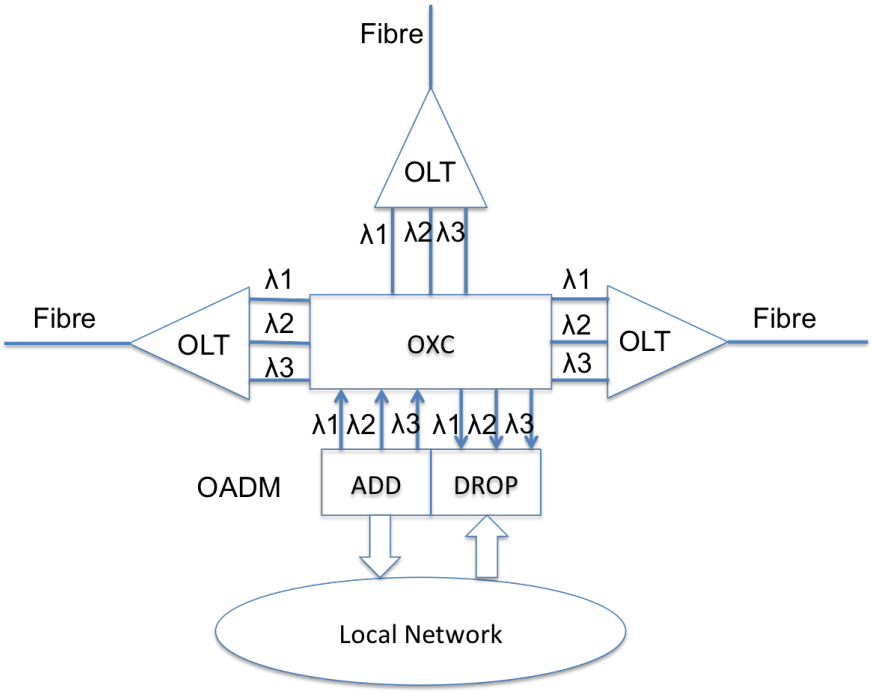

Figure 1: WDM backbone core node infrastructure

2.2 OPTICAL SWITCHING TECHNIQUES IN WDM NETWORKS

In optical networks, we denote a connection as a source/destination pair that represents a traffic demand[1] in which the source vertex needs to send data to the destination vertex. The connection may be routed to a multiple hop path while the vertices in the path should forward the data of the connection to the right outgoing port. This technique is called optical switching and is the enabling technology for optical networks. This section covers three branches of optical switching–circuit switching, optical packet switching (OPS), and optical burst switching (OBS).

2.2.1 Optical Circuit Switching

In optical circuit switching, an optical path along with its wavelength (s) are reserved for a connection, until the connection ends or is shifted. A connection request packet is first sent from the source to the destination along a path. Upon receiving the request packet, the destination will return an acknowledgment packet if the connection could be set up. The wavelengths requested by the connection along the path are then reserved in this process. After the reservation, the data stream is forwarded to the destination along the path in the reserved wavelengths. Circuit switching has stable network delay and guarantees QoS. When multiple connections compete for the links and wavelengths, we need to solve the state-of-art problem of routing and wavelength assignment (RWA) to optimize the routing paths to reach the best service through the network. The optimal solution of RWA problem for the connection requests can be modeled as an integer linear programming problem, which is introduced in detail in Section 2.3.

[1] Terms "connection" and "session"...etc are used to refer the same concept in literatures.

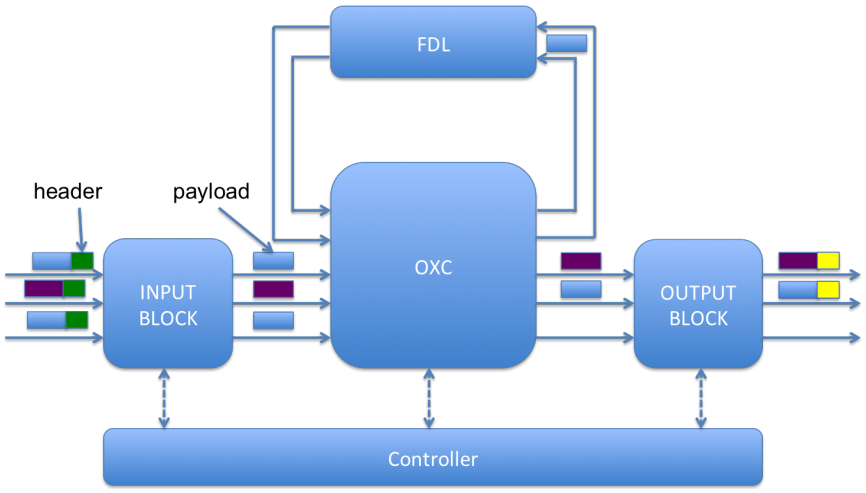

Figure 2: Optical packet switching facility infrastructure

2.2.2 Optical Packet Switching

Unlike circuit switching, optical packet switching processes each packet individually. Each packet has a header which contains its source and destination. The switching process is performed based on the header information for every packet. Most variations of OPS use a synchronized scheme to switch the packets (also known as slotted switching). The length of the slot is predetermined and the packet is fit into the slots for synchronized processing. It is illustrated by the block diagram in Figure 2.

The header is first processed by the input block, and handled over to the controller. The controller reads the content in the header and determines the outgoing port of the packet, and sends this command to the OXC, before its payload reaches it. Once the payload reaches the OXC, the OXC will switch the payload to its destination port according to the information in the header. The output block wraps the payload with a new header and then sends the "new" packet to the network. The header processing can either be accomplished electrically or optically. Note that this diagram focuses on the switching facilities and the multiplexer/demultiplexers and OADMs in the node are omitted.

Note that the arriving packet may not be perfectly synchronized due to inconsistent

network delay even the output block is sending out synchronized packets. Thus both the input and output ports have to synchronize the packet to the slots. The scheme reduces the chance of port contention by regulating the packets into slots.

However port contention may still happen when multiple packets are heading to the same port at the same instant. At this instance only one of them can be forwarded to the port while the remaining packets are redirected to optical buffers. They will be dropped if there are no optical buffers available. Optical buffers are made with fiber delay lines (FDL). When the output port is idle, a contention packet returns to the OXC via the input port from the FDL and gets forwarded. The delay time may be set to integer multiples of the slot length such that a contention packet could stay in the system for more time slots and has more chances to be forwarded. Currently, FDLs are scarce and would result in a QoS degrade if packets are dropped when the FDLs are full.

We can alternatively solve port contention by using wavelength converters to redirect the packet to another wavelength for a second chance to be switched. It utilizes the resource in another wavelength in the same fiber to route the packet. However, this approach would increase the cost due to wavelength converters being expensive.

OPS offers fine granularity to the sub-wavelength lever because packets in the same wavelength can be switched to different ports. However its QoS is dependent on optical buffers and wavelength converters. On the other hand, circuit switching does not provide finer granularity than the wavelength level because all traffic in a wavelength in a connection must be switched to the same destination but enjoys a lower hardware cost. Optical Burst Switching is a compromise on both approaches, aiming to balance among the granularity in OPS and the good QoS in circuit switching.

2.2.3 Optical Burst Switching

OBS inherits both the features from circuit switching and OPS. It processes switching at the packet level while it reserves paths like circuit switching. Thus it offers fine granularity and keeps a decent QoS. In OBS, a one-way control message is sent along to reserve the path.

During this process traffic is aggregated at the source vertex, which will later be routed to the reserved path. Aggregated packets will be sent continuously to the network which is described as a "burst".

The control message of OBS is isolated from the packets in the burst. In general, when the first packet reaches the node, a control message about an upcoming burst is sent out to the network. The nodes along the path will reserve bandwidth for the burst upon seeing the control message. The aggregated packets are sent sequentially to the network and forms a burst once the traffic aggregation process is finished. Consequently there is a delay for all the packets in the burst. This is similar to an analogy that a bus only departs when all seats are occupied while most passengers must wait for some time until the bus moves out.

The OBS path reservation process is different from that in circuit switching. Circuit switching utilizes a three-way handshake process. This process has a response time of at least 1 round trip time (RTT). The sender must wait for 1 RTT to receive the acknowledgement before the first packet can be sent. If the burst aggregation time is less than 1 RTT, the packets need to wait for the acknowledgement even after the aggregation process is finished, which is an undesirable overhead cost in both data rate and delay. It would be even worse if the aggregation process finishes because the transmission buffer is full in this period, which may cause packet drops. To reduce the overhead costs, along with potential packet drops, a simpler protocol is preferred such that the control message should directly reserve the resources. It is worth mentioning that the buffers for the bursts might be implemented with electrical buffers and thus don't suffer from the drawback of optical buffers.

There are a variety of standards for the length of burst and the delay between the control signal and the burst for different scenarios because OBS is a tradeoff between performance and delay. A large aggregation time would increase the overall delay but guarantee low drop rate as a long burst can be formed. The overhead cost can also be reduced; A short aggregation time would guarantee the delay below a certain threshold. Nevertheless the control overhead and drop rates might be large compared to long bursts due to limited burst length.

2.2.4 Summary

This section gives a brief idea about the three optical switching techniques and their features. Circuit switching guarantees QoS but does not guarantee fine granularity. OPS regulates the packets into constant length slots and thus offers fine granularity. OBS is a compromise between both circuit switching and OPS.

All the three switching method are different in time scale. An OPS switching decision is done once for a packet. The OBS switching decision lasts for a burst, which is essentially a number of packets. The circuit switching decision could lasts for arbitrary time. Despite this great difference, they share the same feature in nature that at any instant, the source node, in one wavelength, can only send to one destination node, which is essentially the same premise for Clos structures. This fact provide us the corner stone to study non-blocking property in optical switching networks with regarding to permutation traffic sets (see Appendix A.2).

2.3 WDM NETWORK DESIGN

This section discusses the topic of optical networking design problems. A typical network design problem is to find optimal routing paths for certain traffic in a network such that the network get best utilized for a certain objective. In another words, it aims to find the best routing plans given the network, traffic, and optimization objective.

Especially in optical networking, the routing process not only computes the paths, but also assigns the wavelengths to them. The assignment of wavelength is affected by the presence and capability of wavelength converters. This problem is also referred as routing and wavelength assignment (RWA) problem. Henceforth, this dissertation uses the term "RWA" to refer to the WDM networking design problem.

As can be implied, the RWA problem can be modeled as an optimization problem. It produces the optimal routing paths as the output based on the traffic scenario and the structure of the connected graph. This method is capable to provide a solution for a variety of

networks and traffic patterns in a case by case manner.

It is critical to distributed network designers to know whether the traffic scenarios are known or not. The optimization method requires that the traffic in the entire network be known before it can produce the optimal results. Thus, it is categorized as an off-line algorithm, which needs to collect information from every node in the network. In contrast, on-line algorithms operate on the basis that they do not need all the information to make the optimal result. The comparison of off-line and on-line algorithms is discussed in subsequent chapters.

There are three major design topics in which the optimization approach for the method is widely used. We summarize them as: traditional telecom, IP-over-WDM, and elastic networks.

The telecom branch focuses merely on the optimization of the use of telecom links. Usually it takes advantage of the optimized routing solution and sets the number of links or wavelengths accordingly to reach the best balance between QoS and cost; while the specification of QoS and cost could be flexible to meet different preferences.

The IP-over-WDM framework is a two layer networking infrastructure: the optical layer is responsible for transmission and switching; the IP layer is responsible for routing. Virtual circuits may also be set up in this framework which enable a virtual layer between the IP layer and the physical layer. Thus, the networking design problem becomes finding the optimal decision of routing path, assignment, either in physical layer or virtual layer.

The concept of elastic networks is proposed to provide more throughput by manipulating the frequency assignment and modulation schemes together. For example, by using large spectrum grids, we use fewer guard bands, which means more spectrum is utilized. By using high-order modulation, more bitrate can be achieved from the same bandwidth of spectrum. Thus, state of the art networking design in this field involves the manipulation of modulation schemes and spectrum assignments.

2.3.1 General RWA Models

In this section we introduce and review the RWA methods in the topics mentioned above. We begin with the most general optimization model, which takes the traffic scenario and the structure of the network as input, and produces the routing path of the traffic in the network as the output. The routing function of the traffic is conducted by an optimization approach in the form of an ILP/MILP problem. In addition, we mention heuristic algorithms based on this model which provide suboptimal routing paths with reduced complexity.

An RWA problem can be formulated as a multi-commodity flow optimization problem. The inputs to this problem are the structure of the network, a traffic matrix, and an optimization objective. The structure of the network is usually modeled as a simple connected graph $G = \{V, E\}$. The traffic matrix can be written as an n-by-n matrix A where $n = |V|$. Element A_{sd} represents the amount of traffic from vertex s to vertex d.

The optimal routing path is modeled as a set of indicator variables F_{ij}^{sd}. The variable F_{ij}^{sd} represents the amount of traffic from node s to node d that is routed in directed edge (i, j). If (i, j) is not part of the routing path, $F_{ij}^{sd} = 0$. There is an F variable for each traffic and each link. The combination of F variables represents the possible routing paths, which are subtly related to the underlying graph and its traffic. As will be seen later, the F_{ij}^{sd} values are regulated by the Kirchoff's Law in order to validly model the graph and the traffic.

We offer an example to introduce the core of a general RWA problem. We omit the objective function in this example since it is not critical for this part, while its consideration may add several constraints which may be distracting. The collection of objective functions will be introduced and compared in detail in the next sections.

An example environment:

A given objective function (usually associated with F_{jk}^{sd} and other factors).

has the constraints:

$$\forall j, \sum_i F_{ij}^{sd} - \sum_k F_{jk}^{sd} = \begin{cases} A_{sd} & \text{if } s = j \\ -A_{sd} & \text{if } d = j \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

$$
\begin{aligned}
A_{sd} &\in \{0,1\} \\
F_{ij}^{sd} &=\in \{0,1\}
\end{aligned}
\tag{2.2}
$$

The set of constraints in equation 2.1 represents Kirchoff's Law, which regulates that the amount of incoming traffic be equal to that of outgoing traffic at each node. These constraints integrate the structure of the network and its traffic. Further, they define the valid combination of the values of the F variables in the optimal solution. Equation 2.2 sets the variables as binary, which describes a typical case where a traffic stream needs to reserve the bandwidth equal to the link capacity in the network along its routing path, and all streams are identical.

The computational complexity for this model is significant. If the network is modeled by a graph $G = \{V, E\}$, the number of F variables is equal to $\rho\frac{|V|(|V|-1)}{2}|E|$, where ρ represents the fraction of active source-destination pairs. Thus, the number of combinations of F variables is $2^{\rho\frac{|V|(|V|-1)}{2}|E|}$, which is not polynomial and has gone beyond bound for practical computing, even if we only have the most basic Kirchoff's law constraints in the problem. If more case-specific constraints are added to the problem, the computational complexity will inevitably be greater.

The integer constraints for F and A variables can be loosened to reduce the computational complexity as real-valued optimizations are more easily compared to that of integer-valued. In this case, the solution may be multiple routing paths, each of which carry a portion of the total traffic to the destination. The paths are different but not necessarily node or edge disjoint. They allow more flexible routing as one traffic stream can be divided into several parts and delivered to the destination via different paths. We can use randomized rounding to obtain a suboptimal integer result from the solution with loosened constraints, however care has to be taken as most real networks do not support fine granularity in traffic processing and the extent of the sub-optimality of the solution might be opaque.

Despite omitted in the example, it is worth mentioning the following general aspects of an objective function. First, the objective function is determined by the design criteria, which evaluates the optimality of the network. Second, it usually introduces more constraints to the problem. There are a great variety of constraints that was brought about by objective

functions, for different goals and scenarios of network design. These variations will be summarized in the following sections with detailed explanations.

As mentioned, this ILP approach is an off-line algorithm and needs to be implemented to collect traffic information from all nodes in the network. On the other hand, there are also online algorithms (also known as heuristic algorithms) proposed to find the routing paths. On-line algorithms do not need to collect all information around the network. Among them, the K-Shortest Path Algorithm is most widely used.

The K-Shortest Path Algorithm returns a result of at most K different shortest paths in a graph for an s/d pair. It only needs to know the remaining bandwidth/wavelengths in the network without knowing the traffic in the paths. Based on this algorithm, a heuristic algorithm can be implemented as a "pick from pool" process from the K shortest path returned by the algorithm, in which greedy algorithms are usually implemented to achieve a good balance between optimality and efficiency. The "pick from pool" preference can be flexible to model different optimization objectives.

This section reviews the ILP method as the typical off-line algorithm to the network design problem and the K-shortest path algorithm as the counterpart in on-line algorithms. In next subsections, we introduce the different networking design topics mentioned above: Telecom, Elastic Optical Network (EON), and IP-over-WDM. The telecom topic mainly covers the classic models toward a telecom network, and its CAPEX, and OPEX optimizations, which are popular in the 90's research papers. The EON topic introduces the approaches for networks which feature dynamic spectrum allocation. The IP over WDM section concentrates on the network design variants in its special framework and has a large number of variations.

2.3.2 Telecom Variants

Telecom network designers are interested in serving the maximum possible number of clients with minimum cost. In another words, they want to make best use of the current network to earn more business. The cost is usually evaluated by link usage, or similar parameters in optimization objectives.

One of the most classic objective functions is to minimax the use of each link, that is:

$$\text{Minimize the value of } \max_{\forall i,j} \sum_{\forall s,d} F_{ij}^{sd}$$

This objective intends to arrange routing paths such that each link is equally used. The result provides a reference for the necessary capacity for each link in the network.

In this era, the use of K-Shortest Path routing is not yet popular due to its complexity. Alternatively, a set of primitive on-line algorithms are carried out as heuristics. The fixed routing method computes a path for each source-destination pair as the only entry in the routing table for the traffic. It will be blocked if part of its pre-computed path is occupied. Fixed Alternative Routing computes an alternative path in addition to the first fixed path. If the fixed path is not available, the network then attempts to set it on the alternative path. In addition, Adaptive Routing is an online method which finds a path according to the current network state.

The wavelength continuity constraint is another key factor in this problem. The network is constrained by wavelength continuity if no wavelength converters are used. This means that a traffic stream must be assigned to the same wavelength along its path. A node equipped with wavelength converters may switch the wavelengths for traffic streams passing it.

A graph coloring method is used to solve the wavelength assignment problem with this wavelength continuity constraint. In this approach, each stream in the network is modeled as a node. Two nodes are adjacent (connected by a link) as long as the two paths for the streams are overlapping. At this point, assigning wavelengths to the traffic streams so that they don't conflict is equivalent to coloring the nodes such that two adjacent nodes can't have the same color.

This graph coloring problem is widely studied in graph theory. A global optical solution is very hard to compute, but suboptimal results can be obtained by sequential coloring. Sequential coloring is a greedy algorithm which starts at a random vertex and colors one more vertex at a step. In each step the coloring plan is selected with the criteria of minimizing the total colors possible. A sequential coloring method may achieve the optimal solution if the starting vertex, along with the coloring sequence, happens to be the optimal one.

The networking design problem with only a fraction of nodes equipped with wavelength converters are also studied. In this case the routing and wavelength assignment becomes different since the paths passing certain wavelength converter equipped nodes may be more flexible in wavelength assignments than others. Also, the gain of using wavelength converters is discussed. These problems are studied in [49, 50, 51, 52].

2.3.3 EON Variants

The concept of Elastic Optical network is described in [54]. It suggests that we can make flexible use of frequency bands to improve the throughput. In a fixed grid network, the frequency bands are fixed, along with the modulation schemes, and guard bands. It is obvious that by merging serval neighboring spectrum together, the guard bands can be saved. Furthermore, if the modulation schemes can be determined on the fly and adaptively, more bandwidth can be saved by using high-order modulation for large volume traffic. Thus, in EON studies, we can make dynamic decision for allocating frequency bands and modulation schemes so as to save bandwidth.

In EON studies, we prefer to aggregate traffic into neighboring slots in a link since neighboring slots can be "merged" and bandwidth gets saved. On the other hand, we tend to preclude spectrum fragmentation because it limits the flexibility of band assignment and hence reduces the benefit of spectrum merging.

Research work [39] proposes a modified algorithm for EONs based on K-shortest path and first fit scheme. For each candidate path, a weight parameter is introduced to evaluate the spectrum fragmentation if traffic is set on this path. Zhu [40] proposes a set of algorithms based on the K-shortest path algorithm. A variety of parameters including distance, number of slots, etc are considered in manipulating of paths. The blocking probabilities for these algorithms on a variety of networks are evaluated by simulations.

2.3.4 IP over WDM

The IP over WDM design is a packet switching network infrastructure where WDM technology serves as physical layer transmission media, while the routing is performed by IP

routers. Due to lack of processing power for optical bits, a WDM OXC can't directly forward the packet to its next hop dynamically by processing the header of the packet. The OXCs, alternatively, forward the packet to an IP router, where routing is performed.

Thus IP over WDM has a two layer structure. In the optical layer, the network is a set of optical links connected to a set of nodes, in which IP routers reside. In the IP layer, the network is a set of interconnected IP routers. Each IP router is connected to an OXC by several short reach (SR) interfaces. The OXC forwards the packets which need IP routing to the IP router through the SRs. The IP router performs routing, and sends it back to the OXC through an SR. The OXC further switches the routed packet from its SR to the outgoing port.

The scenario above describes the case when all optical links are mandatorily terminated by IP routers. On the other hand, optical bypass at an IP router may be allowed, which enables an optical link be directly connected to another by OXC, without going through the IP router. It is very similar to virtual circuits in an electronic network. In this case, the optical layer topography is different from the IP layer topography.

An MILP approach, aiming to minimize energy consumption in IP over WDM network, is proposed [41]. It adds some variables depicting the usage of fibers, channels, ports, and necessary EDFAs to compute and optimize power cost. Paper [43] proposes a model to evaluate the energy consumption with the consideration of making best use of renewable energy. In [42], heuristic routing is proposed to pinpoint energy and delay problems by balancing the tradeoff between shortest path IP routing and virtual circuit routing. Paper [46] proposes a set of methods to optimize the CAPEX of this infrastructure by adjusting the geographic positions for core and metro nodes. An MILP method and its heuristics are used in the optimization process. Zhang [44] proposes an MILP problem to minimize energy consumption using sliceable transponders. A set of variables are introduced to depict if a transponder is used by a connection. Suman [47] proposes an comprehensive MILP model to study both the cost and energy efficiency in IP-over-WDM networks. This model includes parameters for wavelength assignments, regenerators, and the association between virtual and physical layer networks.

The variations of the MILP model also covers the topic of survivable networks with the

IP-over-WDM infrastructure. Survivable networks are designed to guarantee that the virtual topography could remain after physical layer failures, by redeploying the virtual circuits. Lin[45] proposes a set of MILP models to study this reroute problem considering both topography and capacity limitations. Research [48] addresses both survivable and energy efficiency issues for a double link failure case. It uses greedy algorithms to establish virtual topology to make the network efficient and survivable. Efficiency and survivability are evaluated by simulation.

2.3.5 Summary

This section introduced the variation of network design models using ILP/MILP methods and discussed its inherent drawbacks in methodology. First, they are not scalable due to their computation cost. As mentioned, the number of F and A variables escalates with the scale of network and traffic. With more features included in the problem the total number of variables are usually integer multiples of this number. Consequently the total computation cost becomes more formidable, which goes further beyond the practical threshold.

Second, the computation time of the optimal solution may render the solution obsolete in dynamic traffic. ILP/MILP methods are off-line algorithms and need to know the traffic pattern in advance, which are modeled as the A variables in the constraint, before computing the solution. Note that the solution does not hold optimality if the traffic changes; instead a new optimal solution needs to be recomputed. The computation of the optimal solution for an obsolete traffic scenario is wasted.

In addition, network delay poses a great challenge to the promptness of the optimal solution due to its centralized mechanism. The ILP method needs to collect the A variables from every node. Thus, for each node, it needs to wait for at least two RTTs plus the computation time in order to be orchestrated by the central control plane. Thus, unstable delay or packet drop may prolong the decision time which is more likely to produce obsolete or wrong solutions, which potentially hampers the performance of the network.

If implemented as an online algorithm, a K-shortest path algorithm could provide a prompter answer for dynamic traffic. The routing paths for every source-destination pair

can be computed ahead of time and saved. Thus the routing process is certainly more efficient than the centralized optimization method. For new traffic, we don't need to recompute the routes in contrast to MILP approaches.

However, the optimality of the "pick from pool" mechanism of the K-shortest path algorithm is hard to evaluate. In most papers, the approaches using this algorithm are proposed as heuristic substitutes for the models while their optimality is rarely evaluated. The optimality is not guaranteed because, as mentioned, a considerable part of these process is usually a greedy algorithm.

We observed that both approaches are very limited in handling dynamic traffic. The ILP/MILP approaches are off-line algorithms and treat different traffic as a separate problem and they are solved separately. It is also complicated to evaluate the optimality of K-shortest path algorithm for different traffic. When handling bursty and dynamic traffic, MILP approaches are too expensive in computation and delay, which lacks scalability and promptness. On the other hand, on-line algorithms based on the K-shortest path algorithm are quick but its optimality is hard to evaluate.

To deal with this difficulty, we come back to the earlier thoughts which aim to solve the problem by combinatorics. Similar to the switching structure, we can study non-blocking network topographies for a WDM network, or in general, a mesh network. We can model the dynamic traffic in a combinatorial way and study if a connected mesh network could handle this traffic and thus be non-blocking. Knowing the non-blocking property, we can determine if a network is able to handle dynamic traffic. It is a comprehensive study of the relationship between the structure of the underlying simple graph of a network, traffic, and routing algorithms. This approach is more general than the ILP and heuristic algorithm approaches and are expected to provide solid results.

2.4 NON-BLOCKING OPTICAL NETWORKS

This section introduces the concept of non-blocking networks. We begin by the well studied topic of non-blocking switching structures and its main result–the Clos structure. Then we

summarize the literature on non-blocking studies of optical networks. Based on this, we propose and formulate our study of non-blocking mesh WDM networks.

2.4.1 Non-blocking Switch Structures

While non-blocking WDM mesh networks are a rarely studied topic, non-blocking general electronical switching structures have been widely studied. In practice, a large scale switch is usually implemented by a set of interconnected small scale switches, each of which has several incoming/outgoing ports. The switching structure is essentially the way its small scale switches are connected and they way they are integrated as a large scale switch. A non-blocking switch guarantees that a pair of idle incoming/outgoing ports can always be connected without affecting the traffic in other ports. If we assume the small scale switches are non-blocking, the switching structure has a deterministic effect of the non-blocking property of the large switch. Thus a general non-blocking structure is of great interest. We begin the study by defining terms "connection" and "traffic pattern" to describe traffic and non-blocking properties.

Definition 2.4.1. *A **connection** $\overrightarrow{x} = (s, d)$ is a vector that represents the one way traffic from the source to the destination, either routed or to be routed, for which input port s needs to be connected to output port d.*

Definition 2.4.2. *A **traffic pattern** $X = \{\overrightarrow{x_1}, \overrightarrow{x_2}, ...\}$ is a collection of connections that could exist simultaneously at an instant in the network*[2].

Here are some general rules for the traffic in a switch for this dissertation:

1. The switch, in general, has an equal number of input and output ports.
2. One input port, at an instant, can only send traffic(i.e. be electronically wired) to one output port, and vice versa.[3]
3. The wiring of a pair of input/output ports can be reconfigured when necessary.

We propose the following definitions to depict these rules in math.

[2]WLOG, a switching structure is a special kind of network. Thus we use the term "network" in the definition.

[3]Multicast traffic may have multiple destinations from the same origin. This type of traffic is not considered in this paper.

Definition 2.4.3. *Vertex v is **involved in traffic pattern** X if there exists a connection such that v is the source or destination vertex in the connection. If v is the source vertex, we say v is **involved as source** in X. If v is the destination, we say v is **involved as destination** in X. If v is neither the source nor destination of any $\overrightarrow{x} \in X$, we say v is not involved in X.*

Remark 2.4.4. *We can also say that v is **involved in a connection** if v is the source or destination of that connection.*

Definition 2.4.5. *Denote $\eta_s(X, v)$ as the number of connections in which v is **involved as source** in X. Similarly denote $\eta_d(X, v)$ as its counterpart in which v is **involved as destination**.*

We can consider the input ports as the source and output ports as the destination of a connection in a switching structure. Recalling that we regulate one input port can only be connected to one output port at an instant and vice versa, we have:

Proposition 2.4.6. *A traffic pattern X is **applicable** to a switching structure if and only if for all v, we have $\eta_s(X, v) \leq 1$ and $\eta_d(X, v) \leq 1$.*

Definition 2.4.7. *A **traffic set** $T = \{X_1, X_2, ...\}$ is a set of all applicable traffic patterns.*

The traffic set T for a switching structure must hold Proposition 2.4.6. This traffic set is also referred to as a *permutation traffic set* as all the traffic patterns in it can be represented by permutations. This paper studies the non-blocking property for the permutation traffic set and use permutations to represent traffic patterns in it. For a formal explanation as to why this approach is done, see Appendix A.2. Specially for the example of connections and traffic patterns, refer to Example A.2.3.

With the above definitions, we can formally define a series of non-blocking properties:

Definition 2.4.8. *A switching structure is **strict sense non-blocking (SSNB)** if, at any instant any arriving connection $\overrightarrow{x} \in X$ arrives (the current traffic pattern must be an $X \in T$), $\overrightarrow{x}$ can be routed via an arbitrary available path.*

Definition 2.4.9. *A switching structure is **wide sense non-blocking (WSNB)** if, at any instant any arriving connection $\overrightarrow{x} \in X$ arrives (the current traffic pattern must be an*

$X \in T$), $\overrightarrow{x}$ can be connected via a path determined by an algorithm.

Definition 2.4.10. *A switching structure is **rearrangeably non-blocking (RNB)** if, at any instant any arriving connection $\overrightarrow{x} \in X$ arrives (the current traffic pattern must be an $X \in T$), $\overrightarrow{x}$ can be connected connected via a path, which may need to rearrange paths of other traffic to make the path available.*

As can be observed, all non-blocking properties require that all traffic patterns in T be routed. SSNB is the strongest non-blocking property because it demands that the path can be arbitrarily taken and still guarantees non-blocking. WSNB is weaker than SSNB in that it requires that path be deliberately selected according to an algorithm instead of arbitrarily.

Rearrangeably non-blocking is the weakest non-blocking property. It is satisfied as long as there is one solution of routing paths for each traffic pattern. Once we have the solution for a traffic pattern, we can route the connections in the traffic according to the solution. Note that this process implies that some connections may be rearranged during the shift of traffic patterns in the network. And, hence, since we can rearrange routed connections during the change of traffic patterns, we only need to have one solution for each traffic pattern.

Remark 2.4.11. *We can prove the RNB property if we can find routing paths for every applicable traffic pattern. The traffic throughout the network must be an applicable traffic pattern at any instant. The term "rearrangeably" subtly implies that for the current traffic pattern, as long as there is one solution, no matter how other traffic is routed, we can rearrange the traffic according to the solution such that every connection is routed.*

However, for the WSNB and SSNB properties, we need to consider randomness in the routing algorithm and the arrival sequence of the connections, as a connection will be kept in the path at the instant it is routed, which will affect the routing process of subsequent connections, which introduces an astronomical amount of enumerations, and is beyond practical computation power.

Figure 3 illustrates the relationship between these non-blocking properties. The RNB property is the least restrictive property and the WSNB property is a subset of it. SSNB is the most strict property which is a subset of the WSNB property, and hence also of the

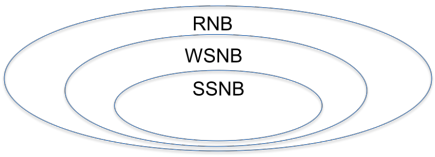

Figure 3: Venn diagram for non-blocking properties

RNB property.

For any of the non-blocking properties, a direct approach is difficult because the number of traffic patterns escalates with the number of ports. Computing the routing path for every traffic pattern is mandatory yet obviously non-scalable, even for the easiest RNB property, not to mention the SSNB and WSBN properties.

In the next section, we will introduce a general non-blocking switching structure known as the Clos structure. The analysis of the non-blocking properties in Clos structures can be simplified by its special way to connect the ports. By this structural advantage, there is a general routing method such that the routing paths for different traffic patterns can be summarized and represented by several integer parameters.

2.4.2 Clos Structure

A Clos structure (also referred to as a Clos network) is a modular 3-stage switching system which functions as a single $k \times k$ switch, where $k = r \times n$. A Clos structure can be characterized by r, n, and m. Its structure and the small scale switches it connects are plotted in Figure 4.

An example Clos structure with $r = 2, n = 3$ and $m = 4$ is shown in Figure 4. It functions as an $rn \times rn$ (6×6) switch and contains three stages. The first stage has two (actually r) $n \times m$ switches. Each switch connects n input ports to m output ports, each of which is connected to one of the m switches in the middle stage respectively. The switches in the first stage are also called **ingress switches**. The middle stage has four $r \times r$ switches. The third stage is symmetric to the first stage. Each switch in this stage forwards the traffic

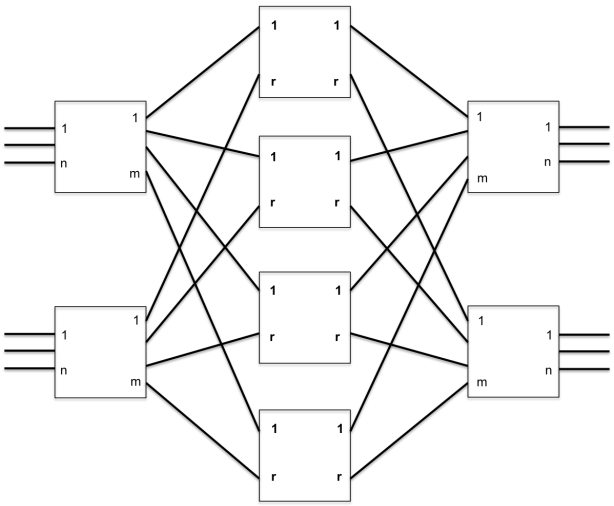

Figure 4: Example of a 3 Stage Clos Structure

from m middle stage switches to n outgoing ports. The switches in this stage are called **egress switches**.

Note that in general, there are r switches in the first and third stage and there are m switches in the middle stage. Usually r and n are fixed by their relationship to k however the selection of number m is flexible. We show later that the value of m can determine the non-blocking property for this structure.

The Clos structure has a special feature that each middle stage switch has one port connected to each ingress switch and one for each egress switch. It is impled that two connections switched to the same ingress/egress switch must be routed to different middle stage switches respectively.

The Clos structure is found to be SSNB when $m \geq 2n - 1$. The reasoning process is as following. For an ingress switch, there are at most $n - 1$ other calls handled by this switch and hence the n^{th} call may only have $m - (n - 1)$ middle stage switches to choose from. Meanwhile, consider the egress switch where the destination port is located. In the worst case there exist another $n - 1$ calls which use another $n - 1$ middle stage switches. This call could have its path through this switching structure as long as we have at least one more switch in the middle stage. Thus we have $m - 2(n - 1) \geq 1$; and we have $m \geq 2n - 1$. It

is also found that when $m \geq n$, this structure is RNB. For its proof refer to [53]. No clear relationship between the WSNB property and the parameters m and n is observed.

Note that a Clos structure can be expanded to more stages by replacing the $r \times r$ switches in the middle stage by another 3-stage clos structure. This would extend the number of overall stages to arbitrary odd number greater than three if k is sufficiently large. It adds great variation to the switching structures and thus addressing optimal route selection becomes even more difficult.

Regarding the path selection preferences, Benes proposes a conjecture in page 29 in [53] that "*a call should be routed through the most heavily loaded part of the network that will still take the call*". He offers discussion of this conjecture; however, he does not provide a generalized proof for it.

2.4.3 Overview of "Non-blocking" Studies

The study of Non-blocking optical networks can be traced back to the 1990s, principally focused on designing non-blocking optical switches. These studies extend the proof method of the Clos structure to optical switching devices. The features of optical networks are added to the analysis process such as wavelength conversion functions and the lack of optical buffers. This section begins with a brief review of the studies in this topic, and then proposes the problem and framework for our study.

The non-blocking network analysis can be categorized into two branches. The first branch focuses on the design of a non-blocking optical switch. The second branch studies the traffic and the performance of a network.

Qin and Yang[63] studied the performance of non-blocking switching structures with different wavelength converting functions. By comparing the permutation capacity versus the number of cross points using no/limited/full wavelength conversion functions, it is concluded that limited wavelength conversion could balance the cost yet achieve good performance.

Rasala and Wilfong[64] proposed a delicate design scheme for heterogeneous switching (have different numbers of input and outgoing ports) and demonstrated the lower bound

of wavelength converters to guarantee strictly non-blocking. This design is based on the assumption that the wavelength converters in a switch can be dynamically allocated to the ports or links where necessary.

Zhou and Yang [65] investigated wide-sense non-blocking networks for multicast traffic. It studies the routing method for multicast traffic in a variety of regular networks and summarized the minimum number of wavelengths needed to the WSNB property. Similar to the discussion of the Clos structure, this study summarized the routing results in different traffic patterns and computed the lower bound. The result is based on the assumption that the traffic is routed to the shortest path or according to a fixed path selection scheme.

Barry and Humblet[68] analyzed the performance of wavelength routing switches to handle partial and full permutation traffic sets. They model the traffic with their proposed prototype switching structures without explicitly mentioning the graph structure of the network. This delicate reasoning process bypassed the difficulty brought by the graph structure and routing method. The lower bound for the number of wavelengths are deducted for both passive and configurable all optical networks.

In addition, blocking rates are collected to evaluate the performance of a network. They are either computed by probability models or collected from simulation results. Particularly in OBS systems, queue theory methods are widely applied to compute the blocking probability.

Barry and Humblet [66] computed the blocking rate along a path of a network using probability methods. The analysis is based on a routing path between two nodes while other traffic using part of the path is modeled without knowledge of the remaining section of the underlying simple graph of the network. It evaluates the effect of path length and number of wavelengths towards the blocking probability either with or without wavelength interchangers. Based on this result, the benefit of wavelength converters in different networks are discussed.

Hsu and Liu[67] applied queue theory to study the blocking probability for an OBS switch. It proposed a two stage queue by the Markov model and computed its loss probability. In addition, a set of simulations are conducted. The simulation results are compared with the analytical result to discuss the accuracy of the queue model to predict traffic.

In summary, the core of the non-blocking switching studies is based on the result of Clos structure. New elements in optical networks are added to the structure and their impact towards the non-blocking property are discussed comprehensively. The problems are either formed as a variation of Clos structure so that its result can be applied, or formed as a resource allocation problem to reach the minimum resource needed.

In the studies of blocking probabilities, most studies use properly defined probability models, simulations, or both. As the infrastructure of an optical network is complicated, the probability model is in general complex, and the solution may be approximated. Simulations may be conducted as a control group to better evaluate probability models.

2.4.4 Non-blocking Optical Mesh Network

Similar to study of the non-blocking switching switching structures, the analysis of non-blocking optical mesh networks is based on the same definition and analysis framework. However, there are two differences between the structure of a mesh network and a general switching structure. First, a link in a general mesh network works both ways, in contrast to its counterpart in a switching structure, which makes routing analysis totally different. Second, every node generates traffic in a mesh network and there are no middle stage nodes in contrast to a switching structure, which poses a more stark constraint for traffic analysis. Obviously, they render the analysis of the routing problem more complicated and the result of Clos structure can not be directly applied. Despite the differences in analysis, we continue to use the definitions for switching structures to the study of non-blocking mesh networks due to their inherent conceptual consistency.

This paper aims to study non-blocking properties in a mesh network. We model the mesh network as a simple graph by assuming there is only one pair of input/output ports at each node, and there is only one identical wavelength in every link[4]. To get a more general result we also assume this network is free of wavelength converter as there is only one wavelength. The applicable traffic patterns in this network are similar to that of a switching structure,

[4]For the discussion of why this model could facilitate designing a non-blocking network without loss of generality, see Appendix A.

a permutation traffic set. We investigate the non-blocking property of the simple graph by studying whether all permutation traffic patterns can be routed and the routing methods.

In this dissertation we found that a subset of the graph may have non-blocking properties by certain routing methods, which is a structural advantage that is meaningful to the theoretical study of non-blocking networks. The scale of the non-blocking graph in the subset may be up to any scale. What's more, the routing method to achieve the non-blocking property is shown to be contradictory to Benes' conjecture. These contents are introduced in Chapter 4.

2.5 SUMMARY

This chapter provided a brief overview of WDM networks and proposed the main problem in this dissertation, which is the study of non-blocking properties in optical mesh networks. This problem is essentially a comprehensive investigation of a network's power to route different traffic. There are two motivations of this study. First, the non-blocking switching techniques have been widely studied. The theoretical background for its counterpart in mesh networks is ready. Second, the current network design and traffic engineering schemes have several drawbacks in implementation, either in complexity or infrastructure (Section 2.3). Thus, investigating non-blocking mesh networks would be very promising to get over this obstacle and to provide a solid, yet general, understanding of the capability of general connected graphs to handle dynamic traffic.

3.0 GRAPH THEORY STUDIES REGARDING NON-BLOCKING NETWORKS

Graph theory is a branch of study of discrete mathematics, which includes a set of practical problems that can be modeled by graphs. Among the realm of graph theory studies, we discover two closely related topics to non-blocking properties: the connectivity/cutset, and the Hamiltonian property. We compare these studies to the RNB property to investigate their in-depth coherence. In addition to the traditional graph theory topics, we propose our own lattice approach to study the RNB property and present several general facts.

The non-blocking properties can, to some extent, be interpreted as multiple routing problems, which can be studied with graph theory methods. In this chapter we compare the least demanding RNB property to the graph theory studies. As stated in Definition 2.4.10, an RNB graph should route every traffic pattern, where a traffic pattern is a collection of connections. Thus the RNB property for a graph could be formulated as a complex multiple routing problem for a variety collections of connections in the graph theory scope, which is an in-depth structure property for connected graphs. While connectivity and Hamiltonian property are also structure properties, we compare the RNB property to them.

3.1 CONNECTIVITY AND CUTSET

The classic connectivity test examines the *minimum cut set* for a graph. In general, a cut set of a graph refers to a vertex cut set of the graph. A set of vertices S of graph G is a vertex cut set if the subgraph induced by the set of vertices $V \backslash S$ is disconnected. Note all edges incident to a vertex are removed upon the removal of the vertex.

The number $\kappa(G)$ is the minimum *cardinality*[1] of a vertex cut sets of graph G, which is also called the **connectivity** of a graph. We note that we do have the number $\kappa_e(G)$ as the minimum cardinality of the edge cut sets of graph G, and the number $\delta_{min}(G)$ as the minimum degree of vertices in graph G. There is a basic theorem indicating that $\kappa(G) \leq \kappa_e(G) \leq \delta_{min}(G)$. In general, a graph G can be called k–**connected** if it has vertex connectivity k.

The following related studies indicate how connectivity describes the structure feature of a graph. The connectivity values have distinct advantage that can be computed in polynomial time[3, page 236] for both vertex connectivity and edge connectivity. Thus, it is efficient to obtain the connectivity values. Whitney's theorem gives the structure property of a graph with connectivity value k.

Whitney's Theorem[4, page 234] indicates that:

Theorem 3.1.1 (Whitney)**.** *A non-trivial graph G is k-connected if and only if $\forall u,v \in V$, $u \neq v$, $\exists$ in G at least k internally disjoint u-v paths.*

This theorem indicates that for any pair of vertices, there are always k vertex–disjoint paths between them if a graph is k-connected. For any pair of vertices, ideally, a connection between them has k mutual disjoint paths to choose. This property also indicates the removal of any $k-1$ vertices in the graph would always render the remaining vertices connected.

However this theorem has limited power to investigate the RNB problem especially when there are multiple traffic patterns present in the network, because it is hard to determine the paths of the traffic from other vertices. They may take part of k candidate paths. In the worst case if all paths are taken by other traffic, the traffic between this pair of vertices may be blocked.

Consider a graph G in which there are k vertex–disjoint paths between vertices v_s and v_t, with vertex set V where $v_s, v_t \in V$. Further, let v_s and v_t be not incident to each other, and let the shortest distance between v_s and v_t be 3. Thus any of the k vertex–disjoint paths brought about by Whitney's theorem would be in the form: $v_s, v_i^s, ..., v_i^t, v_t$, where v_i^s and v_i^t are two vertices adjacent to v_s or v_t respectively on the i^{th} path. The traffic between v_s and

[1] Cardinality means the number of elements in a set.

v_t would be blocked if every pair of vertices (v_i^s, v_i^t) has traffic between them and the traffic is routed on the $v_i^s, ..., v_i^t$ part of the i^{th} candidate path between v_s and v_t respectively. As the traffic is variant, it is hard to determine the RNB property from the structure property provided by Whitney's theorem. This case depicts the limitation of applying Whitney's theorem to the RNB property.

Dirac[7] proved that a k-connected graph contains a cycle through any given k vertices if $k \geq 2$. This fact implies that a k-connected graph could guarantee a ring pattern transmission for up to k vertices. The sequence of the k vertices is not specified. This means given a set of k vertices, there may be one sequence of them that are guaranteed in the cycle. This means that a k-connected graph would always guarantee $\binom{n}{k}$ of the $n!$ traffic patterns. It definitely does not cover all the traffic patterns.

In addition, this theorem does not cover the case in which two or more cycles are needed to route a traffic pattern. For example, traffic pattern $(a_1, a_2, ...)(b_1, b_2, ...)$ may be routed into two cycles, which is not covered by the theorem. What's more, the non-specified sequence may raise a critical limitation. For example, the cycle guaranteed by the theorem to traverse the vertex–set $\{v_1, v_2, v_3, v_4, v_5\}$ may be $(v_1, ..., v_3, ..., v_2, ..., v_5, ..., v_4, ..., v_1)$; thus, a traffic pattern in a different sequence, for example a cycle in the form of $(v_5, v_4, v_3, v_2, v_1)$ may not be routed because this sequence is not guaranteed. These examples indicate the limitations of applying Dirac's theorem to the RNB problem.

Finding k vertex or edge disjoint paths for k pairs of different prescribed vertices is also studied in this topic. If the paths are vertex–disjoint, the graph is said to be k**-linked**. If the paths are edge–disjoint, the graph is said to be weakly k**-linked**. The study from[8][9] suggests that, for each integer k, there exists an integer $f(k)$, such that graph G is k-linked given $\kappa(G) \geq f(k)$. If these paths exist, it may satisfy the traffic patterns which only have pairwise connections. There are at most $\lceil \frac{k}{2} \rceil$ pairs of vertices. If the connectivity κ is great enough to make the graph $\lceil \frac{k}{2} \rceil$-linked, we know from the structure property that there exist a solution for pairwise communication traffic patterns. However the bound for κ to be " great enough" is not specified. How to find the specific form of the bound $f(k)$ is not mentioned. The bound may also be so large that it is beyond the cost budget for industry investments. However, by assuming that the network only has pairwise connections, this concept has more

power.

Connectivity tests study how strongly a graph is connected, which has a very subtle relationship to the RNB property. However, there are differences. The relationship between connectivity testing and other approaches to the non-blocking network design problem is very complicated. The comparison between connectivity and RNB property is discussed in Section 3.4.1. Besides connectivity, the Hamiltonian graph is another widely studied graph–theoretic topic concerning structure. This topic is discussed in the next section.

3.2 THE HAMILTONIAN PROPERTY

The Hamiltonian property is a classic graph theory problem. The Hamiltonian property investigates if there exists a cycle or path that passes through every vertex in the graph once and only once. Specially, the concept of **Hamiltonian graph** is defined by such cycles. Mathematically: A graph G is said to have the Hamiltonian property if G has a spanning cycle that includes every vertex in G. In addition, the **Hamiltonian path** concept refers to the path obtaining that property. Although finding Hamiltonian cycles or paths seems easy, it is very complicated to determine its sufficient and necessary conditions.

The studies about the Hamiltonian property are interesting because the approaches to Hamiltonian graphs may inspire a heuristic for the RNB problem. They both find cycles/paths in the graph. The Hamiltonian property guarantees a cycle/path trespassing every vertex while the RNB property subtly demands a variety of cycles. Thus, there are great similarities between the Hamiltonian property and the RNB property. Although determining the Hamiltonian property is computationally complicated, plenty of sufficient conditions to it have been found. Thus we expect its counterpart in the RNB problem can be found.

It is worth mentioning that in some branch of the Hamiltonian study, stronger statements are discussed, such as " cycle extendable", "k-ordered Hamiltonian", " Hamiltonian property preservation", etc. These stronger statements are even closer to the RNB property. Their impacts towards building a RNB backbone problem are discussed at the end of this section.

3.2.1 Approaches to the Hamiltonian Property

There are two main approaches to the Hamiltonian property. One is based on the size of the independent set derived from the graph, which is also known as the independence number. It is used to evaluate mutual connectivity for a simple graph. The other is based on a "neighbor" concept which depicts the size of the neighboring vertices of a vertex set. Based on these parameters, several sufficient conditions to the Hamiltonian problem are developed.

A set of vertices $X \subseteq V(G)$ is said to be *independent* if, for any pair of vertices $v_1, v_2 \in X$, v_1, v_2 are NOT incident. The notation $ind(G)$ is defined as $ind(G) = \max ||X||$ where X is an independent set of G. The parameter $\sigma_k(G)$ is widely used in Hamiltonian problems. It is defined as:

$$\sigma_k(G) = min\{\sum_{i=1}^{k} deg(x_i)\} \text{ where } X \subseteq V(G),\ X = \{x_1, x_2, ..., x_k\} \text{ is an independent set}$$

Based on this parameter, Ore[10] proved that, if $\sigma_2(G) \geq ||V(G)||$, G is Hamiltonian. Later, Ore[11] further proved that, if $\sigma_2(G) \geq ||V|| + 1$, G is Hamiltonian connected. Jackson[12] proved that a d-regular 2-connected graph with $d \geq \frac{||V||}{3}$ is Hamiltonian. Bauer[13] found that a balanced bipartite graph G is Hamiltonian if $deg(u) + deg(v) \geq ||V|| + 1$ where u and v are not incident and belong to different parties respectively. In [3, 14], it is found that a 2-connected graph with $min\{max(deg(u), deg(v))|d(u, v) = 2\} \geq \frac{||V||}{2}$ is Hamiltonian. In[15] it is mentioned that a 2-connected graph G is Hamiltonian if $\sigma_3(G) \geq ||V|| + \kappa_v(G)$. In [16] it is stated that, for a graph G with $||V|| \geq 3$, $\kappa_v(G) \geq ind(g)$ implies the Hamiltonian property; $\kappa_v(G) \geq ind(g) + 1$ implies the Hamiltonian connected property; $\kappa_v(G) \geq ind(g) - 1$ implies the traceable property.

These facts determine the Hamiltonian property by analyzing the independent number in combination with other graph metrics. It can be summarized that the studies mentioned above establish a sufficient condition for Hamiltonian property by manipulating degree, $ind(G)$, and σ.

The studies in the "neighbor" approach to the Hamiltonian property are focused on the number of neighbors $|N(S)|$ for a subset of vertices $S \subseteq V$. Woodal[17] proved that, for any subset $S \subseteq V$, G is Hamiltonian if $|N(S)| \geq \frac{||S||+||V||+3}{3}$. Fraisse[18] proved the Hamiltonian property given the assumption that $\exists\ t \geq \kappa_v$, such that for any independent set

$||S|| = t$, $|N(S)| \geq \frac{t(||V||-1)}{t+1}$. These two statements have high theoretical value and develop connections between graph structure and the Hamiltonian property which can be described numerically. However the implementation values are limited, by the inherent difficulty to be implemented. For example, to test the conditions for [17], it is required to find and test every subset of the vertex set $||V||$. Also, a significant amount of computation is needed to find the parameter t in the aforementioned statement[18].

In addition, the concepts of *graph powers*, and *line graphs* are related to the Hamiltonian property. A *line graph* transforms a graph G into another graph denoted by $L(G)$ by converting edges in G into vertices in $L(G)$. The two vertices in $L(G)$ are adjacent if and only if the corresponding edges are incident. The *k-power* graph of G can be generated by the k-th power of the adjacent matrix B of G. To obtain the k-th power of B, we need to set all diagonal elements to zero and all other non-zero elements to one in B^k. In the k-th power graph of G, all vertices within distance k of G are adjacent in the power graph.

Chartrand [19] proved that given a connected graph G with $\delta_{min}(G) \geq 3$, $L^2(G)$ is Hamiltonian, where $L(G)$ is the line graph of G. Fleischner [20] reached the statement that, for a 2-connected graph G, G^2 is Hamiltonian. For a connected graph G, [21] found that G^3 is Hamiltonian connected. Komlós [27] proves that for a sufficient large graph with $\delta_{min}(G) \geq \frac{kn}{k+1}$, G has the k-th power of a Hamiltonian cycle.

Line graphs and power graphs study the Hamiltonian property of a transformed general graph while they are two ways of transformation. Similarly, we may be able to generate RNB graphs by applying some trick or modification of a general graph. This approach is worthy further investigation.

The facts mentioned above can be classified as the traditional approaches to the Hamiltonian property as their target is the Hamiltonian property. The metrics and parameters used above to reach the Hamiltonian property might also be illuminative to the RNB property in a heuristic. It can be seen that the RNB property is more demanding than the Hamiltonian property.

In addition to the traditional approaches, advanced Hamiltonian studies focus on more demanding properties based on the Hamiltonian property. These studies may be closer to the RNB property. This topic is introduced in the next subsection.

3.2.2 Advanced Hamiltonian Study

In this section we introduce studies that research more demanding yet complicated properties based on Hamiltonian graphs. The content of this subsection concentrates on studying multiple cycles in Hamiltonian graphs which are more demanding because Hamiltonian graphs require only one such cycle. They are expected to be more similar to the RNB property.

A *pancyclic graph* contains cycles of all lengths between 3 and $||V||$, given $||V|| \geq 3$. A bipancyclic graph contains all even–length cycles between 4 and $||V||$. Bondy [25] finds that a Hamiltonian graph is pancyclic if $||E|| \geq \frac{||V||^2}{4}$ unless the graph has a balanced bipartite subgraph. For balanced bipartite graphs, [26] finds two sufficient conditions to bipancyclic property.

A graph G is said to be *cycle extendable* if any cycle C in $||G||$ with $||V(C)|| < ||V||$ can be extended to a cycle C_1 with $||V(C_1)|| = ||V(C)|| + 1$. Moreover, if G is cycle extendable and every vertex belongs to a triangle, G is said to be fully cycle extendable[3]. Hendry [28] investigates the cycle extendable property with graph metrics $\sigma_2(G)$ and minimum degree of G.

A **k-ordered (Hamiltonian graph)**, has a cycle (Hamiltonian cycle) for every sequence of k vertices that encounters the vertices of the sequence in a given order. Kierstead [29] found if $||V|| \geq 11k-3$ and $\delta_{min}(G) \geq \lceil \frac{n}{2} \rceil + \lfloor \frac{k}{2} \rfloor - 1$, G is k-ordered Hamilton. Faudree [30] proved that for $3 \leq k \leq \frac{n}{2}$, if for any non-adjacent vertices pair u and v, $deg(u) + deg(v) \geq n + \frac{3k-9}{2}$, then G is k-ordered Hamilton.

In addition to the facts mentioned above, [3] also summarizes a variety of related topics of Hamiltonian property. These include multiple Hamiltonian cycles, Hamiltonian decomposition, forbidden graphs, random graphs etc. Most of the approaches are not close to the RNB problem and thus not listed in this paper.

The relationship between the Hamiltonian graphs and the RNB graphs is very complicated. The Hamiltonian property guarantees cycles in a simple graph. RNB property requires cycles in different patterns in the simple graph, each edge from which is considered a two way link. The RNB property is more demanding in the specification of cycles. However, finding cycles in two way links are easier. These facts will be discussed in detail in

Section 3.4, after some typical RNB graphs and blocking graphs are introduced in the next section.

3.3 A LATTICE APPROACH

Graph theory analysis in the above topics is not a perfect matching to our RNB problem study because the RNB property is strong and complicated. Hereby we introduce our own lattice approach to study the RNB property. We begin by studying the RNB property for small scale simple graph because brute force computation for them is practical. We can record the non-blocking property of the small scale graphs and summarize them into a lattice structure.

The lattice contains non-isomorphic graphs as its elements. The graphs in a lattice are associated by a partial order relationship. Here we use the subgraph relationship as the partial order relationship to describe the similarity of graphs. By studying the RNB property of the graphs in the lattice, it is expected to shed light on the relationship between RNB property and structure similarities in graphs. The details of this approach are introduced in the following subsections.

3.3.1 The Partial Order Relationship in a Lattice

In the lattice structure in this section, we use the subgraph relationship as the partial order relationship to build the lattice. This relationship is partial order because it not applicable for every single pair of graphs. That is, there may be two graphs such that no one is the subgraph of the other. It is ordered because if G_1 is the subgraph of G_2, G_2 is the subgraph of G_3, we know G_1 is the subgraph of G_3.

For example, consider set S_5 contains all non-isomorphic connected graphs with five or fewer vertices. A lattice $(L, \leq)$ can be built with S_5 with a subgraph relationship, as shown in Figure 5. Consider $G_1, G_2 \in S_5$; if G_1 is a subgraph of G_2, then we have $G_1 \leq G_2$. The 5-vertex complete graph K_5 is the maximum element of the lattice while K_2 is the minimum

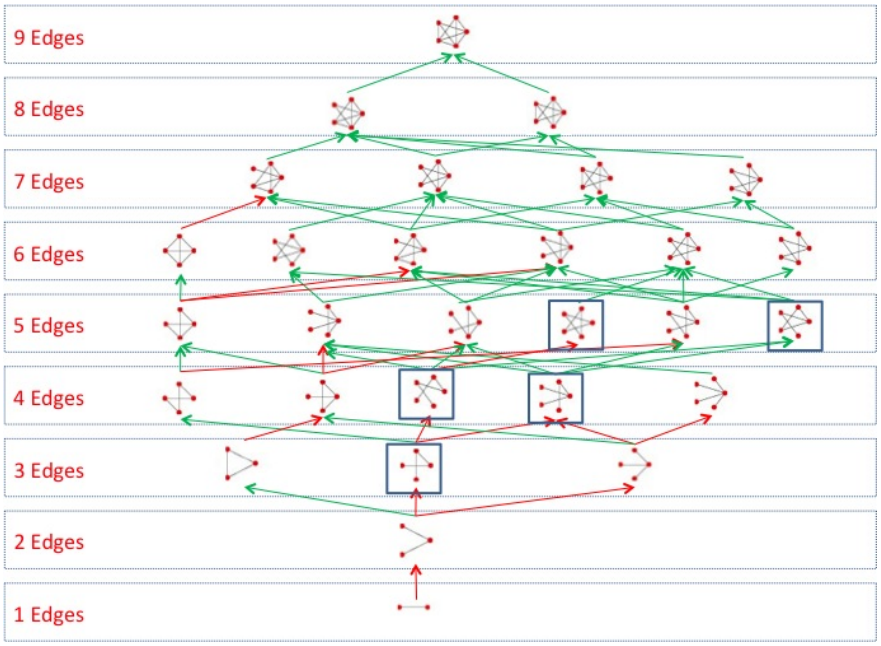

Figure 5: Lattice built on five or less vertices graph with subgraph relationship

element of the lattice[2]. Note if two graphs have the same number of vertices and the same number of edges, they don't have this subgraph relationship.

Join and meet operations can be defined as following. Join operation $a \vee b$ returns the least element (the lowest order) c such that $a \leq c$, and $b \leq c$. Similarly, meet operation $a \wedge b$ returns the greatest element c such that $c \leq a$, and $c \leq b$. From the graph structure scope, the join graph $a \vee b$ inherits the graph structure from both graphs. In the same way, the meet graph $a \wedge b$ represents the most common graph structure in both graphs. These operations are used in the analysis of the graph similarity.

3.3.2 An Example Lattice

Figure 5 presents the lattice structure defined with S_5 and the subgraph relationship. The subgraph relationships are shown by arrows colored either green or red. A green arrow's

[2]The 1-vertex graph is trivial and not included in S.

tail points to a spanning subgraph of the graph at the head. The graph at the tail has one less edge than the graph at the head. The graph at a red arrow's tail can be generated by removing a one-degree vertex in the graph at the red arrow's head. For the convenience of representation, other subgraph relationships are not plotted as they can be implied from the plotted relationships.

It can be intuitively observed that all graphs share a common structure–K_2–that's how a graph gets itself connected. And all graphs inherit part of the structure from the complete graph K_5. In math languages: $\forall a,b \in L$, $K_2 \leq a \wedge b$, and $a \vee b \leq K_5$.

The blocking graphs are enclosed in a box while the others all have the RNB property. It can be observed that all blocking graphs contain a subgraph–the blocking graph with three vertices and two edges–a short chain. By some simple reasoning and proof[1], the following intuitive rules have been summarized:

1. Chain graphs with greater than or equal to three vertices are blocking.
2. Ring graphs with greater than or equal to five vertices are blocking.
3. Star graphs are RNB.

Chain graphs are connected graphs with n vertices and $n-1$ edges, $n > 3$. Among the n vertices there are two 1-degree vertices and $n-2$ 2-degree vertices. All chain graphs can be found blocking because, when the two 1-degree vertices are sending traffic to each other, all links are used and the traffic between any of the remaining n-2 vertices will be blocked.

A ring graph has n vertices and n edges, where all n vertices have degree 2, which is essentially a cycle with n vertices. Ring graphs are blocking when $n \geq 5$. For example, consider the 5 node ring graph in Figure 6. A blocking traffic pattern can be found by letting all vertices send traffic to a vertex next to its neighbor in sequence, which could be in permutation form as (1 3 5 2 4). It can be observed that this traffic pattern will be blocked.

A star graph is an n-vertices tree with $n-1$ leaves. For its special structure, the connections in the graph can be routed as following. If the traffic is between two leaves, it can be hopped by the root vertex, which lies right between the source and destination. If the traffic is initiated from or destined to the root, this traffic can be delivered directly. For every traffic pattern, its connections can be routed by the method above. Thus every traffic

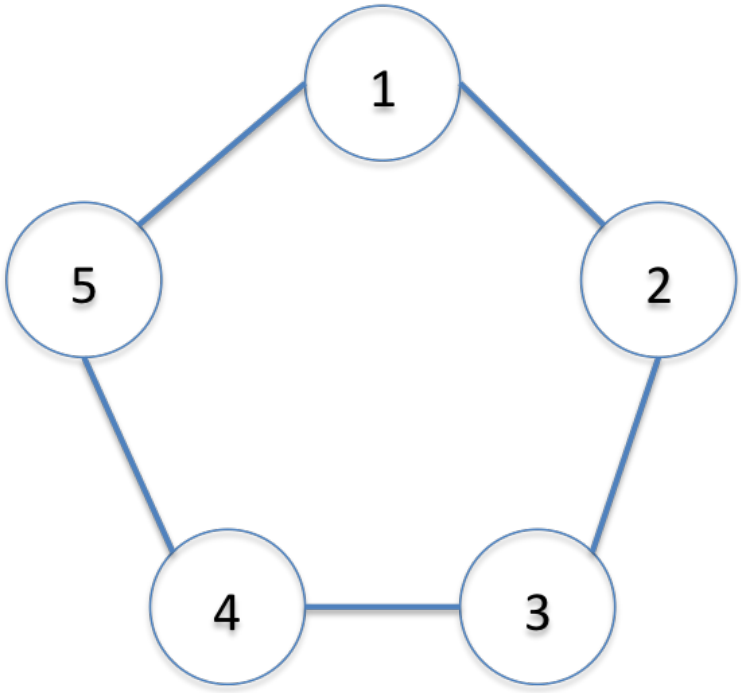

Figure 6: Example blocking ring graph

pattern can be routed and hen star graph is RNB. The detail of the formulation and the formal proof of this statement can be found in the next chapter.

The Chain graph rule and Ring graph rule can further be developed into a more general form:

Theorem 3.3.1. *Cut Set Theorem: For a connected graph G if there exists an edge cut set E_c such that $||E_c|| < min\{||C_1||, ||C_2||\}$, then G is blocking. C_1, C_2 are the two components E_c cuts G into.*

Proof. Denote vertices $v_i^1 \in C_1$, and $v_i^2 \in C_2$. We know there is a traffic pattern containing connections (v_i^1, v_i^2) and (v_i^2, v_i^1) where $i \leq min(||C_1||, ||C_2||)$. This traffic pattern can NOT be routed because each pair of the connections needs to be routed by an edge (two links, one in each direction) connecting C_1 and C_2. However there are only $||E_c||$ edges connecting them which is strictly less than $min\{||C_1||, ||C_2||\}$ which is the number of such pairs of connections in the traffic pattern. We know there must exist a pair of connection such that they have no link to use. Thus this traffic pattern can't be routed and the graph is not RNB. The proof is complete. □

Theorem 3.3.1 is a sufficient condition to a blocking graph. Alternatively it suggests that an RNB graph must be *at least* sufficiently connected so that the graph can't be cut into two large parts by a small number of edge removals. It is not a necessary condition to be a blocking graph because we have found an exceptional blocking graph which does not satisfy the theorem.

The exceptional graph has six vertices, which was found in an extensive study of a lattice structure consisting of 6-vertex graphs. We observed four greatest blocking graphs among 6-vertices graphs, which are shown in Figure 7. Three of them satisfy Theorem 3.3.1 and the exceptional one is labeled by a box. For this blocking graph, we can't find an edge cut set that cut the graph into two parts such that either part is "bigger" than the edge cut set. Thus we could see that only a portion of blocking graphs satisfies Theorem 3.3.1 and thus the cut set theorem is only a sufficient condition for a blocking graph.

The lattice approach is an intuitive way to investigate the RNB problem. It arranges a set of graphs according to their structure similarity, if we use subgraph as the partial order

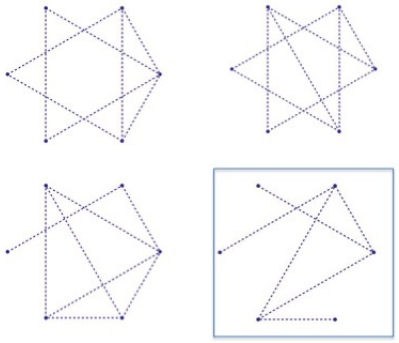

Figure 7: 6-vertices Blocking Graph

relationship, so that we can conveniently observe the relationship between the RNB property and their structures.

The advantage of the lattice approach is its simplicity and flexibility. The disadvantage is the computational complexity due to two facts. First, the number of non-isomorphic graphs grows rapidly with the number of vertices in the graph. Second the computation complexity to determine the RNB property increases drastically with the number of vertices in the graph as well. Thus the scales computable graphs are limited. While it could be possible some of the rules may NOT be obvious under this lattice with small-scale graphs, the future research directions with this method may be:

1. Build up a lattice with new partial order relationships, such as " split/contraction".

2. Build up a lattice with more graphs, which needs more computational power.

Split/contraction is another relationship between two different graphs. A contraction for a connected graph with at least two vertices is to merge two adjacent vertices into one vertex. The new vertex is connected to all the neighbor vertices of the two merged vertices. Split is the inverse operation of contraction. In this relationship, it can be seen that the graphs contracted from an RNB graph are all RNB graphs. By applying contraction, the contracted graph has one less vertex than the original graph. This is very similar to the subgraph relationship and may be used to build up a lattice to study the RNB property.

3.4 SUMMARY

This section summarizes the three topics in this chapter–the RNB property, the connectivity tests, and the Hamiltonian property. We compare the RNB property and other two topics. As the main topic for this paper focuses on the problem of designing non-blocking networks, the comparison between connectivity tests and Hamiltonian graphs are omitted.

Recall that a simple graph G is said to be RNB if a simple graph G could "route" every possible traffic pattern while each edge in G works in two directions. The term "traffic pattern" and its routing are defined in detail in Section 2.4. It is a structural property of a graph which represents its capability to handle dynamic traffic. Connectivity tests study the firmness of a simple graph, which is also a structural property. The comparison between connectivity and RNB are discussed in Section 3.4.1.

Section 3.4.2 discusses the relationship between the RNB property and the Hamiltonian property. The Hamiltonian property guarantees a cycle or path passing all vertices in a simple graph. On the other hand, a traffic pattern (in the form of a permutation) has traffic cycles, the routing paths of which must be cycles. While each link has two directions, the cycles can be find in the symmetric associated digraph of the graph. Thus we can see, to some extent, that the RNB property is another "cycle finding" game and has more restrictions of the cycles which comes from the amount and variety of traffic patterns. However, due to cycle finding in the symmetric associated digraph is easier than in an undirected graph, RNB graphs are not a subset of Hamiltonian graphs. That is, in RNB property, the traffic is routed to cycles in the graph in which each edge has two directions and may handle two traffic streams in different directions and thus be a part of two cycles at the same time. On the contrary, edges in a Hamiltonian graph can only be used once for a cycle. Given the similarity and trickiness, we conduct a scrutinized study toward the relationship between Hamiltonian graphs and RNB graphs.

3.4.1 Connectivity vs RNB

Connectivity represents how strongly a graph is connected, that is, how many vertex/edge removals can be made before the graph becomes disconnected. This can be directly applied to the study of designing a resilient network. In general a graph with high connectivity is less likely to be blocking.

However, it is imprecise to determine the RNB property based merely on the connectivity value. That is, we have found blocking graphs with good connectivity, for example, the exceptional blocking graph in Figure 7. Nevertheless, it is observed that some RNB graphs do have low connectivity values. For example the star graphs are RNB but only have edge connectivity value of 1.

Based on the cut set theorem (Theorem 3.3.1), we can subtly connect edge connectivity to the RNB property. We offer a Venn Diagram in Figure 8 to discuss the differences between the edge connectivity and the RNB property.

In the Venn Diagram, the universe is the set of connected graphs–also 1-connected graphs. It can be observed that the cut set theorem observed from the lattice approach is a necessary condition to the RNB property. That is, all RNB graphs could pass the cut set theorem test, that is, they can't be cut into two large parts with a small number of edges. However there are some blocking graphs that still pass the cut set theorem test. A typical example of this is the exceptional blocking graph shown in Figure 7, as mentioned above.

The edge connectivity test overlaps the RNB graphs. The set of $\kappa_e = 2$ is plotted for illustration. It consists of both non-blocking and blocking graphs. While the non-blocking graphs are easy to observe, we only provide some examples for blocking graphs. First, the 5-vertex ring graph has $\kappa_e = 2$ and blocks. In addition, all the three unboxed 6-vertices greatest blocking graphs in Figure 7 also have $\kappa_e = 2$.

The relationship between edge connectivity and bipartite complete graphs is illustrated in Figure 8. Some bipartite complete graphs have very low edge connectivity values. For example, a star graph is a bipartite complete graph with 1 edge connectivity. We thus know that some RNB graphs do not have edge connectivity 2. The second 5-vertex, 5-edge graph from right to left in the lattice (Figure 5) is an example of 1–edge connectivity RNB graph

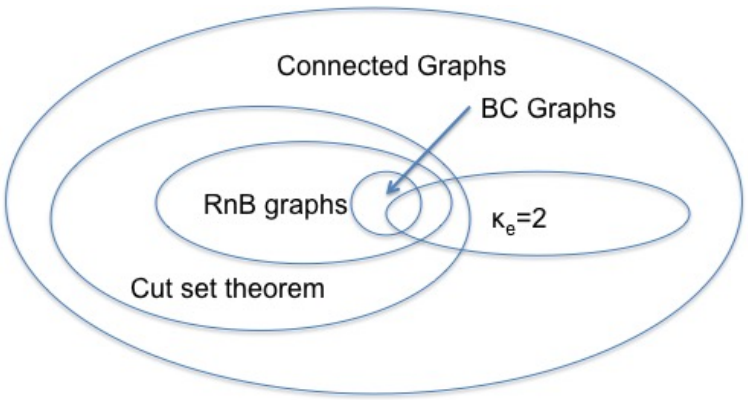

Figure 8: Venn Diagram for the relationship between connectivity tests and RNB graphs

that is not bipartite complete. Clearly there are some bipartite complete graphs with high connectivity. It can also be observed that the number κ_e for a bipartite complete graphs is essentially the minimum number of vertices in the parties. Thus for the set of bipartite complete graphs, it may have arbitrary edge connectivity number while they are RNB graphs.

Despite the edge connectivity value κ_e itself does not directly imply the RNB property, especially when this value is low, we can deduce that, if a simple graph has edge connectivity more than $\lceil \frac{||V||}{2} \rceil$, this graph passes the cut set theorem test. That is, this graph is more likely to be RNB. However, whether there is a blocking graph with $\kappa_e \geq \lceil \frac{||V||}{2} \rceil$ is currently a hard problem. If there is no such graph, we may filter out RNB graphs using the condition $\kappa_e = \lceil \frac{||V||}{2} \rceil$.

To summarize, the relationship between connectivity value and the RNB property is tricky. The cases are complicated and it is imprecise to conclude the RNB property based merely on connectivity values. However, we can take the advantage of the cut set theorem and the computation efficiency of obtaining the edge connectivity to obtain a set of graphs that are more likely to be RNB.

3.4.2 Hamiltonian vs RNB

The Hamiltonian property guarantees that a graph has a cycle or path with every vertex in it. This is an excellent structure property. And it is closer to the definition of the RNB property. It promises that a Hamiltonian graph could route at least two traffic patterns, in which the vertices are sending and receiving traffic according to the sequence in the Hamiltonian cycle. However, the RNB property requires that a graph should satisfy every traffic pattern. The Hamiltonian cycle in a Hamiltonian graph may not support every traffic pattern, for instance, the 5-vertices ring graph. In addition, some traffic pattern may require multiple cycles to be routed. In this sense, the Hamiltonian property is less demanding than the RNB property.

The *advanced Hamiltonian property* studies multiple cycles in a graph. This concept is more demanding than the Hamiltonian property in the structure of the graph. It is more close to the RNB property in this extent. These advanced Hamiltonian concepts are summarized as following:

A *pancyclic graph* is more likely to route more traffic patterns because the graph has cycles of all lengths. This is stronger than the aforementioned Dirac's Theorem. Further, if a *cycle–extendable* graph routes traffic pattern $\{v_1, v_2, ..., v_k\}$ in a cycle, this cycle may route some traffic pattern $\{v_1, v_2, , ..., v', ..., v_k\}$, since the cycles can be extended. However both the pancyclic and cycle–extendable concepts do not consider the traffic patterns requiring multiple cycles simultaneously. Except for cycles of length $||V||$, pancyclic graphs and cycle extendable graphs do not guarantee the vertices in the cycle. Neither do they consider the sequence of vertices of the guaranteed cycle. In this way these two statements are close to, but less demanding than, the RNB property.

The *k-ordered* Hamiltonian property considers the sequence of vertices that lies in the cycle. However, it is not discussed that some specific traffic patterns may need multiple cycles to be routed. It is neither discussed whether a $2k$-ordered Hamiltonian graph has two cycles in which specific k vertices occur in sequence respectively. In this scope, the k-ordered Hamiltonian property is also weaker than the RNB property.

The relationship between the Hamiltonian graphs and the RNB graphs can be shown

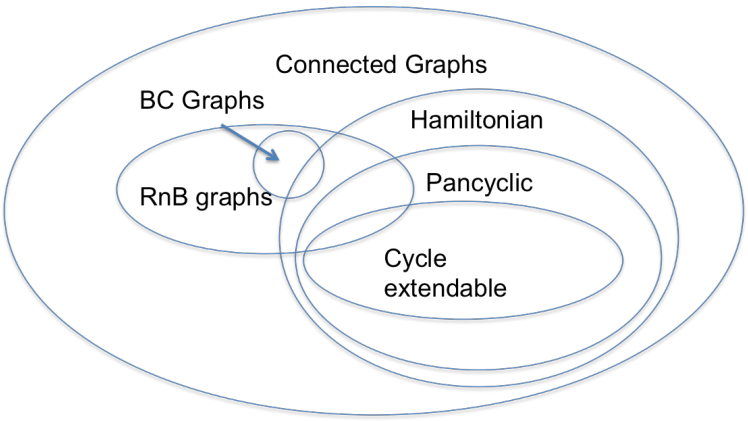

Figure 9: Venn Diagram for the relationship between Hamiltonian and RNB graphs

in the Venn Diagram in Figure 9. Both pancyclic graphs and cycle extendable graphs are subsets of the Hamiltonian graphs. Some Hamiltonian graphs are not RNB, such as the 5-vertex ring graph. Some RNB graphs are not Hamiltonian, for example, the star graphs.

Graph C_4 is RNB and only has cycles of length 4. It is neither pancyclic nor cycle extendable. The cycle extendable graphs are obviously a subset of pancyclic graphs. Clearly, a complete graph is RNB, pancyclic and cycle extendable. And thus we conclude that these three ovals overlap.

It is a very hard problem to distinguish the intersections between RNB graphs and these advanced Hamiltonian concepts. This problem is beyond the scope of this paper and is not plotted in the Venn Diagram.

It is interesting to look at the 5-vertex star graph. It is an RNB graph. It is an acyclic graph with the Hamiltonian property. The simple graph itself does not have any cycles. However, if we consider an undirected edge as two directed edges each in one direction, the simple graph becomes a digraph and has a lot of directed cycles. This illustrates that in this sense it is easier to find RNB cycles is easier than to find Hamiltonian cycles. Thus, to this extent, the RNB property is less demanding. It explains that, star graphs, despite $\kappa_e = 1$, are RNB. This tricky point makes the graph theory approaches to the RNB property more difficult since most graph theory studies are focused on simple graphs instead of digraphs.

This topic is difficult yet worth studying in the future.

4.0 MAIN RESULTS OF NON-BLOCKING GRAPHS

This chapter introduces the main result of non-blocking network structures and its online routing algorithms. We continue to use Definition 2.4.8, 2.4.9, 2.4.10 to depict non-blocking properties in mesh network. Although the nature of non-blocking is consistent over both switching structures and mesh networks, it is worth noting that mesh networks don't have middle stage switching facilities in contrast to the Clos structure. In addition, unlike switching structures, links in mesh networks work in both directions. These features will lead to a different approach to prove non-blocking properties.

This chapter contains two theorems for RNB and WSNB properties respectively, along with two corresponding routing algorithms for each theorem respectively. We begin with an observation of a simple RNB graph. The structural features and routing patterns of the case are then discussed and extended without loss of generality, which later become the non-blocking theorems and non-blocking algorithms in the main result. The optimality of non-blocking graphs in the theorems and implementation issues for the algorithm are discussed.

4.1 AN EXAMPLE RNB GRAPH

According to Remark 2.4.11, we can prove the RNB property by enumerating routing path solutions for every traffic pattern. Here we investigate the RNB property of C_4 by enumerating routing paths of its traffic patterns. From its routing paths we summarize a general routing rule which can be used to generate routing paths for all traffic patterns in this graph. A possible set of routing paths for the traffic patterns are recorded in Table 2.

Table 2 records each pattern in the form of permutations in the left column. For example, the #2 traffic pattern is (B) (D) (A C) which represents that at this instant A is sending traffic to C and C is sending traffic to A.

In the right column, the routing paths for every traffic pattern are recorded. A routing path is represented by a series of vertices connected by "$\rightarrow$". The source and destination vertices in the traffic patterns are bold. For example, in the path for traffic pattern # 2, vertices A and C are sending/receiving traffic while vertex B is not actively sending or receiving any traffic but just forwarding the traffic between A and C. Thus A and C are bold but B is not.

Note that a vertex may be a source/destination for a traffic stream and at the same time forward traffic from other vertices. For example, in traffic pattern # 20, the connection vectors in the traffic pattern are { (A,C), (C,D), (D,B), (B,A)}. We can observe that vertex B is the destination of connection (D, B), the source of (B, A), and simulteneously forwards connection (A, C).

Based on Definition 2.4.10, Remark 2.4.11 and the solution shown in Table 2, we have:

Lemma 4.1.1. *The simple graph C_4 is rearrangeably non-blocking.*

We can observe an interesting two-case routing rule based on the routing paths in Table 2. For a connection denoted by a pair of source/destination vertices, if the source vertex and destination vertex are adjacent, the connection is routed directly to the link between them; otherwise, the traffic can be routed to a length-2-path through an intermediate vertex in the other party. In the next section, we extend this routing rule to a general routing algorithm, which servers as a cornerstone to the non-blocking theorems in the main result.

4.2 THE MAIN RESULT

The main result contains two original theorems. The first theorem proves the WSNB property for bipartite complete graphs and proposes a WSNB routing algorithm while the second

Table 2: Possible routing solutions for C_4

Pattern #	Permutation	Possible Routing Path (s)
1	(A) (B) (C) (D)	N/A
2	(B) (D) (A C)	$\mathbf{A} \rightarrow B \rightarrow \mathbf{C} \rightarrow B \rightarrow \mathbf{A}$
3	(B) (C) (A D)	$\mathbf{A} \rightarrow \mathbf{D} \rightarrow \mathbf{A}$
4	(D) (C) (A B)	$\mathbf{A} \rightarrow \mathbf{B} \rightarrow \mathbf{A}$
5	(A) (C) (B D)	$\mathbf{B} \rightarrow A \rightarrow \mathbf{D} \rightarrow C \rightarrow \mathbf{B}$
6	(A) (B) (D C)	$\mathbf{D} \rightarrow \mathbf{C} \rightarrow \mathbf{D}$
7	(A) (D) (B C)	$\mathbf{B} \rightarrow \mathbf{C} \rightarrow \mathbf{B}$
8	(B) (A C D)	$\mathbf{A} \rightarrow B \rightarrow \mathbf{C} \rightarrow \mathbf{D} \rightarrow \mathbf{A}$
9	(D) (A C B)	$\mathbf{A} \rightarrow B \rightarrow \mathbf{C} \rightarrow \mathbf{B} \rightarrow \mathbf{A}$
10	(B) (A D C)	$\mathbf{A} \rightarrow \mathbf{D} \rightarrow \mathbf{C} \rightarrow B \rightarrow \mathbf{A}$
11	(C) (A D B)	$\mathbf{A} \rightarrow \mathbf{D} \rightarrow A \rightarrow \mathbf{B} \rightarrow \mathbf{A}$
12	(C) (A B D)	$\mathbf{A} \rightarrow \mathbf{B} \rightarrow A \rightarrow \mathbf{D} \rightarrow \mathbf{A}$
13	(D) (A B C)	$\mathbf{A} \rightarrow \mathbf{B} \rightarrow \mathbf{C} \rightarrow B \rightarrow \mathbf{A}$
14	(A) (B D C)	$\mathbf{B} \rightarrow A \rightarrow \mathbf{D} \rightarrow \mathbf{C} \rightarrow \mathbf{B}$
15	(A) (B C D)	$\mathbf{B} \rightarrow \mathbf{C} \rightarrow \mathbf{D} \rightarrow A \rightarrow \mathbf{B}$
16	(A C) (B D)	$\mathbf{A} \rightarrow B \rightarrow \mathbf{C} \rightarrow D \rightarrow \mathbf{A}$ $\mathbf{B} \rightarrow A \rightarrow \mathbf{D} \rightarrow C \rightarrow \mathbf{B}$
17	(A D) (B C)	$\mathbf{A} \rightarrow \mathbf{D} \rightarrow \mathbf{A}$ $\mathbf{B} \rightarrow \mathbf{C} \rightarrow \mathbf{B}$
18	(A B) (D C)	$\mathbf{A} \rightarrow \mathbf{B} \rightarrow \mathbf{A}$ $\mathbf{D} \rightarrow \mathbf{C} \rightarrow \mathbf{D}$
19	(A C B D)	$\mathbf{A} \rightarrow B \rightarrow \mathbf{C} \rightarrow \mathbf{B} \rightarrow A \rightarrow \mathbf{D} \rightarrow \mathbf{A}$
20	(A C D B)	$\mathbf{A} \rightarrow B \rightarrow \mathbf{C} \rightarrow \mathbf{D} \rightarrow C \rightarrow \mathbf{B} \rightarrow \mathbf{A}$
21	(A D B C)	$\mathbf{A} \rightarrow \mathbf{D} \rightarrow A \rightarrow \mathbf{B} \rightarrow \mathbf{C} \rightarrow B \rightarrow \mathbf{A}$
22	(A D C B)	$\mathbf{A} \rightarrow \mathbf{D} \rightarrow \mathbf{C} \rightarrow \mathbf{B} \rightarrow \mathbf{A}$
23	(A B D C)	$\mathbf{A} \rightarrow \mathbf{B} \rightarrow A \rightarrow \mathbf{D} \rightarrow \mathbf{C} \rightarrow D \rightarrow \mathbf{A}$
24	(A B C D)	$\mathbf{A} \rightarrow \mathbf{B} \rightarrow \mathbf{C} \rightarrow \mathbf{D} \rightarrow \mathbf{A}$

theorem proves the RNB property along with an RNB routing algorithm which contains rearrangement methods. Interestingly, the two non-blocking routing algorithms are based on the same general routing method.

We begin with the general routing algorithm and describe a need rearrangement issue raised when this algorithm is applied to a bipartite complete graph. Next, we investigate this issue and provide two approaches to solve this issue. One approach leads to the WSNB property and the other approach leads to the RNB property.

The core of the main result is the in-depth analysis of the relationship between the graph structure and the non-blocking properties, which reflects the capability of a network to handle dynamic traffic. As discussed in Remark 2.4.11, we know it is astronomically complicated to analyze dynamic traffic routing for non-blocking properties. However, we found the complexity could be reduced because of the graph structural advantages of a bipartite complete graph. Taking this advantage, we are able to study the routing process for dynamic traffic without loss of generality by math reasoning, which, in later sections, turns out to be the main result.

We introduce several definitions to address non-blocking routing in bipartite complete graphs.

Definition 4.2.1. *A **bipartite graph** G is a graph whose vertices can be divided into two **party sets** U and V such that every edge of G has one vertex in U and the other in V. The vertices sets U and V are called **parties**.*

Definition 4.2.2. *A **complete bipartite graph** is a bipartite graph $G = (U, V, E)$ such that for any $v_1 \in U$ and $v_2 \in V$, v_1 and v_2 are adjacent. A complete bipartite graph with $|U| = r$ and $|V| = t$ is denoted as $K_{r,t}$.*

Definition 4.2.3. *A connection vector $\overrightarrow{x} = (s, d)$ in a traffic pattern in a bipartite graph is called an **intra party connection** if both $s, d \in U$ or both $s, d \in V$. Otherwise, it is an **inter party connection**, e.g. $s \in U$ and $d \in V$.*

Note that a traffic pattern in the form of a permutation can be written as a set of connection vectors equivalently. Thus the routing solution of a traffic pattern is essentially a set of paths for each corresponding connection. The paths in the solution for a traffic pattern

should not overlap if the solution is valid for a simple graph. For the details of transforming a permutation to a set of connections and its related discussions, please refer to Appendix A.2.

Based on the definitions of non-blocking properties, network and traffic in Section 2.4, we can proceed to introduce the main result in the following sections. Section 4.2.1 introduces the general routing algorithm. Section 4.2.2 discusses the routing issue in combination with the graph structure advantage of bipartite complete graphs. Section 4.2.3 discusses the WSNB property. Section 4.2.4 discusses the RNB property. Section 4.2.5 discusses the optimality of the result.

4.2.1 The General Routing Algorithm

We propose a general routing algorithm as Algorithm 1, which is summarized from the non-blocking case in Section 4.1. Algorithm 1 routes a connection by its connection type: inter party connections are routed to the link directly connecting the source and destination, whereas intra party connections are routed to a length-2 path through a vertex from the other party. We call the midway vertex a **hop** vertex.

The hop vertex of an intra party connection is selected by function *Findhop*, which is implemented to select a hop vertex randomly from available hops at the instant it is called. A hop vertex v is unavailable for connection (s,d) if either link $s \to v$ or $v \to d$ is used.

However, simply applying Algorithm 1 to $K_{r,t}$ without a backup rearrangement method can not promise any non-blocking properties. We offer Example 4.2.4 to study one typical blocking case in $K_{2,2}(C_4)$. In this example, let Table 2 be the routing table for the traffic patterns. As can be observed, the paths in the table could be a valid result of the algorithm if we treat each traffic pattern independently without considering the arrival sequence of connections. Nevertheless, we can spot that, for some connections in a traffic pattern, if they arrive at a particular sequence, we need rearrangements to route the connections as the routing table suggests.

Data: $K_{r,t}$ with party sets U and V where $|U| = r$ and $|V| = t$, a traffic pattern denoted as a permutation π as in Table 2.

Result: Routing paths P for each traffic pattern

Initialization: for all $v \in V(K_{r,t})$ put $hops(v) = 0$;

Initialization for Findhop: Put $S = \emptyset$, $S \subset V(K_{r,t})$;

for *each source destination pair* (s, d) *in traffic pattern* π **do**

  if $s, d \in V$ *or* $s, d \in U$ **then**

    v=$Findhop(s, d, K_{r,t}, S)$;

    Add link $s \to v$ and $v \to d$ for connection vector (s,d) to P;

    Mark the link $s \to v$ and $s \to d$ as used;

  else

    Add link $s \to d$ to the entry of connection (s, d) in path P;

    Mark the link $s \to d$ mentioned above used;

  end

end

Function $Findhop(s, d, K_{r,t}, S)$;

for *each* v *in the other party than* s *and* d **do**

  if *both links* $s \to v$ *and* $v \to d$ *are both unused* **then**

    Add v to S;

  end

end

Return arbitrary $v \in S$;

Algorithm 1: General Routing Algorithm

Example 4.2.4. *Considering graph* $K_{2,2}$ *and routing table as Table 2 as the routing table, we can observe a blocking scenario. Let connections* (A, C) *and* (C, A) *arrive first at an idle network and the traffic pattern of the network at this instant is #2. The two connections are routed according to the routing table, both of which are hopped by* B*. Next, let connections* (B, D) *and* (D, B) *arrive later at an instant before either* (A, C) *and* (C, A) *leaves. The traffic pattern at this instant is #16.*

By comparing the routing paths of traffic patterns #2 and #16 in Table 2 in C_4*, we find*

that connections (B, D) and (D, B) essentially has no path to route because the routing paths of (A, C) and (C, A) in pattern #2 of the routing table have used all links adjacent to vertex B. Connections (B, D) and (D, B) must be blocked without rearrangements.

In order to successfully transit from pattern #2 to #16, we need to rearrange the routing path for connection (C, A) from $C \to B \to A$ to $C \to D \to A$. After this rearrangement, the paths for both new connections can be routed as shown in pattern # 16 in Table 2.

We can deduce that Example 4.2.4 could be extended to arbitrary $K_{r,t}$ as long as $r, t \geq 2$, which forfeits non-blocking properties for Algorithm 1 in $K_{r,t}$. In the following contents of this paper, we use the term "need rearrangement" to refer this scenario.

If there exists a rearrangement method which can be proven to solve all the need rearrangement issues, Algorithm 1 along with the rearrangement method would make $K_{r,t}$ RNB. If a generalized hop selection preference can be proven to preclude all the need rearrangement scenarios, a modified the algorithm equipped with this hop selection preference, would make $K_{r,t}$ WSNB. To begin either branch of study, we need to first study the need rearrangement scenarios in a general scope for all $K_{r,t}$. These contents are covered in Section 4.2.2.

4.2.2 Rearrangements for Algorithm 1

As pointed out by Example 4.2.4, concerns for blocking and rearrangements may rise while applying Algorithm 1 to C_4, which is further "referred" as need rearrangement issues. This section extends this issue to all $K_{r,t}$ and studies the general phenomenon of this issue. During this study, we take great advantage from the graph structure of bipartite complete graphs because the traffic analysis is greatly simplified compared to a general connected graph.

We begin this topic with the investigation of the path overlaps between a group of connections in a traffic pattern. As shown in Example 4.2.4, the routing path of one connection may take the path(s) for another connection, just like the paths of (A, C) and (C, A) takes all paths for B to handle a new connection. To depict this phenomenon we define a term "interfere".

Next, we proceed to investigate the necessary and sufficient conditions of the need rearrangement cases and we propose a "traffic characteristic set" concept to generally depict

the features for need rearrangement issues. It describes the network status that needs rearrangement, which is a cornerstone for advanced studies of WSNB and RNB properties in Section 4.2.3 and Section 4.2.4.

4.2.2.1 Analysis of Routing Paths This section introduces a term "interfere" to analyze the routing paths. It depicts the scenario that routing one connection to a path reduces the number of paths for another connection.

Definition 4.2.5. *Given a routing algorithm, the* ***candidate paths*** *for a connection x_1 are contained in a set P_1, the elements of which are the routing paths for x_1 computed by the routing algorithm when the network is idle.*

For example, an inter party connection (u, v) has only one candidate path, which is $u \to v$. The candidate paths for an intra party connection (v_1, v_2) is a set of of paths $\{v_1 \to u \to v_2 | u \in U\}$. Its cardinality is equal to the cardinality of U.

The candidate path set P of a connection represents the maximum power of the routing algorithm to route the connection in the graph. A candidate path is not available to route this connection if part of it is used to route other connections. In an extreme scenario every path of a connection may be used by other connections and consequently it has no path to choose, which is essentially the need rearrangement scenarios. Consider two connections x_1 and x_2 and their candidate paths P_1 and P_2.

Definition 4.2.6. *Two connections x_1, x_2 in a traffic pattern X are said to* ***interfere*** *with each other if there exists one path p_1 in P_1 and another path p_2 in P_2 such that they are not link-disjoint*[1]*.*

Definition 4.2.7. *A connection x* ***needs rearrangement*** *if and only if for each of its candidate path $p \in P$, p can not be used because there exists a path p^* of another deployed connection x^* such that p^* and p are not link-disjoint.*

Remark 4.2.8. *Connection x_1 interferes with connection x_2 means that if the connection x_1 arrives first and takes path p_1, connection x_2, if it arrives later, can't take path p_2 and vice versa. In another words, the path decision of one connection may reduce the number of*

[1] In this scenario link (v_1, v_2) and link (v_2, v_1) are disjoint.

available paths of another connection.

The need rearrangement scenario defined in Definition 4.2.7 is an extreme case of interfering connections in which every candidate path of a connection is taken by other connections so that it needs rearrangement. This scenario is very dangerous because the connection will be blocked if no rearrangement method exists.

Based on the definitions above, we are equipped to investigate the need rearrangement issue by studying interfered connections while applying Algorithm 1 in $K_{r,t}$. We prove Lemma 4.2.9 as a property for routing inter party connections.

Lemma 4.2.9. *In graph $K_{r,t}$, an inter party connection (u_1, v_1) can always be routed according to Algorithm 1 without interfering with any other connection in any traffic pattern.*

Proof of Lemma 4.2.9. Algorithm 1 routes an inter party connection (u_1, v_1) directly to $u_1 \to v_1$. It interferes with another connection if this link belongs to one of the candidate paths of the connection. We can prove it by contradiction, assuming one such connection exists in the traffic pattern. The proof process can be divided into three cases by the particular form of the interfering connection.

Case 1: The interfering connection is an inter party connection, in the form of (u_1, v_1). However, we have $\eta_d(X, v_1) = 2$ and $\eta_s(X, u_1) = 2$, which contradicts the property of an applicable traffic pattern in Proposition 2.4.6.

Case 2: The interfering connection is an intra party connection, in the form of (u_1, u), $u \in U$. Because it interferes with (u_1, v_1), we know its routing path contains $u_1 \to v_1$. We know it should be hopped by v_1. However, we have $\eta_s(X, u_1) = 2$, which is, similarly, against Proposition 2.4.6.

Case 3: The interfering connection is an intra party connection, in the form of (v, v_1), $v \in V$. It is a dual case to Case 2 and we observe its contradiction to Proposition 2.4.6 as $\eta_d(X, v_1) = 2$.

To summarize, each case is proven by contradiction and we can deduce that there is no such connection in any traffic pattern that would interfere with (u_1, v_1). Thus the proof is complete. □

Lemma 4.2.9 suggests that inter party connections can always be routed by Algorithm 1

in $K_{r,t}$, no matter the traffic pattern and arrival sequence.

Lemma 4.2.10. *An intra party connection (u_1, u_2) can be routed according to Algorithm 1 without interfering with another intra party connection (u_3, u_4), if u_1, u_3 belongs to the same party.*

Proof. We begin with a straightforward fact that u_1, u_2, u_3, u_4 belong to the same party U[2]. We have $u_1 \neq u_3$ and $u_2 \neq u_4$ from Proposition 2.4.6. The hop vertices for the connections are selected from V. We prove that they do not interfere with each other by showing that the candidate routing paths of the two connections by Algorithm 1 are mutually link-disjoint. Without loss of generality, assume (u_1, u_2) selects v_1 and its routing path is $u_1 \to v_1 \to u_2$. Then we assume that (u_3, u_4) can be routed by a hop v_2 and its routing path is $u_3 \to v_2 \to u_4$. As mentioned, we have $u_1 \neq u_3$ and $u_2 \neq u_4$. We can deduce that link $(u_1, v_1) \neq$ link (u_3, v_2), and similarly link $(v_1, u_2) \neq$ link (v_2, u_4), for each combination of v_1 and v_2. Therefore, the paths are mutual link-disjoint and we conclude that the proof is complete. □

Lemma 4.2.9 states that inter party connections do not interference with any connection. Lemma 4.2.10 states that intra party connections from the same party do not interfere with each other. Thus the only possibility for rearrangements originates from intra party connections from different parties. This fact can also be supported by Example 4.2.4. To depict it formally, we have:

Lemma 4.2.11. *Suppose a graph $K_{r,t}$ is routed by Algorithm 1. If connection x_1 interferes connection x_2:*

- *x_1 and x_2 are intra party connections, and*
- *The source vertices of x_1 and x_2 are from different parties.*

Based on Lemma 4.2.11, we immediately have a preliminary result for a WSNB graphs:

Lemma 4.2.12. *Under Algorithm 1, $K_{1,t}$(and isomorphically, $K_{t,1}$) is WSNB, and also RNB.*

Proof. In $K_{1,t}$ there is only one party of vertices that could be involved in intra party connections and hence we can deduce that two connections, if they exist in the $K_{1,t}$ network at the

[2]It is a dual case if all belong to V and thus omitted.

same instant (they are part of an applicable traffic pattern), do not interfere. Consequently, no rearrangements are needed. We can conclude that $K_{1,t}$ is WSNB by Algorithm 1.
The proof is complete. □

Based on Lemma 4.2.12, we set the convention that $r, t \geq 2$ in subsequent contents when we study the rearrangement issue and non-blocking properties unless otherwise specified.

The number of combinations of the routing paths of deployed traffc to investigate need rearrangement scenarios is reduced by Lemma 4.2.9, Lemma 4.2.10 and Lemma 4.2.11, from all connections, to intra party connections from different parties.

In addition, the enumeration of arrival sequences for the intra party connections from the same party is reduced because Lemma 4.2.10 points out that they don't interfere. That is, no matter the arrival sequence and the path selection, no candidate path for any connection is unavailable due to another intra connection from the same party. Thus, to observe the extent of interference of an intra party connection, we only need to check the deployed intra party connections from the other party.

Remark 4.2.13. *The above advantages, which simplify the traffic analysis, do not apply to a general graph and general routing algorithms. Without the graph structure advantage and the party of vertices, the relationship about how and when the connections interfere becomes more opaque.*

In the next section, we proceed to investigate the cause to need rearrangement scenarios by analyzing the arrival sequence, combination, and routing paths of intra party connections from different parties in a traffic pattern.

4.2.2.2 Analysis of Need Rearrangement Scenarios

In this section we study when and why "needs rearrangement" scenarios will happen in $K_{r,t}$. We have reached the point that only intra party connections from different parties could interfere each other. While "need rearrangements" is an extreme result of interfered connections, we investigate how their interference evolves into need rearrangement scenarios.

Consider a traffic pattern X for $K_{r,t}$ containing an intra party connection (v_1, v_2) along with $|U|$ intra party connections from U. Without loss of generality, let connection (v_1, v_2)

be the first intra party[3] connection from V and suppose all the intra party connections from U (if there is any) have been routed by Algorithm 1. The generality is implied in the fact that the hops for the intra party connections from U can be arbitrarily selected because, according to Lemma 4.2.10, they are not interfered at the instants they arrive.

We demonstrate the fact that the intra party connections from U may interfere with connection (v_1, v_2) and may render that it needs rearrangement. After that, we continue to investigate how they will render that it needs rearrangement in general.

Connection (v_1, v_2) has at most $|U|$ hop candidates which are essentially all the vertices in U. The routing path in the form of $v_1 \to u \to v_2$ is a candidate path, where vertex u is the hop vertex. There are at maximum $|U|$ candidate paths.

Notice that the candidate paths for any intra party connection (v_1, v_2) can be denoted by its hop vertex. Thus, we propose Definition 4.2.14 to define the concept of "unavailable", especially in applying Algorithm 1 to $K_{r,t}$, for a hop vertex of a connection. This will be used to depict that a candidate path can not be used due to another connection.

Definition 4.2.14. *A candidate hop vertex u for connection (v_1, v_2) is **unavailable**, or **taken**, if at least one of the following statements are satisfied:*

1. *Vertex u is involved in a connection (u_1, u) which is hopped by v_1 (routing path: $u_1 \to v_1 \to u$).*
2. *Vertex u is part of a connection (u, u_2) which is hopped by v_2 (routing path: $u \to v_2 \to u_2$).*

Propositions 4.2.15 and 4.2.16 are two immediate results based on Definitions 4.2.14 and 4.2.7. Their proofs are omitted.

Proposition 4.2.15. *A connection (v_1, v_2) needs rearrangement if and only if every hop vertex u is unavailable.*

Proposition 4.2.16. *If connection (u_1, u_2) interferes with (v_1, v_2), connection (u_1, u_2) must be hopped by either v_1 or v_2.*

[3]If it is NOT the first, and there exists routed intra party connections from V, they may interfere the paths for the intra party connections from U that are coming after them. This reduces the path selection for the connections from U, which is no longer "arbitrary " as mentioned and the generality is lost.

Definition 4.2.14 can be used to study how intra party connections from different parties interfere by the selection of hop vertices. Proposition 4.2.15 indicates that a need rearrangement scenario is reached when all hops are unavailable.

Consider connection (v_1, v_2) and its hop candidate vertex set U. We have Proposition 4.2.17 by summarizing Definition 4.2.14.

Proposition 4.2.17. *One intra party connection (u_1, u_2) at most renders one hop vertex u unavailable for (v_1, v_2), and $u \in \{u_1, u_2\}$.*

Proof of Proposition 4.2.17. Proof by contradiction. According to Definition 4.2.14, connection (u_1, u_2) can only render u_1 or u_2 unavailable for connection (v_1, v_2). Assume both hops u_1 and u_2 are unavailable to (v_1, v_2) due to (u_1, u_2).

In order to render u_1 unavailable, connection (u_1, u_2) must be hopped by v_2 whereas in order to render u_2 unavailable, it must be hopped by v_1. Its routing path can't be simultaneously hopped by two different vertices and hence is contradictory. The proof is complete. □

Based on Propositions 4.2.15 and 4.2.17, we have:

Proposition 4.2.18. *If connection (v_1, v_2) needs rearrangements, there exist $|U|$ deployed intra party connections from U.*

Proof of Proposition 4.2.18: We begin this proof by Proposition 4.2.17, which states that an intra party connection from U can at most render one vertex u interfered for connection (v_1, v_2). As definition 4.2.14 requires all $|U|$ vertices must be interfered, we need at least $|U|$ intra party connections to render $|U|$ vertices interfered. Considering the fact that we can at most have $|U|$ intra party connections from U (implied by Proposition 2.4.6), we know there must be $|U|$ deployed intra party connections. At this point, we have reached Proposition 4.2.18. The proof is complete. □

Based on Proposition 4.2.18, we can deduce an immediate result.

Lemma 4.2.19. *If a traffic pattern contains an inter party connection, it can be routed by Algorithm 1 without rearrangements no matter the arrival sequence of the connections.*

Proof. If traffic pattern X has an inter party connection, then we know there are at most $|U|-1$ or $|V|-1$ intra party connections from either party. Therefore, the premise of Proposition 4.2.18 can't be satisfied. Thus we can deduce and conclude that any connection in this traffic pattern can be routed no matter the arrival sequence and hop selection. The proof is complete. □

Combining Proposition 4.2.15, Proposition 4.2.18, and Definition 4.2.14, we found that we need $|U|$ intra party connections deployed so as to make a connection (v_1, v_2) need rearrangements. We can also imply from Proposition 4.2.16 that these $|U|$ intra party connections must be hopped by either v_1 or v_2. Based on this, we have a better grasp of the network status when rearrangement is needed.

In addition, in order to satisfy Definition 4.2.14, we need to confine the $|U|$ intra party connections so that each of them renders a distinct hop vertex u interfered. Otherwise, it might be possible that there are two connections that render the same hop unavailable, and hence there exists an available hop. Based on this idea, Lemma 4.2.20 provides a generalized counter example in which there might exist $|U|$ interfering connections but no rearrangement.

Lemma 4.2.20. *Given an intra party connection (v_1, v_2), if there exists a vertex u such that both statements in Definition 4.2.14 are satisfied simultaneously, connection (v_1, v_2) has an available hop and thus does NOT need rearrangements.*

Proof. Let u satisfy the two statements in Definition 4.2.14. We know it is involved into two intra party connections denoted as (u, u_1) and (u_2, u). According to the statements we know that (u, u_1) is hopped by v_2 and (u_2, u) is hopped by v_1. Thus, there are $|U|-1$ other vertices and at maximum $|U|-2$ other intra party connections from $|U|$. According to Proposition 4.2.17, one connection can at most render one vertex unavailable. Even if all of the $|U|-2$ connections interfere with (v_1, v_2) (that is, there are $|U|$ interfering connections in total), there must exist one available hop, because at maximum $|U|-1$ hops are unavailable. The proof is complete. □

From Definition 4.2.14 to Lemma 4.2.20, we investigated the routing paths of the intra party connections from U, which cause (v_1, v_2) to need rearrangement. At this point, we

have reached several general facts about the need rearrangement scenarios:

1. We need $|U|$ intra party connections that are routed prior to (v_1, v_2) (Proposition 4.2.18, which summarizes Definition 4.2.14, Proposition 4.2.15 and 4.2.17).
2. Each intra party connection should be hopped by either v_1 or v_2 (Proposition 4.2.16).
3. Each intra party connection should render a distinct hop vertex unavailable. In another words, one-to-one rule (Summarized from Proposition 4.2.18 and Lemma 4.2.20).

Based on the above summary, we propose Proposition 4.2.21, which describes a general property of the routing paths of the $|U|$ intra party connections from U when they could cause connection (v_1, v_2) to need rearrangements.

Proposition 4.2.21. *Consider an intra party connection (v_1, v_2). If all its candidate hops u are unavailable, we have, for each $u \in U$, its involved connections $\{(u_1, u), (u, u_2)\}$ are hopped by the same vertex.*

To better depict the relationship of (u_1, u), and (u, u_2), we propose Definition 4.2.22.

Definition 4.2.22. *Two connections (v_1, v_2) and (v_3, v_4) in a traffic pattern are **adjacent** if $v_1 = v_3$ or $v_2 = v_4$.*

Proof of Proposition 4.2.21. It can be implicitly inferred from Proposition 4.2.18 that u is involved into two intra party connections in the form of $\{(u_1, u), (u, u_2)\}$. Because (u_1, u) interferes (v_1, v_2), according to Definition 4.2.14, connection (u_1, u) may:

1. render u unavailable if hopped by v_1.
2. render u_1 unavailable if hopped by v_2.

Similarly, connection (u, u_2) may:

1. render u_2 unavailable if hopped by v_1.
2. render u unavailable if hopped by v_2.

As implied by the counter example in Lemma 4.2.20, we have to let both (u_1, u) and (u, u_2) each render a distinct hop unavailable in order to make (v_1, v_2) need rearrangements. Thus we only have the following combinations:

1. (u_1, u) renders u unavailable by hop v_1 and (u, u_2) renders u_2 unavailable by hop v_1.

   OR:

2. (u_1, u) renders u_1 unavailable by hop v_2 and (u, u_2) renders u unavailable by hop v_2.

It might appear to be a valid combination that (u_1, u) renders u_1 unavailable by hop v_2 while (u, u_2) renders u_2 unavailable by hop v_1. However it contradicts Definition 4.2.14, which implies that u can only be taken by an intra party connection involving it; whereas, in this scenario all the connections involving u do not render u unavailable. Consequently u becomes an available hop and connection (v_1, v_2) does not need rearrangement. Thus this combination is not valid.

At this point, we can observe that either valid combination satisfies the statement. The proof is complete. □

Proposition 4.2.21 describes a rule for the routing paths of the intra party connections from U in the need rearrangement scenarios we proposed in Definition 4.2.23 and 4.2.25. We could proceed to study the relationship between the routing paths of the connections from U and the need rearrangement scenarios to obtain in-depth knowledge about the routing algorithm in order to reach non-blocking properties.

Since Proposition 4.2.21 provides the rule for routing the paths of two adjacent connections, it implies transitivity. We can extend this knowledge to routing the paths for all connections from U.

For example, knowing that connection (v_1, v_2) needs rearrangement and connections (u_1, u) and (u, u_2) are routed to the same hop could confirm connection (u_2, u_3), if in the same traffic pattern, is also routed to the same hop. Thus the hop selection could be propagated by adjacent connections, and their routing paths can be obtained as we learn their hop selection. To better study this transitive process in different traffic patterns, we need to know the group of adjacent connections in a traffic pattern as a whole. Thus we propose Definition 4.2.23.

Definition 4.2.23 (Traffic Cycle)**.** *Consider a traffic pattern X and its routing paths in $K_{r,s}$. A vertex set $M = \{v_1, v_2, ...\}$ is a* ***traffic cycle*** *if:*

1. *For each $v \in M$, v is involved in two connections in X:(v, v_r) and (v_l, v).*

2. $v_r, v_l \in M$.

Specifically, if all the connections mentioned in case 1 of Definition 4.2.23 are intra party connections, we call M *an* ***intra party traffic cycle****.*

Recall that Lemma 4.2.19 reduces the scope to intra party traffic cycles in studying the need rearrangement scenarios. To discuss the routing paths of the group of interfering intra party connections, the term "traffic cycle" refers to an intra party traffic cycle in subsequent discussions unless otherwise specified.

From Definition 4.2.22, we have:

Proposition 4.2.24. *Two adjacent connections belong to one and only one traffic cycle (not necessarily intra party).*

Combining Definition 4.2.23 and Proposition 4.2.21, we know that the routing hop for all the connections in a traffic cycle is the same when rearrangement is necessary. That is, if we know the vertices in a traffic cycle and the routing hop, we know the routing paths of the connections without knowing specifically the connections that defined the traffic cycle.

We then propose Definition 4.2.25 to generalize this idea to the scope of a traffic pattern which might contain multiple traffic cycles.

Definition 4.2.25 (Traffic Characteristic Sets)**.** *A set of intra party traffic cycles is called a* ***traffic characteristic set*** $P_X(v^*)$ *if all the intra party connections in the set of traffic cycles is hopped by* v^**. The traffic characteristic set* $P_X(v^*)$ *may be abbreviated as* $P(v^*)$*.*

The traffic characteristic set is defined on a set of intra party traffic cycles, which essentially represents a portion of a traffic pattern as well as their routing paths. The reason to define it over a set of traffic cycles is to cover the case that all the intra party connections from one party may form multiple cycles while they still are hopped by one identical vertex.

Remark 4.2.26. *Note that requiring all the connections be hopped by the same vertex implied that all vertices in* P *belong to the same party. We can determine the routing paths of all connections in the set of cycles by the vertices in* P *and* v^* *without knowing each specific*

connection. It summarizes a set of traffic patterns in which the connections might be different while the links used by their routing paths as a whole are the same.

Summarizing Definition 4.2.25, Definition 4.2.14, and previous results, we have Lemma 4.2.27, which connects need rearrangement scenarios and the concept of traffic characteristic sets.

Lemma 4.2.27. *Connection* (v_1, v_2) *needs rearrangement* ***if and only if*** *the intra party connections from* U *and their routing paths can be depicted by two traffic characteristic sets* P_{v_1} *and* P_{v_2} *such that:*

1. $P_{v_1} \cap P_{v_2} = \emptyset$.
2. $P_{v_1} \cup P_{v_2} = U$.

Proof. We begin with the forward direction. That is, we know it needs rearrangement and deduce that the two statements hold. They can both be proven by contradiction. Denote vertex $u \in U$.

Contradiction for statement 1: Assume there exists vertex u such that $u \in P_{v_1}$ and $u \in P_{v_2}$, we can deduce that the connections involving v are hopped by both v_1 and v_2, which is contradictory. Thus statement 1 is proven.

Contradiction for statement 2: Assume there exists a vertex $u \in U$ that belongs to neither P_{v_1} nor P_{v_2}. There are three cases:

1. Vertex u may not be involved in any intra party connections;
2. Vertex u is involved in some intra party connections that are hopped by other vertices so that hop u is not taken;
3. Vertex u is involved in inter party connection (s).

Each case above can be shown contradictory to the fact that (v_1, v_2) needs rearrangement and thus statement 2 is proven by contradiction.

Summarizing the contradictions for each statement, we can conclude that the forward direction is proven. We proceed to prove the backward direction. We begin with the two

statements and deduce that it needs rearrangements. From the statements we can deduce that:

1. Every u is involved into two connections, either as source and destination, i.e. in the form of (u_1, u) and (u, u_2).
2. For the above connections, they are hopped by the same vertex: either v_1 or v_2.

We can further deduce the link either $u \leftrightarrow v_1$ or link $u \leftrightarrow v_2$ is used because:

1. If (u_1, u) and (u, u_2) are hopped by v_1, then link $u \leftrightarrow v_1$ is used in both directions.
2. If (u_1, u) and (u, u_2) are hopped by v_2, then link $u \leftrightarrow v_2$ is used in both directions.

We can deduce that u is unavailable according to Definition 4.2.6. Note that the statements apply to every u so we know every u is unavailable, which satisfies Definition 4.2.14. We can deduce that (v_1, v_2) need rearrangements. The backward direction is proven.

To summarize, the proof is complete. □

4.2.2.3 Summary In this section, we discussed the routing paths of Algorithm 1 in $K_{r,t}$ and investigated when rearrangements are needed. As a result we achieved the following facts.

1. Inter party connections can always be directly routed without interfering with, or being interfered by, other connections.
2. Intra party connections from the same party can be routed without interfering with or being interfered by, inter party connections or other intra party connections from the same party.
3. Intra party connections from one party may interfere with an intra party connection from the other party. The worst interference will require rearranging the deployed paths. We proved a necessary and sufficient condition to the need rearrangement scenarios in Lemma 4.2.27.

The results in this section provide a solid theoretic base to study and prove RNB and WSNB properties in the subsequent sections. We can achieve the WSNB property by introducing a modified routing algorithm which prevents the formation of traffic characteristic sets; and thus, the need rearrangement scenarios are precluded, while keeping the properties Algorithm

1 has. To achieve the RNB property, we can find a way to rearrange routing paths to make a routing path available as a solution for these scenarios. These two topics are discussed in detail in Section 4.2.3 and 4.2.4 respectively.

4.2.3 Wide Sense Non-blocking Property

The WSNB property is defined as in Definition 2.4.9. We have found that applying Algorithm 1 to $K_{r,t}$ may need rearrangements in some specific scenarios, and thus is not WSNB. In this section, we illustrate and prove that the WSNB property could be achieved if we properly modify Algorithm 1 to preclude the need rearrangement scenarios. This can be realized by using a modified hop selection method for intra party connections. We propose Theorem 4.2.28 as the WSNB result and Algorithm 2 as the modified algorithm.

Theorem 4.2.28. *Graph $K_{r,t}$ is WSNB if routed by Algorithm 2.*

Algorithm 2 is modified from Algorithm 1 to alternatively route adjacent connections to different hops when possible in an attempt to preclude the need rearrangement scenarios, as suggested by Lemma 4.2.20. A hop vertex set S_1 is introduced, which consists of the hops used by adjacent intra party connections. Set S_1 has at maximum two elements because there are at most two adjacent connections. It serves as a reference and helps the hop selection process to prevent the formation of traffic characteristic sets.

Note that Lemma 4.2.20 suggests that, as long as there exists one traffic cycle that has two hops, we can avoid rearrangements. However, Algorithm 2 is implemented to route every pair of adjacent intra party connections to different hops as long as there is a chance. We will show in the proof that it is essentially over qualifying Lemma 4.2.20 and guarantees non-blocking. We also discuss the implementation concerns for Lemma 4.2.20 and Algorithm 2 in Remark 4.2.29.

Data: $K_{r,t}$ with party sets U and V with $|U| = r$ and $|V| = t$, and a connection (s,d).

Result: Routing paths P for each traffic pattern

Routing section same as in Algorithm 1, with the exception of replacing "Findhop" with "Findhop2";

Modified Routing Algorithm Initialization: Put $S = \emptyset$ and $S_1 = \emptyset$;

if (s, d) *is an intra party connection* **then**
    Route it in link $s \to d$.
else
    for *each* v *in the other party than* s *and* d **do**
        if *Link* $s \to v$ *and* $v \to d$ *are both unused* **then**
            Add v to S;
            if *There exist deployed connections that are adjacent to* (s, d) **then**
                For each of them, add its hop vertex v_1 (if intra party) to S_1;
                Do nothing if it is an inter party connection;
            end
        end
    end
    v_h=Findhop2 (s, d, S, S_1);
    Route the connection to path $s \to v_h \to d$;
end

Function Findhop2 (s, d, S, S_1)

if *only 1 vertex in* S **then**
    Return the vertex;
else
    if $S - S_1 = \emptyset$ **then**
        Return arbitrary $v \in S_1$.;
    else
        Return arbitrary $v \in S - S_1$;
    end
end

Algorithm 2: WSNB Routing Algorithm

Remark 4.2.29. *It is difficult to implement the algorithm exactly in the way Lemma 4.2.20 suggests because we need to collect and keep prompt traffic status for all nodes in the network, which is difficult to implement. As mentioned, the acquisition, maintenance, and propagation of the states of every connection, as well as their routing paths, over the entire network are very complicated and yet sensitive to delay and errors. Hence, the performance will be degraded.*

To minimize the impact from delay and errors as well as to propose an efficient online algorithm, function "Findhop2" is deliberately implemented to guarantee that the adjacent intra party connection pairs are routed to different hops, which requires knowing far fewer states (the routing path of its adjacent connections) than that of the entire network. We will illustrate and prove that this implementation is equivalent to Lemma 4.2.20 in preventing need rearrangement scenarios in the proof of the main result.

In order to rigorously prove that Algorithm 2 makes $K_{r,t}$ WSNB, we need to prove that it does not need rearrangement for any connection at any instant. That is, any connection (as long as the current traffic pattern is applicable to the network) can be routed, and its path kept for arbitrary time, no matter how the deployed connections are routed. Based on 4.2.9, we can reduce the scope so that every intra party connection can always have at least one available hop at any instant, no matter how the deployed connections are routed. This is equivalent to saying that the need rearrangement scenario does not occur if Algorithm 2 is applied. Before the main proof, we review the routing mechanism of Algorithm 2.

When there is only one candidate hop vertex available ($|S| = 1$), it is selected. It represents the instant that we have no choice but to route this connection to the hop that might happen to be the same as in its adjacent connections. This operation appears to be liable to make a traffic cycle hopped by one vertex (traffic characteristic set), which might undermine the main result. We propose Lemma 4.2.30 to discuss this issue.

Lemma 4.2.30. *Consider Algorithm 2 and $K_{r,t}$, $r, t \geq 2$. If the hop of a connection is selected at the branch $|S| = 1$, the connection and its routing path does not form a non-null traffic characteristic set.*

Lemma 4.2.30 defines a scenario in which in an arbitrary sequence of connection ar-

rival/departures as long as the traffic pattern at any instant is applicable. The path selection process of a deployed connection at its arrival instant is hard to determine as the traffic pattern at that instant may be arbitrary. However, we can prove in general, that if the hop is selected by the $|S| = 1$ "tricky move", the connection itself along with its routing path, can't form a non-null traffic characteristic set.

Proof of Lemma 4.2.30. Proof by contradiction. Assume a non-null traffic characteristic set can be formed. Without loss of generality, suppose the **first** formed non-null traffic characteristic set is made up by a set of intra party connections form U and its hop vertex is v. Denote the last deployed connection in the traffic cycle as (u_1, u_2). We know that there exists connection(s) adjacent to (u_1, u_2), which are hopped by v, and hense we know links $v \to u_1$ and $u_2 \to v$ are used by them.

In addition, we know that (u_1, u_2) must only have one hop to choose, which is v. Otherwise, this connection, must be given a hop vertex other than v and the non-null traffic characteristic set can't be formed. We see that all other vertices $v^* \in V$ must be unavailable for (u_1, u_2), and hence there are at least $|V| - 1$ intra party connections from V that are deployed and interfere with (u_1, u_2). From Proposition 4.2.16, we confirm that the $|V| - 1$ interfering connections must be hopped by either u_1 or u_2.

Given these facts above, we divide our proof into two cases.

Case 1: v is involved in the $|V|-1$ interfering connections. It covers both cases in which there exists $|V|$ or $|V|-1$ deployed connections. If there are $|V|$ deployed connections, v must be involved in two of them while one of them must belong to the $|V| - 1$ interfering connections[4].

If the v involved interfering connection is in the form of (v, v^*), $v^* \in V - v$, v is the source and we know the hop of this connection must be either u_1 or u_2. If hopped by u_1, it uses the link $u_1 \to v$, which is supposed to be used by the adjacent connections of (u_1, u_2) since they are hopped by v. Otherwise if hopped by u_2, it uses the link $v_1 \to u_2$ and it contradicts the fact that (u_1, u_2) can be hopped by v_1. Thus a contradiction is found. There is a dual case

[4]We use $|V| - 1$ as the number of interfering connections due to we are only interested to study one interfering connection for each unavailable hop. To be more precise, there might be $|V|$ interfering connections and $|V| - 1$ interfered hops because two connections interfere the same vertex.

if the connection is in the form of (v^*, v). Thus we can summarize that Case 1 is proven.

Case 2: v is not involved in the $|V|-1$ interfering connections. We can deduce that the $|V|-1$ connections must be a collection of traffic cycles; otherwise v must be involved in one of them. Combining Proposition 4.2.16 and Proposition 4.2.21, we know that each of the traffic cycles must be hopped by the same vertex and thus they form traffic characteristic sets. This contradicts the prerequisite that the cycle containing (u_1, u_2) is the **first** traffic characteristic set to be formed. Thus a contradiction is found and Case 2 is proven.

To summarize, we have proven Lemma 4.2.30. □

Now, we are ready to prove Theorem 4.2.28.

Proof of Theorem 4.2.28. In order to prove this theorem, we need to show that need rearrangement cases never occur if we use Algorithm 2. We can reduce the scale of this problem to only intra party connections according to Proposition 4.2.16, Lemma 4.2.9 and Lemma 4.2.19. Thus, in the subsequent development of the proof we focus on traffic patterns containing only intra party connections. Because Lemma 4.2.12 has proven the case for $r = 1$ or $t = 1$, we only prove the case for when $r \geq 2$ and $t \geq 2$. The routing analysis can be divided into two categories according to the logic of the function "Findhop2". We begin with the branch when $|S| \geq 2$.

Case 1: $|S| \geq 2$. This case can be further divided into several subcases depending on the outcome of $S - S_1$:

Case 1.0: If $S - S_1 = \emptyset$ is satisfied, S_1 must have two elements. This means the two adjacent connections must have been routed by different hops. By arbitrarily selecting a vertex in S_1 for the current connection, we can guarantee that the traffic cycle containing this connection is routed by at least 2 hops.

Case 1.1: If $S - S_1 \neq \emptyset$, a new hop vertex in $S - S_1$ will be picked up and the traffic cycle containing this connection is routed by at least $|S_1| + 1$ hops. It should be greater than or equal to 2 unless $S_1 = \emptyset$.

If $S_1 = \emptyset$, no adjacent connections are present in the network. Any hop selection would not contribute a traffic characteristic set as subsequent arrived connections will be routed to

prevent it.

In either case above, we know no rearrangement is needed according to Lemma 4.2.20.

Case 2: $|S| = 1$**.**

If $|S_1| \geq 2$, using similar logic we know that this cycle is routed by least $|S_1|$ hops no matter the hops in S. Similarly, there is no need to rearrange and this subcase is proven.

If $|S_1| = 1$, and $S_1 \neq S$, we know that this connection must be routed by the vertex in S and this cycle is routed by at least $|S_1| + 1 = 2$ hops. Similarly, it is proven that there is no need to rearrange.

If $|S_1| = 1$, and $S_1 = S$, this case has been covered by Lemma 4.2.30, which shows that this connection and its routing path forms no non-null traffic characteristic sets.

Summarizing the above cases[5], we observe that at any instant, there are no non-null traffic characteristic sets in the network and thus the need rearrangement scenarios are precluded. That is, any connection (as long as the traffic pattern is applicable) can have at least one path found by Algorithm 2, which can be kept for an arbitrary time. Combining the fact that in each branch of the routing algorithm, a connection can be routed without the need to rearrange other paths, we reached the WSNB property and the proof is complete. □

At this stage, we have proven the WSNB property for Algorithm 2 and $K_{r,t}$ in Theorem 4.2.28. It is worth mentioning that, if we follow the Benes Conjecture, to route all connections to the busiest part that can still handle them, all the intra party connections in a traffic pattern tend to be routed to the same vertex, which will raise need rearrangement scenarios and turns out to forfeit the WSNB property.

Thus the WSNB property, along with the modified algorithm, are counter examples of the Benes Conjecture because the WSNB property (considered as an efficient property) can be achieved if we distribute the load throughout the network according to traffic states, rather than grooming them together to the busiest part.

[5]We omitted the case $S = \emptyset$, as it can not happen, which can be implied. If it happens, we know the current connection needs rearrangement and a traffic characteristic set is formed, which contradicts the statements in Case 1, Case 2 and Lemma 4.2.30

4.2.4 Rearrangeably Non-blocking Property

This section discusses the rearrangeably non-blocking property for $K_{r,t}$. With the result of the previous section, we can reach a straightforward result that $K_{r,t}$ is RNB by Algorithm 2 because of its WSNB property. In addition, we state that, Algorithm 1, as a less sophisticated algorithm, can also achieve RNB property if supported by proper rearrangement methods.

Theorem 4.2.31. *Applying Algorithm 1 in $K_{r,t}$ is RNB.*

Proof. To prove that applying Algorithm 1 in $K_{r,t}$ is RNB, we need to show that, all the needs rearrangement scenarios can be solved by certain rearrangements. We thus prove the theorem by demonstrating we can rearrange all the "needs rearrangement" cases.

Without loss of generality, suppose an instant (v_1, v_2) needs rearrangement. According to Lemma 4.2.27 we know that all the intra party connections from U are routed and they form two non-null traffic characteristic sets P_{v_1} and P_{v_2}. The proof can be divided into several cases depending on P_{v_1} and P_{v_2}.

Case 0: Suppose $P_{v_2} = \emptyset$ and thus $P_{v_1} = U$. Thus we can find connection (u_1, u_2) where $u_1, u_2 \in P_{v_1}$. Since $P_{v_2} = \emptyset$, connection (u_1, u_2) can be rearranged to hop through v_2, thus (v_1, v_2) can take the path $v_1 \to u_2 \to v_2$, as shown in Figure 10. We can observe that connection (v_1, v_2) can be hopped through u_2 after this rearrangement. The case $P_{v_2} = \emptyset$ is a dual case and omitted. Consequently, this case is proven.

The rectangle labeled as P_{v_1} represents the traffic characteristic set while blank areas are an abstraction of other unspecified vertices in P_{v_1}. The arrows represent the routing paths of the connections involving the vertices, including those abstractly denoted by blank areas. A two-way arrow means at both directions the link has been used. To avoid ambiguity, the routing path for (v_1, v_2) is not plotted. This style applies to subsequent figures in this proof.

Otherwise, we know both traffic characteristic sets are non-empty, and thus each has at least two vertices. Without loss of generality, suppose (u_1, u_2) is a deployed intra party connection from U and $u_1, u_2 \in P_{v_1}$. It is a dual case if they belong to P_{v_2} and thus it is omitted.

Case 1: Suppose $P_{v_1}, P_{v_2} \neq \emptyset$ and v_2 is not involved as a source to another deployed intra party connection. It can be deduced that the path $u_1 \to v_2 \to u_2$ is available to rearrange

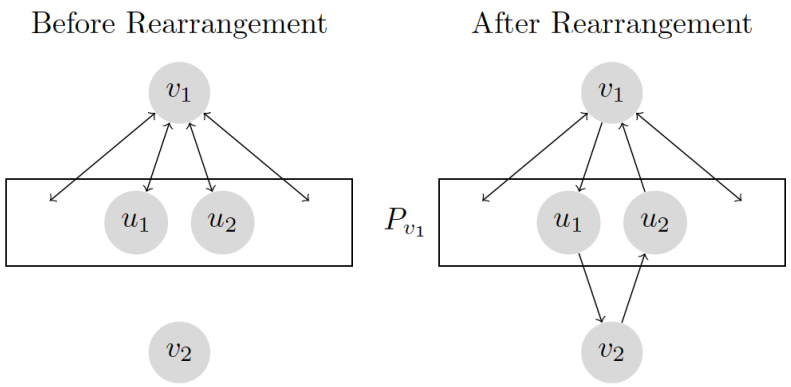

Figure 10: Rearrange process of Case 0

connection (u_1, u_2). After this rearrangement, intra party connection (v_1, v_2) can be hopped through u_2 to path $v_1 \to u_2 \to v_2$. This process in shown in Figure 11.

Case 2: Suppose $P_{v_1}, P_{v_2} \neq \emptyset$ and v_2 is involved into another connection that has been routed. Since (v_1, v_2) belongs to the traffic pattern, v_2 can only be the source of the connection. Note that this connection should be an intra party connection, which can be denoted as (v_2, v^*). We know $v^* \neq v_1$ because there is no path to route (v_2, v_1) given the paths of the connections from U.

Given $v^* \neq v_1$ and (v_2, v^*) is routed, we know it has a hop vertex, denoted as u^*. We can deduce $u^* \in P_{v_1}$. If $u^* \in P_{v_2}$, path $v_2 \to u^*$ is used by an intra party connection from U. However, from the fact that (v_2, v^*) is hopped through u^* we know that path $v_2 \to u^*$ is used to route this connection. A contradiction is spotted and hence we firmly deduce that $u^* \in P_{v_1}$. The routing path of this connection is plotted by dashed arrows to be distinguished from its counterpart of other connections from U.

We can find a vertex $u_1 \in P_{v_1}$, and its involved connection (u_1, u_2) such that $u_2 \neq u^*$ (there are at least two vertices in P_{v_1}). We can observe that it can be rearranged to hop through v_2. After that (v_1, v_2) can be hopped through u_2. Thus this case is proven.

Figure 12 plots the rearrangement process for Case 2 when $u_1 \neq u^*$. If $u_1 = u^*$, the

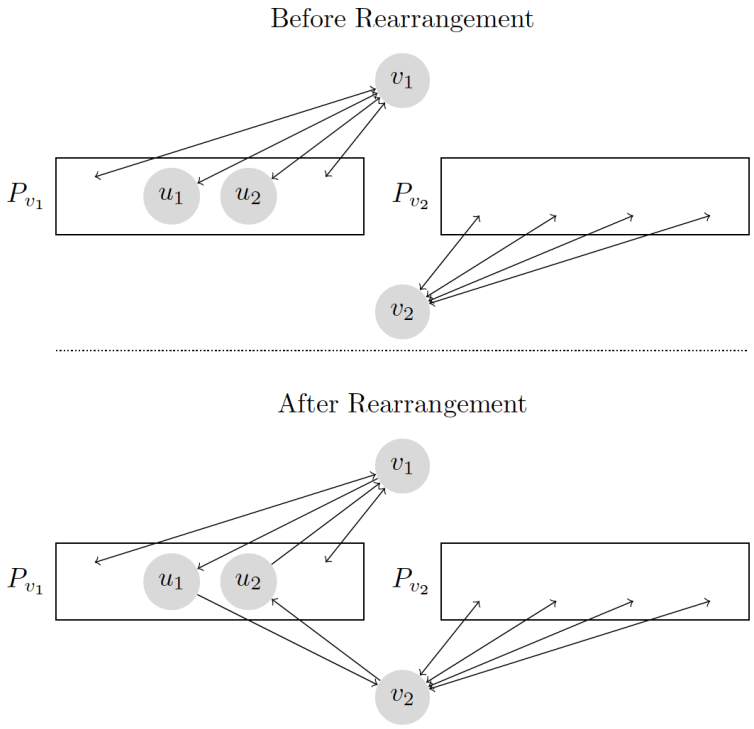

Figure 11: Rearrange process of Case 1

routing path $u_1 \to v_2 \to u_2$ is moved to $u^* \to v_2 \to u_2$ and the remaining parts are the same. The conclusion remains true that connection (v_1, v_2) can be hopped through u_2.

Summarizing the above cases, we have demonstrated a practical and general rearrangement plan for the every need rearrangement cases as defined in Lemma 4.2.27 in $K_{r,t}$. We can thus deduce applying Algorithm 1 in $K_{r,t}$ is RNB. □

4.2.5 Optimality of the Result

The main results in the previous sections suggest that bipartite complete graphs ($K_{r,t}$) are non-blocking. In this section, we discuss the optimality of $K_{r,t}$. We prefer non-blocking graphs with fewer links due to the significant cost of deploying a link. We evaluate the optimality by the number of edges. Specifically, we focus on the RNB property because it is computable for small scale graphs.

It is very difficult to evaluate the optimality of a non-blocking graph comprehensively as either the optimality or the non-blocking property is closely associated to both the graph structure and the routing algorithm. The impact of graph structure on the RNB property is obvious as shown in the lattice in Figure 5. The impact of routing algorithms can be implied from the two theorems in the main results, where different routing algorithms contributed to different non-blocking properties for the same set of graphs.

Instead of merging the factors, we can evaluate the optimality by discussing the two factors separately. Particularly We evaluate different RNB graphs while fixing the routing algorithm as Algorithm 1. Next, we evaluate the effect of routing algorithms on a particular RNB graph.

Remark 4.2.32. *Graph $K_{r,t}$ is the optimal graph for which Algorithm 1 would accomplish rearrangeably non-blocking routing.*

We can reach Remark 4.2.32 by observing that each link in $K_{r,t}$ can not be removed because Algorithm 1 needs it to route an inter party connection when necessary. Consider $u \in U$ and $v \in V$ and link $u \to v$. There exists a traffic pattern π_0 consisting of connection $x_0 = (u, v)$ such that x_0 has to be routed to $u \to v$ by the algorithm for the traffic pattern.

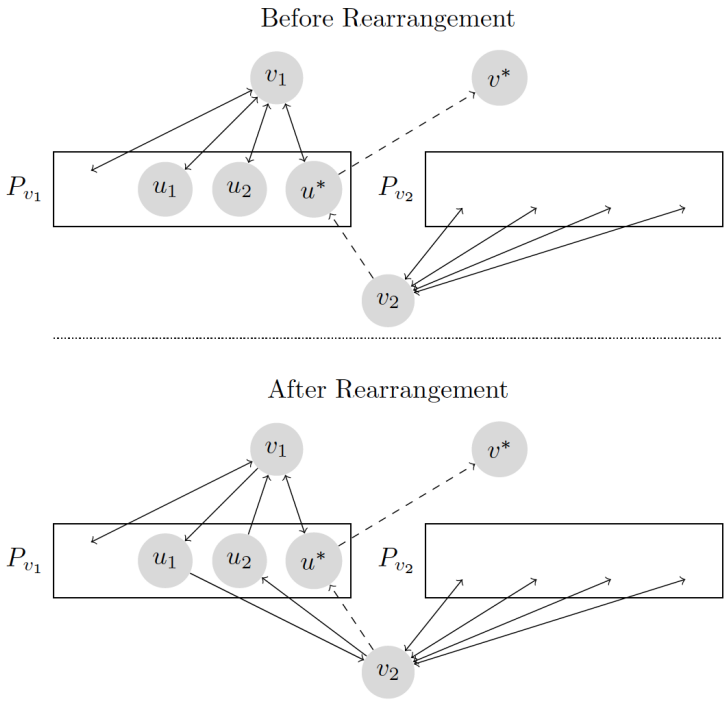

Figure 12: Rearrange process of Case 2

Otherwise, x_0 can't be routed and hence π_0 is blocked. Consequently, there exists no subgraph of $K_{r,t}$ that would keep the RNB property under Algorithm 1. We can conclude that $K_{r,t}$ is the optimal RNB graph by Algorithm 1.

If we fix the number of vertices, we can find several non-isomorphic bipartite complete graphs. Here we consider all of them as optimal because there exists no spanning subgraphs of them that are RNB by Algorithm 1.

Despite finding that $K_{r,t}$ is the optimal graph for Algorithm 1, we can see that there are unused links for every traffic pattern for most bipartite complete graphs. Unused links are observed in $K_{r,t}$ for each traffic pattern when $r > 2$ and $t > 2$, even if we have found that each link is necessary for the algorithm as in Remark 4.2.32. The number of links to route a traffic pattern by Algorithm 1 can be simply computed: an inter party connection uses 1 link in 1 direction; an intra party connection uses 2 links in 1 direction. There will be at most $r + t$ connections in a traffic pattern due to Proposition 2.4.6. They consume at maximum $2(r + t)$ links. Note that the graph has $2rt$ links if we distinguish the direction. Since we have $2rt > 2(r + t)$ when $r > 2$ or $t > 2$, we conclude that there are unused links in routing every traffic pattern, i.e. at every instant.

We can explain the redundant links by the cost of being flexible. The redundant links for a traffic pattern can be seen as back-up resources prepared for other traffic patterns. In order to be flexible as well as handle every traffic pattern, each link, despite not always being used, must be present.

In addition, it should be noted that Algorithm 1 is not a "prefect" routing algorithm which is able to find every possible routing path for a connection at its arrival instant. It only finds the shortest paths for each connection, the lengths of which are 1 or 2 depending on the connection type. It is unable to find longer routes even if there might be one. The limitation of the algorithm is a compromise to use the fewest network states yet guarantee RNB routing, which might subtly be the cause of redundant links.

Thus, it is an interesting question to investigate if there might be rearrangeably non-blocking graphs with fewer links if the routing algorithm is "prefect". We study two cases to shed light on this topic. In the first case we study the RNB subgraph of $K_{r,t}$. We can foresee that an inter party connection (u,v) may take a 3-length path since the direct link

$e = (u, v)$ may be absent in the subgraph. By brute force enumeration, we have Observation 4.2.33.

Observation 4.2.33. *We observed an RNB graph which is a subgraph of $K_{2,4}$. This graph can be obtained from $K_{2,4}$ by removing an edge*[6]*. It is plotted in Figure 13.*

However we found that the RNB property may not hold if we keep removing edges. Specially for all $K_{2,t}$ graphs we have:

Lemma 4.2.34. *Graph $K_{2,t}$, is not RNB if we remove more than m arbitrary links.*

$$m = \begin{cases} \frac{t-1}{2} & t \text{ is odd} \\ \frac{t}{2} - 1 & t \text{ is even} \end{cases}$$

Proof. Denote the parties as $|V| = 2$ and $|U| = t$. To prove the statement, we need to prove whether a spanning subgraph of $K_{2,t}$ after removing m edges, is blocking. As $K_{2,t}$ is symmetric, we know that:

1. we can cut a vertex u off by removing the two links incident to u.
2. we can cut a vertex v off by removing the t links incident to v.
3. we can cut v_1 and v_2 into two components by removing t links, none of which is incident to a same u.

Recall Theorem 3.3.1 which suggests the graph is blocking if two large components are interconnected by insufficient number of edges. We can confirm a blocking graph if we can find two components connected by a number of edges which is no more than the number of vertices in the smaller component. Thus, the minimum number of edges, after the removal of which $K_{2,t}$ becomes blocking, depends on the maximum size of the smaller component.

For an odd t, the largest number of the vertices for the smaller component is $1 + \frac{t-1}{2}$. For an even t, the number is $1 + \frac{n}{2}$. Assume graph $K_{2,t}$ is blocking by identifying two cut set theorem satisfying components, the size of the smaller one is $1 + \frac{t-1}{2}$ (t is odd) or $1 + \frac{n}{2}$ (t is even), after the removal of m edges. From the assumption we can deduce that no two

[6]Removing any edge is equivalent as the graphs are isomorphic.

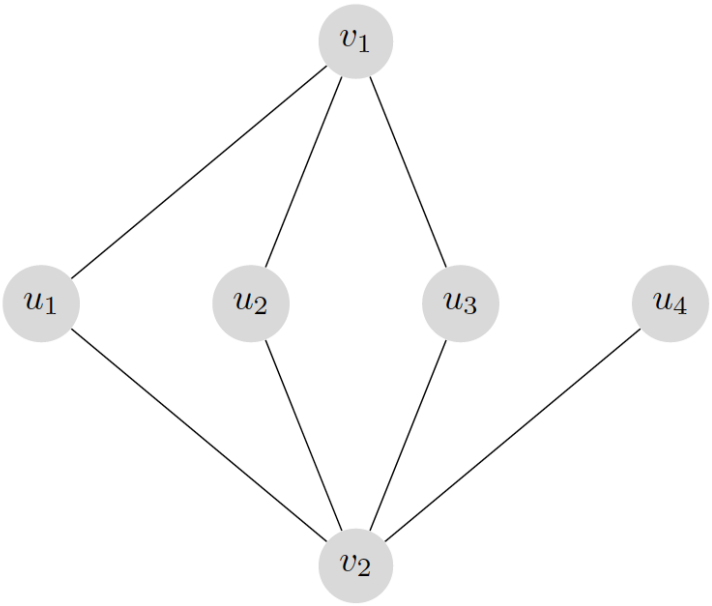

Figure 13: An RNB subgraph of $K_{2,4}$

of the m edges should be incident to a same u. Otherwise, u is disconnected and forms a component of size 1, which forfeits the assumption. We have:

$$t - m < \begin{cases} 1 + \frac{t-1}{2} & \text{t is odd} \\ 1 + \frac{n}{2} & \text{t is even} \end{cases} \tag{4.1}$$

and we have the solution:

$$m > \begin{cases} \frac{t-1}{2} & \text{t is odd} \\ \frac{t}{2} - 1 & \text{t is even} \end{cases} \tag{4.2}$$

Based on the result in equation 4.2, we find the minimum number of m that could make $K_{2,t}$ blocking. Next, we need to illustrate that removing any more than m links will render it blocking by satisfying the cut set theorem. We can divide it in two parts. First, we discuss the cases where no two of the m edges are incident to a same u. Then we discuss other cases.

First, consider removing m edges, where m satisfies equation 4.2. Among the m edges to remove, let m_1 edges be incident to v_1 and m_2 edges be incident to v_2, we have $m = m_1 + m_2$. From the equation we know $m_1 + m_2 \geq \frac{t}{2}$. Considering the symmetric property of $K_{2,t}$, we see that m_1 and m_2 are interchangeable. Without loss of generality, we can set $m_1 \leq m_2$. We know there are still $t - m$ vertices in U that are connected to both v_1 and v_2, and $t - m < m$.

Now after the m edges are removed, we must remove at least $t - m$ edges to cut the graph into two large components. We can selectively cut the edges to keep the size of the smaller component as large as possible. If $t - m \geq m_1 - m_2$, the prerequisite of equation 4.2 comes true and we know it is correct. Otherwise, the size of the smaller component would be $m_1 + t - m + 1$, after the future removal of $t - m$ links, which also satisfies the cut set theorem and hence blocks.

To summarize the first case, if we remove more than m edges while none of them are incident to one vertex u, we can confirm the graph is blocking by the cut set theorem. If fewer than m edges are removed, there might exist a combination of links such that the graph, after removing them, can't be confirmed blocking by the cut set theorem.

Second, we need to show that equation 4.2 holds if we remove any arbitrary m edges. If $t > 3$, we have $m > 1$. We know we need to remove at least two edges to satisfy the cut set theorem. If we set loose the constraint and we remove two removed edges that are incident

to one identical vertex u, vertex u is disconnected and we know the graph is blocking. Thus, we know removing more than m arbitrary edges when $t > 3$ will make $K_{2,t}$ blocking.

If $t = 1$ or $t = 2$, we have $m > 0$, we can intuitively observe that removing any edge would render the graph blocking. Thus the condition of the m edges being "arbitrary" is satisfied and the proof is complete. □

Lemma 4.2.34 provides an upper bound for the number of edges to remove, with the possibility of retaining the RNB property for $K_{2,t}$. Note that removing fewer than the upper bound number may also make a blocking graph while a general result is hard to determine.

In the second case we aim to find a better RNB graph that is not a spanning subgraph of any $K_{r,t}$. The problem becomes finding a rearrangeably non-blocking graph which is neither a spanning super graph nor a spanning subgraph of a bipartite complete graph. Note that it is insufficient to only compare the number of links without analyzing the structure of the graph because an RNB graph, with a very low number of edges, might be a super graph of $K_{1,t}$. This sheds no new light on the relationship between RNB property and the graph structure.

Observation 4.2.35. *A brute force search of simple graphs with no more than 6 vertices reports no such graph. However I speculate it may occur when the scale increases.*

If we fix the number of vertices as $\mathcal{V}$, it is very interesting to observe that graph $K_{1,\mathcal{V}-1}$ simultaneously achieves the following properties compared to all other $\mathcal{V}$-vertices bipartite complete graphs:

1. has least number of links;
2. is RNB (actually WSNB);
3. has no redundant links[7];

We can evaluate the degree of "being distributed" among $K_{r,t}$ by the value of $|r - t|$. The larger this value is, the larger the degree difference between the two parties is, and the less distributed the graph is. The graphs in the form of $K_{1,\mathcal{V}-1}$ are essentially centralized networks as there exists a vertex that is adjacent to all other vertices, which is considered to

[7]Zero redundant links means there exists at least one traffic pattern that make use of all links

be least distributed. The other graphs $K_{r,t}$ when $r, t \geq 2$ can be seen as better distributed networks compared to $K_{1,\mathcal{V}-1}$.

The listed properties of $K_{1,\mathcal{V}-1}$ indicate that the centralized networks have the distinct advantage of being easy and efficient to switch dynamic traffic. On the other hand, it implies that it costs more in links and efficiency for a more distributed network guaranteeing the same non-blocking property using the same routing algorithm. Further, we notice that the smaller the degree difference ($|r - t|$), the more links we need for $K_{r,t}$, given $r + t = \mathcal{V}$.

Thus we can conclude that for $K_{r,t}$ graphs, a centralized network is most efficient in link usage and algorithms whereas distributed networks are more costly in links.

In this section we analyzed the optimality of our main result. We demonstrated that $K_{r,t}$ is the optimal RNB graph for Algorithm 1. We also showed that by improving the algorithm, we found RNB graphs which are a subgraph of a bipartite complete graph. However, the improvement in scale may be limited since we demonstrated by Lemma 4.2.34 that all of its subgraphs might not be RNB anymore if we remove more than a certain number of edges. We also found that a centralized network ($K_{1,t}$) is the most simple yet efficient RNB graph while other bipartite complete graphs with the same scale are more costly in number of links.

4.3 CONCLUSION

This chapter studied the WSNB and RNB properties for bipartite complete graphs $K_{r,t}$ and proposed two routing algorithms for these two non-blocking properties respectively. Algorithm 1, along with several rearrangement techniques, led to the RNB property. Algorithm 2, modified from Algorithm 1, guaranteed the WSNB property. We proved these facts by analyzing the routing algorithm with the topology advantages of $K_{r,t}$.

The structure advantage of bipartite complete graphs reduces the difficulty of non-blocking routing analysis for dynamic traffic, which is rooted in the fact that the difficulty in analyzing traffic patterns and arrival sequences are greatly reduced in $K_{r,t}$. By the structure advantage, we only need to study a small group of traffic patterns and arrival sequences to

achieve non-blocking properties, whereas in a general connected graph we must enumerate everything.

These two algorithms are deliberately designed as online algorithms in a distributed manner such that they could route connections without knowing the global network status. This is granted by the structural advantage of $K_{r,t}$, which is a very typical example that the information from the structure of the graph can be used to facilitate routing.

The WSNB result is a new generalized result for non-blocking connected networks in switching/routing theory, to the best of our knowledge. It finds a WSNB non-blocking structure that is scalable up to infinity, along with a distributable routing algorithm that efficiently carries out non-blocking routing.

In addition, the WSNB result provides a solid counter example because Proposition 2.4.6 suggests that we need rearrangement if we route all intra party connections from one party to the same vertex through the busiest part of the network that can handle it. Rather, the WSNB property is achieved by Algorithm 2, which uses a smarter yet efficient way to route the traffic throughout the network.

Last but not the least, the results shed light on optimal non-blocking graph structures. Graph $K_{1,t}$ is bipartite complete and is WSNB even with algorithm 1, and enjoys 100% utility rate for some traffic patterns. In addition to this excellent utility rate, it even has the fewest possible number of edges needed–t, while the total number of vertices in $K_{1,t}$ is $t+1$. However, this is a centralized design because there exits a node that is connected to all other vertices. It provides evidence that a cost-efficient network might be centralized. A distributed network might need more links to achieve the same non-blocking property.

5.0 IMPLEMENTING NON-BLOCKING TOPOGRAPHIES TO REAL NETWORKS

This chapter introduces the design paradigm to set up an non-blocking network in NSFNet based on the main results in the previous Chapter. However, we can deduce that NSFNet is blocking by Theorem 3.3.1. As shown in Figure 14, after the removal of the four red edges, the entire network is cut into two components each with seven vertices. Obviously four is 4 than 7 and hence Theorem 3.3.1 is satisfied. Unfortunately we know that NSFNet is NOT RNB.

Albeit NSFNet is blocking, we can set up a virtual topology as a bipartite complete graph which is known to be non-blocking. In section 5.1 we use $K_{7,7}$ as the virtual topology and we compute the virtual circuit paths to build a $K_{7,7}$ network over NSFNet. In section 5.2 we propose two implementation frameworks to set up the virtual topology and demonstrate that the virtual topology can be implemented with current available devices. To evaluate the potential of redundant links in $K_{7,7}$ to handle overflown traffic, we propose a simulation framework which loosens the traffic constraint in Proposition 2.4.6 in section 5.3. We discuss and compare the simulation results and partially verifies the correctness of the simulation framework with queue theory estimations.

5.1 VIRTUAL TOPOLOGY

We can make the NSFNet virtually a bipartite complete and thus non-blocking, by setting up virtual circuits. We need to find the optimal paths for virtual circuits that uses least

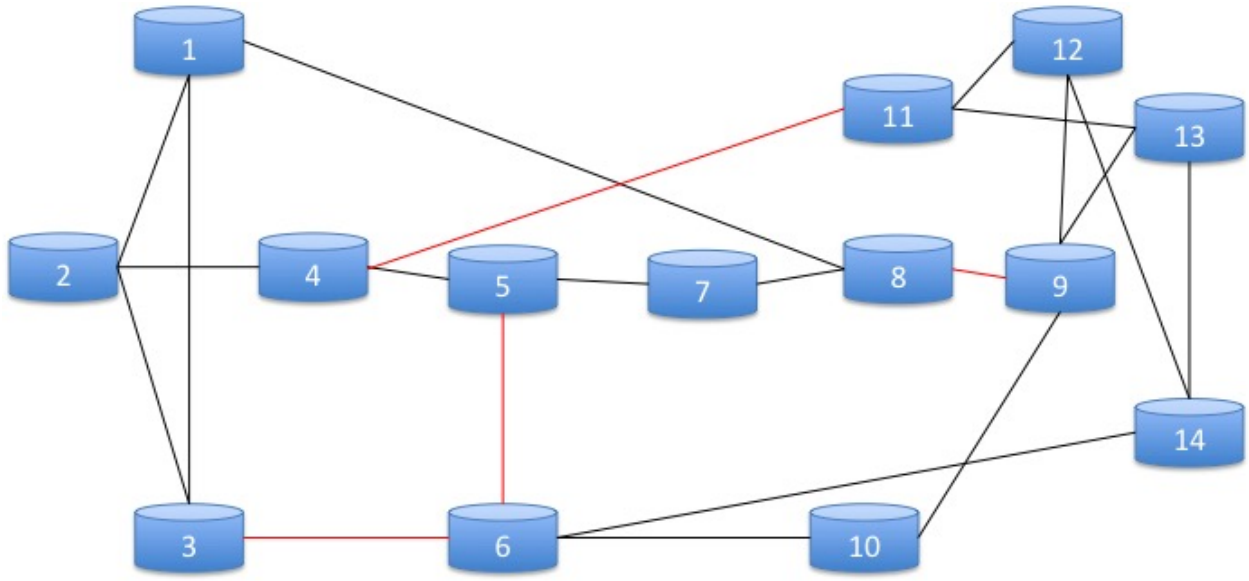

Figure 14: Edge cut set for NSF network

resources. This problem is in nature similar to the RWA problem and could be solved by ILP/MILP methods.

To obtain more general results, we consider that there is no wavelength converters in the system. To model the wavelength continuity constraint, we need to apply the graph coloring method to the ILP result to determine the wavelengths assignment.

We develop an optimal design scheme for virtual circuits by minimizing the number of virtual circuit we need. Note that each edge in $K_{7,7}$ are adjacent to two vertices in the different party. If we can properly set the party for the vertices in NSFNet such that a maximum number of edges are adjacent to two vertices belonging to different parties, the number of virtual circuits needed is minimized. This problem can be modeled as a graph coloring problem in NSFNet. The vertices can be painted two colors and the optimal painting method should produce the minimum number of edges between the vertices of the same color.

We can solve this problem by implementing greedy algorithm based on a minimum spanning tree. The algorithm starts with assigning a color to a random vertex, which is the

root of the spanning tree[1]. Then at each step a new vertex is attached to the tree according to the following criteria:

1. The new vertex is unpainted.
2. The color to be assigned to the new vertex should be different from its parent in the spanning tree.
3. The new vertex should be adjacent to minimum number of vertices with the same color (in the original graph) compared to that of other possible new vertices.

This algorithm is greedy and expected to be suboptimal as it only optimizes the minimum number of adjacent same color vertices pairs at one step. The algorithm has random factors as the result of this algorithm is dependent on the selection of root vertex of the tree and the selection of new vertices if multiple vertices have the minimum number of adjacent same color vertices. Fortunately for the scale of $K_{7,7}$, we can obtain a best solution to set up the virtual circuits by a number of brute force test runs.

Figure 15 illustrates a typical result of the above algorithm. We use this solution to build a virtual $K_{7,7}$ graph on NSFNet. In this color scheme 18 of the 21 links are inter-party edges, which are painted as purple links. The remaining 3 edges are not used and are separately painted.

Graph $K_{7,7}$ has 49 links. The greedy algorithm suggests 18 of them (the purple links, thereafter called "physically connected circuits") can be set in the physical layer directly. At these physical links one more wavelength/channel needs to be reserved. The remaining 31 of them needs to be set up as virtual circuits.

Next, we solve the problem to optimally set up virtual circuits by the MILP method mentioned in Section 2.3. The virtual circuit problem fits right into the model by modeling the virtual circuits as optical traffic connections.

We use the most classic model to arrange the the virtual links such that the number of virtual circuits on each edge is minimized. This objective is a minimax function for the number of connections in each link. The problem formulation process and notations are already mentioned in Section 2.3 and not reinstated. The complete model is presented as

[1]The initial color does not affect the solution due the the duality of the two colors.

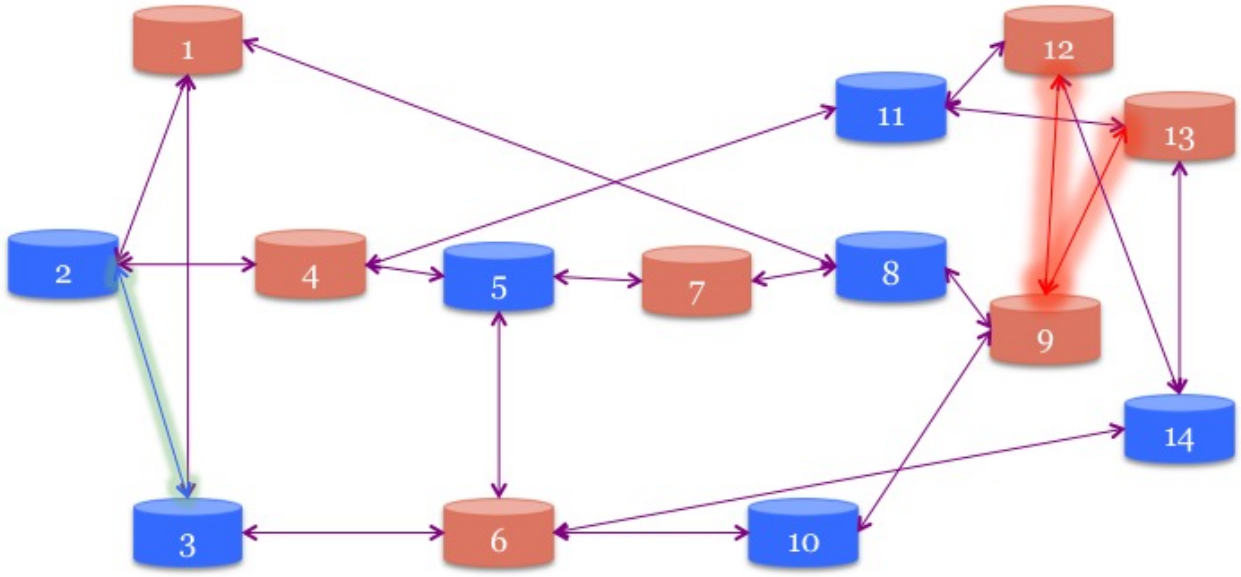

Figure 15: Bipartite Coloring on vertices in NSF network

following:

$$\text{Minimize } F \text{ where } F = \max_{\forall i,j} \sum_{\forall s,d} F_{ij}^{sd} + \sum_{\forall s,d} F_{ji}^{sd} + C_{ij}$$

with the following constrains:

for each vertex j, and each virtual circuit (s, d) pair, there is:

$$\sum_i F_{ij}^{sd} - \sum_k F_{jk}^{sd} = \begin{cases} 1 & \text{if } s = j \\ -1 & \text{if } d = j \\ 0 & \text{otherwise} \end{cases}$$

where:

$$C_{ij} = 0, 1$$

$$F_{ij}^{sd} = 0, 1$$

The F variables represent the path decisions for the virtual circuits. Variable C_{ij} is an indicator of extra wavelength usage for physically connected circuits. We have $C_{ij} = 1$ if (i, j) is a purple link in Figure 15. Otherwise $C_{ij} = 0$.

In this equation there are 31 virtual circuits and thus 31 (s, d) pairs. For each pair,

there are 42 directed edges related to it in F variables. So there are overall 1302 variables in this optimization problem. The optimal result is selected from a pool of 2^{1302} combinations, which renders brute force search impractical.

A technique called randomized rounding is used to acquire sub-optimal solution at lower computation costs. To achieve this, the integer constraint of F variables is loosened. The result for a connection may be multiple paths, each with a probability. The sum of the probabilities for the possible paths is 1. The routing path decided by a random picke from the possible paths according to their probabilities[2].

Next, we continue to use the sequential coloring method for the wavelength assignment problem, as we previously assumed wavelength continuity integrity (no wavelength converters) for a better generality[3]. Each virtual circuit is modeled as a vertex. Two vertices are adjacent to each other if and only if the two corresponding virtual circuits are routed to overlapping paths[4]. Thus by coloring the graph in a way that each vertex has a different color from its adjacent vertices, each virtual circuit is assigned a color that guarantees, the virtual circuits deployed on the links are differently colored for any link in the network. Despite a global optimal solution–a minimal number of colors– has been proven to be NP hard, sequential coloring offers acceptable and efficient suboptimal solution.

The paths to set up the virtual circuits are recorded in Table 5.1. It can be seen that the virtual circuit would take up 9 colors in NSFNet. Note that it is still necessary to assign one more wavelength to the link with physical connections. A total of 10 colors are needed to make NSFNet RNB.

For the design scheme of virtual circuits in Table 5.1, there are two different frameworks to carry out these virtual circuits, which are introduced in the next section.

[2]This algorithm has randomness and the optimality is discussed in [5]

[3]The solution with wavelength continuity constraint has more generality since it can be converted to a solution without it.

[4]Here, link(i, j) and (j, i) overlap as both directions needs to be reserved.

Table 3: Virtual Circuits Setup Configuration

VC	Path	Color	VC	Path	Color	VC	Path	Color
2 ↔ 6	[2,1,3,6]	6	2 ↔ 7	[2,1,8,7]	5	2 ↔ 9	[2,3,6,10,9]	2
2 ↔ 12	[2,4,11,13,14,12]	4	2 ↔ 13	[2,1,8,9,13]	7	3 ↔ 4	[3,1,8,7,5,4]	4
3 ↔ 7	[3,2,4,5,7]	5	3 ↔ 9	[3,6,10,9]	7	3 ↔ 12	[3,1,8,9,12]	2
3 ↔ 13	[3,1,8,9,13]	1	1 ↔ 5	[1,2,4,5]	2	5 ↔ 9	[5,4,11,12,9]	7
5 ↔ 12	[5,6,10,9,12]	4	5 ↔ 13	[5,6,14,13]	2	4 ↔ 8	[4,2,1,8]	3
6 ↔ 8	[6,5,7,8]	1	8 ↔ 12	[8,7,5,6,14,12]	3	8 ↔ 13	[8,9,13]	3
1 ↔ 10	[1,3,6,10]	5	4 ↔ 10	[4,5,6,10]	8	7 ↔ 10	[7,5,6,10]	9
10 ↔ 12	[10,9,13,11,12]	5	10 ↔ 13	[10,6,14,13]	1	1 ↔ 11	[1,2,4,11]	1
6 ↔ 11	[6,10,9,12,11]	3	7 ↔ 11	[7,5,4,11]	6	9 ↔ 11	[9,13,11]	2
1 ↔ 14	[1,8,9,13,14]	6	4 ↔ 14	[4,11,12,14]	2	7 ↔ 14	[7,8,9,13,14]	8
9 ↔ 14	[9,12,14]	1						

5.2 DESIGN FRAMEWORKS AND NODE FACILITY

In this section, we introduce and compare two frameworks to implement the virtual circuits according to Table 5.1. In the first framework, the virtual circuits are deployed at the same wavelength in different fibers. Every virtual circuit on a particular link (u, v) is put into different fibers such that they don't interrupt. This is called the " spatial framework". In the second framework, different virtual circuits are deployed in the same fiber using different wavelengths. This is called the " wavelength framework". These two frameworks are similar; however they have significant difference in the necessity of wavelength converters.

Each framework is illustrated in brief by a system diagram showing the hardware facilities in a specific node. The hardware facilities are connected/wired according to the virtual circuit setup scheme in Table 5.1. In addition, the wavelength assignments for the wires (if applicable) are plotted by the color of the wire. Next, the configuration of the switching facility for different traffic is introduced.

Different virtual circuits over a link are set in different fibers. Each link may have different number of virtual circuits, as listed in Table 5.1. The edges with an asterisk are the non-purple edges as shown in Figure 15. For example edge (2,3) only has two virtual circuits and no physically connected circuit, while edge (6,10) has eight virtual circuits and one physically connected circuit. This difference is due to the effect of randomized rounding in acquiring the solution.

5.2.1 Spatial Framework

In spatial framework the virtual circuits in a link are deployed in different fibers but have the same wavelength. The virtual circuits for different slices can be set in different wavelengths. Due to this way of assignments, the spatial framework is wavelength converter free.

To have a brief idea of the node structure in this framework, we present the switching facility for node 3 given the virtual circuits in Table 5.2.1 in Figure 16. All the links are labeled green as all links are assigned by the same wavelength. Different connections in the

Table 4: Number of virtual circuits in each link

Edge	No.	Edge	No.	Edge	No.
(1,2)	6	(1,3)	5	(1,8)	7
* (2,3)	2	(2,4)	5	(3,6)	4
(4,5)	6	(4,11)	5	(5,6)	6
(5,7)	6	(6,10)	8	(6,14)	3
(7,8)	5	(8,9)	6	(9,10)	5
* (9,12)	5	* (9,13)	7	(11,12)	4
(11,13)	3	(12,14)	4	(13,14)	5

same slice are spatially multiplexed into different fibers. According to Table 5.1, the link connecting node 3 and node 1 has five virtual circuits–$\{3 \leftrightarrow 4, 3 \leftrightarrow 12, 3 \leftrightarrow 13, 2 \leftrightarrow 6, 1 \leftrightarrow 10\}$. There is also one physically connected circuit $\{3 \leftrightarrow 1\}$ in the link. The virtual circuits $\{2 \leftrightarrow 6, 1 \leftrightarrow 10\}$ pass node 3, whose ports needs no reconfigurable optical add/drop multiplexer (ROADM). The remaining 3 virtual circuits and 1 physically connected circuit may be connected to an ROADM in case node 3 is the terminal for the traffic.

The OADMs needs to be reconfigurable as the destination of the traffic in the wire is unknown. To be more specific, the traffic could be destined at node 3, and the OADM should be configured to drop the traffic to local networks. However, it is possible that the traffic in the wire is an intra-party connection hopped by node 3, when when the OAM should be configured to be skipped, or handle the traffic to the OXC. For example, consider a pair of intra-party connection $(4, 13)$ and $(13, 4)$, which is hopped through node 3. This means that this intra-party connection uses both virtual circuits$\{4 \leftrightarrow 3, 3 \leftrightarrow 13\}$ while node 3 is the hop node. Upon the connection setup process, the ROADM should be reconfigured to forward the data from the port for virtual circuit $4 \leftrightarrow 3$ to the OXC. The OXC will be configured to switch the port for virtual circuit $4 \leftrightarrow 3$ to the port for virtual circuit $3 \leftrightarrow 13$.

In the infrastructure illustrated in Figure 16, seven ROADMs are needed since node 3 is

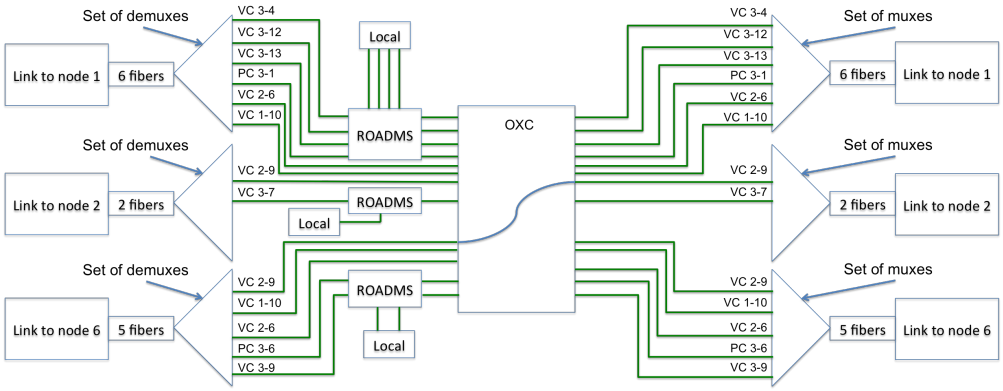

Figure 16: System Diagram for Node 3 in spatial framework with ROADMS

the terminal for seven connections. The flying-through virtual circuits are directly connected to the OXC without ROADMs.

The switching mechanism of the ports for flying-through virtual circuits can be designed as fixed[5], because the virtual circuits setup scheme for a RNB slice is expected to last unless reconfigured and does not change for different traffic patterns. The fixed wiring is shown by the blue curve (only one of them plotted). Under this assumption $7^2 = 49$ switching unites are needed. Otherwise $13^2 = 169$ switching units are needed. We can use fixed transponders to connect the ROADM since every " wire" in the framework uses the same wavelength.

Another node infrastructure to implement this framework with only OADMs is also feasible with the cost of more switching units. As shown in Figure 17, extra ports of the OXC are used to switch the data to or from an OADM. The A and D represents the line for add or drop respectively. It becomes the OXC's responsibility, other than the ROADM's, to determine whether to switch the data to local interfaces, by using fixed OADMs. However it requires more switching units to host the ports for OADMs. The OXC in Figure 17 needs $20^2 = 400$ switching units which is more than its counterpart in the ROADM design because the OXC needs more ports to host OADMs. To conclude, the OADM infrastructures in Figure 16 trades the cost of reconfiguration functions off against the cost of switching units

[5]It can also be designed that it does not pass the OXC. This design saves switching units. However it is risky because the node has no surveillance over the virtual circuit.

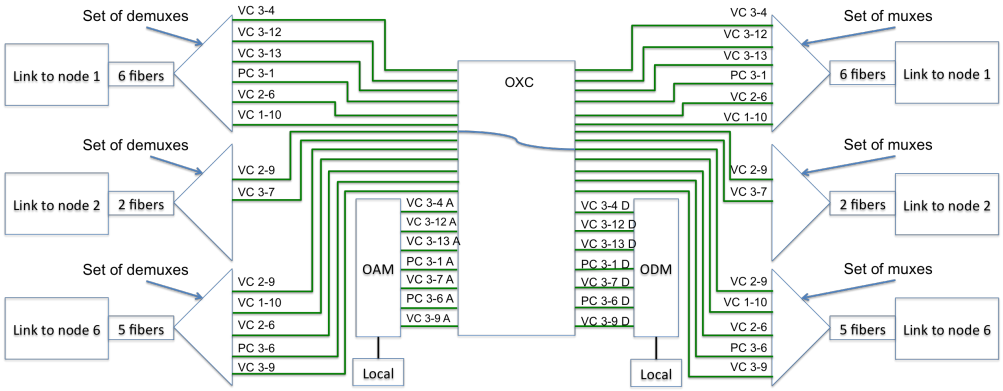

Figure 17: System Diagram for Node 3 in spatial framework with OADMS

of that in Figure 17.

The advantage of this spatial framework is that it does NOT need any wavelength converters. It saves the overall hardware cost in the system. Besides, the configurations of the infrastructure of the switching facility in the node can be pre-determined according to the virtual circuit paths in Table 5.1, while guaranteeing RNB and preventing unnecessary reconfiguration for dynamic traffic.

What is more, if every slice deployed is identical and the OXCs are transparent, we can upgrade the number of wavelengths in each fiber without altering the OXCs because every wavelength in the same fiber goes to the same destination.

The disadvantage is that it may need extra fibers for the virtual circuits, which may be a huge cost to deploy more fibers in the physical layer. In addition, the number of fibers needed in each link is different, which may bring trouble to slice management. If where there are 15 fibers with one wavelength in each link, only one slice can be set–one slice in edge (6,10) needs 10 fibers–and the remaining fibers are not used. This may be a potential waste of telecom resources.

5.2.2 Wavelength Framework

In the wavelength framework, the virtual circuits are deployed in the same fiber but in different wavelengths. This design mandates small scale wavelength converters in the switching facilities. Wavelength converting functions are necessary at the hop vertex of the path of an intra-party connection, if it connects two virtual/physical circuits which are assigned different wavelengths in case of intra party traffic. The scale of the wavelength converter should be at least equal to the wavelengths in the slice. Because every node may be a hop vertex for an intra party connection, a wavelength converter is mandatory for all nodes in this framework.

Figure 18 illustrates the infrastructure of the wavelength framework. There is only one fiber for every link connected to node 3, in which virtual/physical circuits are multiplexed into different wavelengths, which is differently colored. The OXC is equipped with wavelength conversion functions. There are at minimum $2 \times 8 = 16$ ports that need wavelength conversion. To save wavelength conversion functions, the ports of the OXC connecting OADMs are not equipped with the wavelength conversion function. The wavelengths assigned to these ports must be the same as the ports they are connecting to.

The number of wavelengths can be saved if wavelength conversion function is available at every port connecting the mux/demux when there are multiple slices. Consider two identical RNB slices set in the system shown in Figure 18. There are 6 wavelengths in link (3,1) and 2 wavelengths in link (3,2). Clearly in link (3,1), the first RNB slice would use wavelength 1–6 while the second use 7–12. If every port from the OXC to the links has **global** wavelength conversion, the wavelengths in link (3,2) can be assigned as 1–2 for the first slice and 3–4 for the second. Otherwise if only in-slice wavelength conversion is available, the wavelengths in link (3,2) must be assigned to 1–2 for the first slice and 6–7 for the second. Thus, wavelengths 3–5 in the link is dark due to the gap resulted from the imbalance usage of wavelengths. If first-fit wavelength assignment is desired, **global** wavelength conversion function is needed at $2 \times 13 = 26$ ports in node 3.

According to Table 5.2.1, there are at most 8 virtual circuits in a link. Thereby a wavelength converter capable of switching among 9 wavelengths would satisfy the minimum

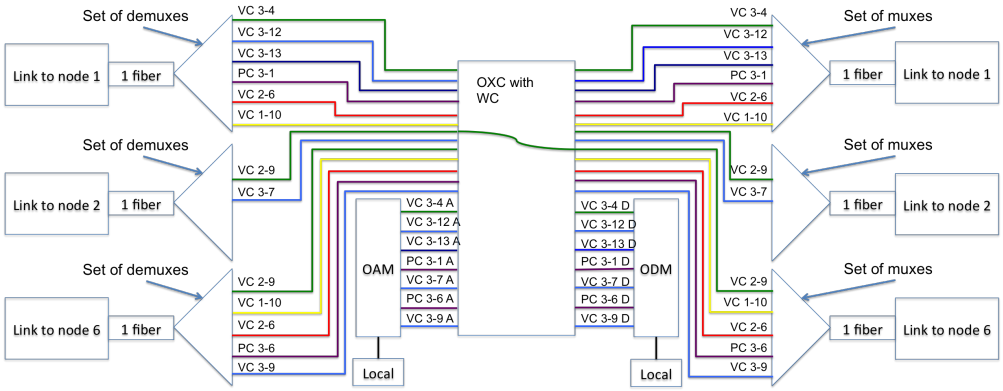

Figure 18: Node infrastructure for Node 3 in wavelength framework with OADMS

demand to make the slice RNB. However the wavelengths may not be allocated continuously due to the scale of wavelength conversion function and hence some wavelengths may be unused. This insufficient wavelength usage can be alleviated by using **global** wavelength conversions. Note that the trade off between switching units and reconfigurable OADM still applies in this framework. The transformation is nearly identical so the ROADM counterpart for Figure 18 is omitted.

The advantage of this implementation is we may have better wavelength utility by using, if possible, **global** wavelength converters to more ports. The system needs wavelength converters but small scale wavelength converters are still feasible–at the cost of less wavelength utility rate, or wavelength gaps. We may consider developing an alternative way to use the dark wavelengths in the gap to compensate the insufficient usage. This may be a meaningful future work if wavelength conversion techniques remains in bottleneck.

The disadvantage of this design is, similarly but more trickier, the unbalanced use of wavelengths because wavelengths in a fiber are a scarce and expensive resource. The number of wavelength conduits may be limited. In addition, wavelength converters may be expensive and a wavelength converting OXC for each slice at every node may be costly, even each in a small scale.

5.2.3 Summary of Frameworks

Both the frameworks provide an IP-router-free network infrastructure. The the routing algorithm (Algorithm 1) is online and can be implemented into the OXC and ROADMs without using IP routers because the routing algorithm is designed assuming the topology is $K_{r,t}$. The OXCs and ROADMs are placed and configured according to the virtual topology to properly route the traffic, which substitutes IP routing. With IP routers bypassed, we can design transparent all optical networks and reduce the system CAPEX.

The configuration can be divided into three cases for a node. First, for the traffic in the virtual circuit bypassing this node, it comes into and out of the OXC via fixed ports. These settings can be kept as long as the virtual topography holds. Second, for the traffic starting and terminate at the node, the ports and OADMs can also be fixed. If ROADMS are used, prompt commands are necessary to configure the ROADMs to direct the traffic to the proper destination. The third, this node is a hopping vertex for an intra-party connection. The OXC needs to be configured during the connection setup process to forward the traffic accordingly.

Table 5.2.3 summarizes the differences between the two frameworks. To better describe the detail, we divided the wavelength framework into two columns by whether it is equipped with small scale wavelength converters (SSWC) or global scale wavelength converters (GSWC). It is worth mentioning that when there are new wavelengths updated in each fiber, the spatial framework only needs to update the mux/demux and there is no need to update the OXC if the OXC is transparent to wavelengths.

Using wavelength framework with SSWCs may lead to wavelength gaps. This issue can be prevented by using GSWCs at extra cost. Considering the fact that adding new fibers/wires are more difficult and limited performance of wavelength converters, we prefer the spatial framework in real scenario implementation.

Table 5: Compare between Frameworks

Framework	Spatial	Wavelength /w SSWC	Wavelength /w GSWC
Fiber per slice	9	1	1
Wavelenth converter	NO	Yes	Yes
Wavelength per slice	1	9	9
Wavelength gap	N/A	Yes	NO
Wavelength update	minimum 1	minimum 9	minimum 9
Facilities to update	Only mux/demux	All	All
Fiber update	minimum 9	minimum 1	minimum 1
Facilities to update	All	All	All

5.3 EVALUATION OF SPARE CAPACITY

In Section 4.2.5 we observed that $K_{r,t}$ is, to some extent, redundant to a traffic pattern. This section evaluate the the spare capacity of the $K_{7,7}$ with a simulation framework in which the traffic constraint in Proposition 2.4.6 is loosened. The additional traffic are modeled and referred as "extra connections". This experiment is designed to satisfy the curiosity to evaluate the power of the redundant links to handle traffic. In practice, this scenario can be applied to the case that the unused links are leased.

The gain of link utility by allowing extra connections may be significant. In the previous chapter, we are using a virtual $K_{7,7}$ topography to make the NSFNet RNB. The total number of directed channels is 98. The most consuming traffic patterns use 28 of them. Suppose we can take full use of all the remaining directed edges. The link utility gain from allowing extra connections is estimated to be at least 250% if we can make full use of all links.

To allow extra connections, we need to assume that there are always a sufficient number of transponders in a node to make this network model valid. In real networks, a number of

tunable transponders may be used to dynamically meet this request.

This section proposes a mechanism to generate traffic, simulate the routing process, and evaluate the performance which is implemented by Matlab. This content is organized as following. Section 5.3.1 introduces the mechanisms to handle extra connections. Section 5.3.2 describes the setup and configuration for the simulations. Section 5.3.3 offers a sample run for the simulation and discusses the output. Section 5.3.4 studies the effect of priority on the blocking rate in the simulations. Section 5.3.5 discusses the differences between the stable period and the full timeline of the simulation. Section 5.3.6 partially verifies the simulation result with Queuing Theory. Section 5.3.7 analyzes and compares the blocking rate for intra party and inter party connections under different traffic densities.

5.3.1 The Mechanism

In this section a mechanism is proposed to handle extra connections with priorities. We allow extra connections to be served in best effort priority while regular traffic has first priority and still guarantees non-blocking. The best effort priority is lower than the first priority and thus a first priority connection may take the path of a deployed best effor connection, which may be rearranged or dropped. The priority of a connection is set according to the following rules.

1. A connection $v_1 \to v_2$ is set as *first priority* when node v_1/v_2 is not starting/ending any other *first priority* connections respectively.
2. Other connections are set as *best effort*.

The above rules guarantee that each node only initiates/ends one first priority connection and thus the set of first priority connections complies Proposition 2.4.6. Thus they are guaranteed RNB in $K_{7,7}$. The remaining best effort connections are served in a FCFS manner while the routing path is still determined by the same algorithm. A best effort connection may be blocked if it can't be routed. It may also be dropped by a first priority connection upon a path overlap.

To make it simple, blocked or dropped connections do not re-enter the system. Also, we do not elevate the priority of best effort connections to first priority even if at some instant

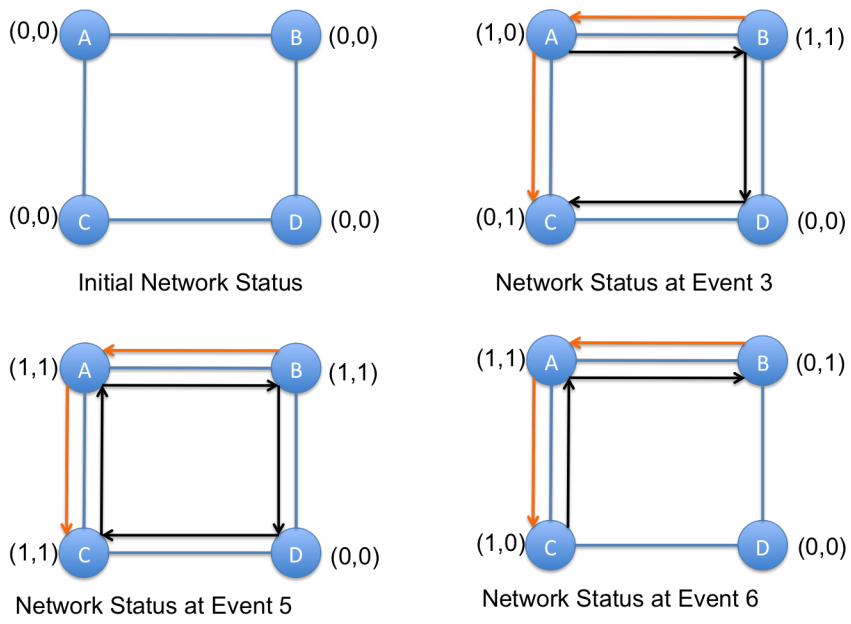

Figure 19: Snapshots of an example simulation

the best effort connection is the only connection active at both end nodes. If we do so, it becomes tricky to evaluate the performance of extra connections because the priority may change.

An example walkthrough of the priority assignment process is provided in Table 6 and Figure 19 to illustrate the priority mechanism in a $K_{2,2}$ graph, the vertices of which are denoted as A,B,C,D, respectively. We go through the priority assignment and path selection process of the set of events listed in Table 6. In order to better visualize rule 1 mentioned above, a binary indicator vector (i_s, i_d) is introduced for each vertex. The indicator i_s is set to 1 if the vertex is initiating a first priority connection. Similarly, i_d is set to 1 if the vertex is ending a first priority connection. The paths of first priority connections are plotted by black arrows while the paths for best effort connections are plotted by orange arrows.

Initially there is no connection in the network, every link is unused and the indicator vector for every vertex is $(0, 0)$. Before the arrival of event 3, there are two connections in the system, the ids of which are 1 and 2 respectively, both of which are set as first priority

Table 6: Example events for priority assignment and path selection process

Event	Connection Vector	Connection Id	Status	Priority	Path Selection
1	(A, B)	1	Arriving	First	$A \to B$
2	(B, C)	2	Arriving	First	$B \to D \to C$
3	(B, C)	3	Arriving	Best Effort	$B \to A \to C$
4	(A, B)	4	Arriving	Best Effort	Blocked
5	(C, A)	5	Arriving	First	$C \to A$
6	(B, C)	2	Disconnected	First	N/A

and the path is selected according to the proven method. These first priority paths are plotted in black arrows. At the instant connection 3 arrives, which is (B, C), we see that vertex B is the source of a first priority connection, and vertex C is destination a first priority connection–both are connection 2. Thus, connection 3 is given the best effort priority. Fortunately, wen can still find a path for connection 3. Despite vertex D is in the path of a priority connection, the indicator vector of D remains (0,0) because D is neither initiating nor ending any first priority connection.

For the same reason, connection 4 is given the best effort priority upon arrival. However, it has no path and thus it is blocked. At event 6, connection 2 is disconnected from the system. The i_s at vertex B and i_e at vertex C are both set to zero since the first priority connection has left. At this time connection 3 is the only traffic involving either node B or node C. However, as mentioned we do not elevate its priority to make the analysis simple. Note that if a connection vector (B, C) arrives the system at the timing of Event 7, it will be set as a first priority connection.

5.3.2 Simulation Parameters

We use the $K_{7,7}$ graph as the RNB virtual topography in this simulation. The set of virtual links that was set up for the system are configured as Table 5.1. The two parties are shown in Figure 15. It has 49 links in total, among which 31 are implemented via virtual circuits. Considering directed links/edges, the system has 98 directed virtual links which needs 250 directed physical links.

This simulation aims to study the blocking rates and link utility rates under different traffic scenarios. The simulation is implemented in MATLAB. To make it simple, we neglect the delay in connection setup process since it is small compared to the length of a connection. Furthermore, we assume there is no buffer in the system, due to the nature lack of optical buffers.

Without loss of generality, we assume traffic model for each connection (u, v) is Poisson. The event table with the arrival and departure time of the connections are generated according to the corresponding exponential distribution before the simulation starts. We denote the traffic density as $\rho = \frac{\lambda}{\mu}$, where λ and μ are the arrival rate and departure rate respectively. We assume the same traffic density for connections between each pair of vertices. This makes the result comparable with queuing theory solutions. We collect the link utility rate both in virtual and physical layers. We also collect drop rates and blocking rates. The system performance is evaluated by comparing utility rates and blocking/drop rate with the traffic density ρ. The utility usages of a sample run are plotted in Figure 20 and Figure 21, for virtual layer and physical layers, respectively.

We denote the arrival time of the i^{th} connection as t_a^i, and the disconnection time of it as t_b^i. Then the entire simulation timeline is the sorted array after merging t_a and t_d. The number of used virtual circuits (directed) at particular time t is denoted as l_v^t and its counterpart in physically connected circuits (directed) is denoted as l_p^t. Their counterparts with regarding to first priority connections are denoted as l_{v1}^t and l_{p1}^t. The number of blocked or dropped connections are denoted as c_d and c_b respectively while the total number of connections in the full simulation timeline is denoted as C.

Figure 20 plots virtual circuit usage metrics l_v^t and l_{v1}^t over the simulation timeline, with

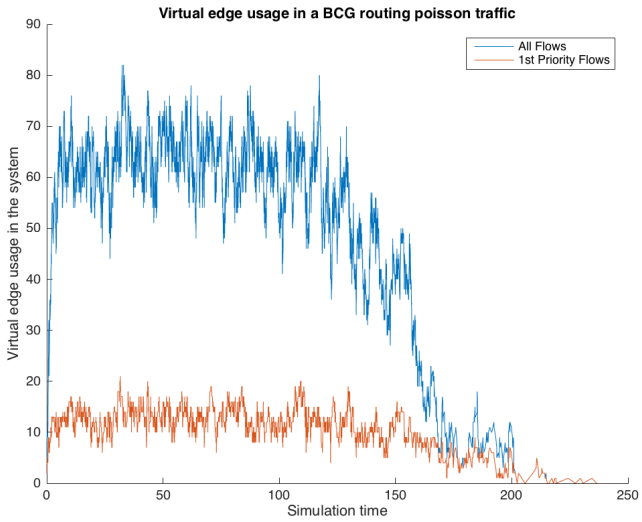

Figure 20: Virtual Circuits Utility Example

$\rho = 0.35$. The blue line represents l_v^t while the orange line indicates l_{v1}^t. Figure 21, similarly, plots the physical links usage (directed) l_p^t and l_{p1}^t.

As can be observed from both figures, all the utility rates keep stable until approximately 60% of the total simulation time. After that the lines trend to decrease and finally reaches zero because that majority of traffic is leaving the system in this period.

Based on the sample run above, we denote the time period $[2\%, 60\%]$ of the total simulation time as the *stable period*. That is, we consider that the system in this time period is in a stable state. We will further compare the difference of system performance in the stable period and full timeline in section 4.3.

In addition, we observed that, given ρ is the same, the performance metric does not have significant change with regard to different set of λ and μ. This means that a single parameter ρ would be sufficient to describe the system in our simulation. This is an expected statement from general queuing systems and thus not further extended.

We have the denotations for the other parameters that are also collected or studied in the simulation as following: VU is the virtual circuit utility rate, which represents the ratio

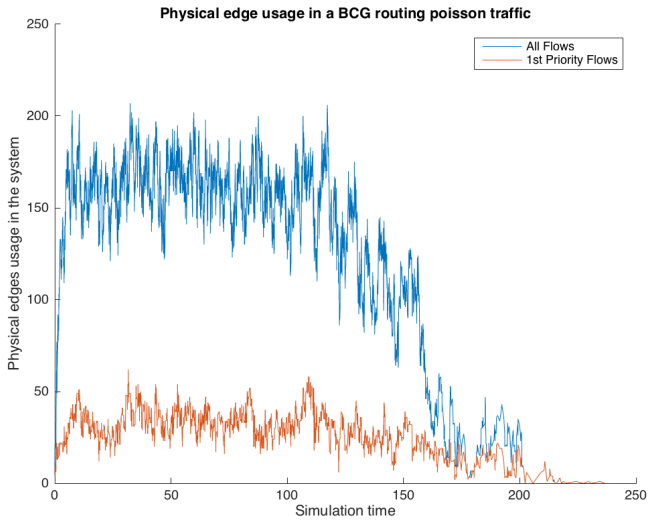

Figure 21: Physical Links Utility Example

of virtual circuits used to serve the traffic connections. Similarity, PU is the physically connected circuit utility rate, which is the ratio of physical links used. B is the blocking rate for the connections and B_d is their drop rate. Symbols ϕ_v and ϕ_p represent the ratio of virtual circuits and physical links that are used for first priority traffic to those used by all traffc. Symbol T_1 and T_2 represent the number of connections arrived in the stable period that are assigned first priority and best effort respectively. Among these parameters VU, PU, ϕ_v, ϕ_p, T_1, and T_2 are computed during the stable period as defined above. Parameters B and B_d are computed by the full simulation timeline data.

The computation of the parameters can be formalized as following:

$$VU = \frac{mean(l_v^t)}{98}, t \text{ in } \textit{stable period}$$

$$PU = \frac{mean(l_p^t)}{250}, t \text{ in } \textit{stable period}$$

$$\phi_v = \frac{mean(l_{v1}^t)}{mean(l_v^t)}, t \text{ in } \textit{stable period}$$

$$\phi_p = \frac{mean(l_{p1}^t)}{mean(l_p^t)}, t \text{ in } \textit{stable period}$$

$$T_1 = \#\text{ of first priority connections, } \ t \text{ in } \textit{stable period}$$

$$T_2 = \#\text{ of best effort connections, } \ t \text{ in } \textit{stable period}$$

$$B = \frac{c_b}{C} \ \text{ in } \textit{full timeline}$$

$$B_d = \frac{c_d}{C} \ \text{ in } \textit{full timeline}$$

$$B_d = \frac{c_d}{C} \ \text{ in } \textit{full timeline}$$

5.3.3 Performance Analysis

In this section we evaluate the performance for the proposed prioritized system in different traffic densities. We concentrate on system utility rate, blocking rate, and the portion of links used by first priority connections.

The data of the above parameters are summarized in Table 7. The utilization rates, both virtual and physical, increase with ρ, and have an upper bound at approximately 80%, even when the blocking rate is approximately 60%. It may be a critical factor that there is no buffer in this system, which limits the power to schedule traffic. This 80% value could be referred as an upper bound for the the link utility rate for poisson arrival connection requests without any buffers.

The blocking rate (B) is significant even at $\rho = 0.1$. It rises rapidly with ρ, which means that the network in a congested state when the traffic density increases. The drop rate (B_d) increases with ρ when $\rho \leq 0.5$. At this interval, with the rise of ρ, the number of connections getting admitted to the system increases as implied from the soaring VU and PU values. It enlarges the probability for a best effort connection to be dropped. Because there is more

Table 7: System performance for traffic connection with priorities.

ρ	VU	PU	B	B_d	ϕ_v	ϕ_p	T_1	T_2
0.1	0.253	0.257	0.069	0.006	0.356	0.359	1374.6	2326.2
0.2	0.455	0.458	0.140	0.015	0.249	0.251	882.2	2524.8
0.35	0.625	0.625	0.289	0.029	0.211	0.211	605	2223.6
0.5	0.701	0.702	0.414	0.033	0.202	0.204	457.2	1824.6
0.65	0.747	0.747	0.504	0.034	0.201	0.200	368.2	1569.6
0.8	0.774	0.775	0.568	0.034	0.198	0.200	308.2	1385.8

best effort connections in the network, it is more likely that a first priority connection is routed to overlapping path of a best effort connection which will be dropped.

The drop rate remains constant when $\rho \geq 0.5$. At this interval, the rise of ρ does not lead to significant increase of the number of admitted connections as implied from VU and PU values. This is due to the fact that higher ρ does not make more connections enter the system since the system is already highly congested. Most additional arrivals brought about by the additional traffic density are essentially blocked, instead of being served. Since the number of admitted connections remain stable, the chance for a connection to be dropped by a first priority connection does not increase significantly. This explains the ceiling of the drop rate at $\rho \geq 0.5$.

When $\rho \geq 0.5$, the system is largely congested. First priority traffic would use approximately 20% of the occupied resource. The percent remains relatively constant, and does not increase with the rise of ρ. In contrast, at low densities, a greater portion of the occupied telecom resources are used to than those at high densities. It can be explained by that an arriving connection is more likely to be assigned as first priority. And hence the portion of first priority connections is greater. So is the ratio of occupied resources. We could conclude that a greater fraction of the traffic connections will be assigned the first priority when ρ is low.

The 20% ceiling for ϕ_v and ϕ_p suggests that, ideally we can gain a 400% gain in link utility by allowing best effort connections. This fact indicates strong potential to utilize this best effort mechanism to share the links not used by a first priority connection.

We can observe that both T_1 and T_2 have its maximum value at $\rho = 0.1$ and keep decreasing with the increase of ρ. The value of T_1 decreases more rapidly than that of T_2. It can be explained by the fact that at a high ρ value the competition for the first priority privilege becomes more fierce so a limited number of connections are assigned to it compared to a simulation with low ρ value.

We also observe that the sum of T_1 and T_2 is decreasing with ρ. To understand the decrease of their sum, we should begin with the service rate configuration in the simulation mechanism. The service rate (μ) is kept constant for all simulations while the arrival rate is manipulated to according to the ρ values. The total number of connections generated is constant. Thus the simulation timeline for a large ρ value tends to be short as the time between two connection arrivals are short compared to its counterpart for a small ρ value.

Thus, it can be explained that the sum of T_1 and T_2 is decreasing because of increased blocking and drop rate with ρ. The number of connections the network can handle in a short period in the stable period is approximately constant when the network is over congested. The experiment with a high ρ value s more connections arriving at the short period while the number of connections admitted is constant. It explains why we observe a smaller portion of connections admitted in experiments with high ρ values. That explains the decrease of $T_1 + T_2$ with the increase of ρ.

By observing the utility parameters, it can be estimated to obtain at minimum 250% gain in link utility for our bipartite complete graph, as mentioned in the beginning of section 4.3. Clearly our result outperforms this bound. The reason is that, in our simulation environment, first priority connections are not using the most number of links needed. Thus more links can be used by best efforts connections.

However, we can't ignore the fact the link utility rate does not exceed 80%, no matter virtual circuits or physically connected circuits, which means that this system has redundant links.

5.3.4 Effect of Priority

This section studies the effect of setting priorities on the performance of the system. As we all know, first priority connections are guaranteed to be non-blocking, and best effort connections may be blocked or dropped. If we do not assign priorities, every connection is treated as a first come first serve manner. We investigate any potential impact of priority on the blocking rate with regarding to different traffic densities.

Specifically we denote *no priority mode* as the case in which all connection priority are set as best effort. This mode is designed to simulate the case that all connections have no priority difference. In no priority mode, no rearrangement is made for any connections and there is no drop. We use the term *priority mode* for the priority mechanism mentioned Section 5.3.1.

In this study we aim to compare the performance between the *priority mode* and the *no priority mode* under different traffic densities by collecting and evaluating the blocking rates and drop rates. We also introduce the concept of service-incompletion rate, which is the sum of blocking rate and drop rate, to better evaluate the no priority mode where there is no drop rates. In priority mode, we study both the blocking rate and the service-incompletion rate while the drop rate can be implied from difference of them. In no priority mode, we only study the service-incompletion rate, which is essentially the blocking rate.

As can be seen from Figure 22, each line raises monotonously with the increase of ρ. The service incompletion rate in priority mode is constantly higher than its blocking rate, the gap between which is essentially the drop rate. The service incompletion rate in no priority mode is lower than both the blocking rate and the service incompletion rate when ρ is low. It exceeds the blocking rate of the priority mode with the increase of ρ, and further converges to the service-incompletion rate of the priority mode.

To understand the performance we have to note that the system is congested when the traffic density is high. In a congested network the number of connections getting admitted in the system at an instant are bottlenecked, which does not have a strong dependence to whether we set priority or not. That is, the number of connections that get admitted has reached the capacity limitation of the network. In addition, we know the total number of

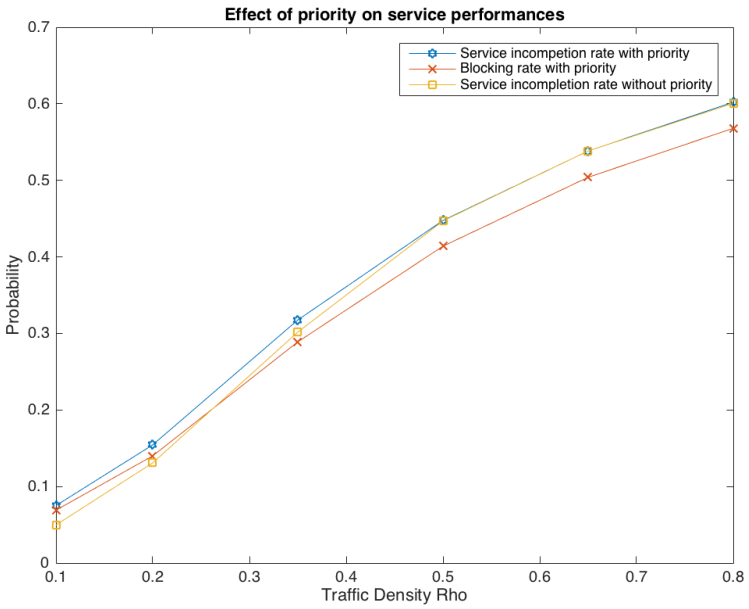

Figure 22: Service performances for priority mode and no priority mode

connections are close in the two modes with the same ρ, the service-incompletion rate of both modes becomes very close at high traffic densities.

At low densities, no priority mode has lower blocking rate. It indicates no priority mode accepts more connections than the priority mode at low densities. It can be argued that, in priority mode, there is a potential overhead cost for guaranteeing the prioritized connections RNB.

If we assume dropped connections have no QoS, dropped connections are then considered same as blocked connections. We can conclude that the QoS of the no priority mode is not less than the priority mode at all range of traffic densities.

It raises an interesting question that, if dropped connections have partial QoS, the compare between priority mode and no priority mode becomes tricky since it depends on the weight for the partial QoS for dropped connections. Based on this condition we can conclude that the QoS of no priority mode definitely beats that of priority mode when $\rho \leq 0.3$

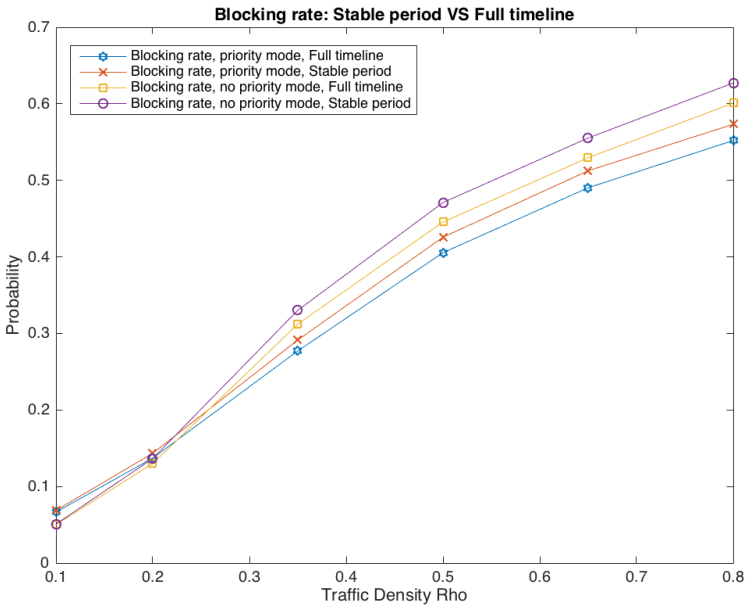

Figure 23: Blocking rate: Stable period vs Full timeline

approximately. When $\rho \geq 0.3$, the service incompletion rate in no priority mode lies in the gap between the regular mode blocking rate and priority mode service incompletion rate. The QoS comparison result in this interval depends on the weight of the partial QoS of the dropped connections.

5.3.5 Stable Period

In this section we compare the results collected from the stable period and the full timeline period of both the priority mode and no priority modes. In this experiment we aim to shed light on the difference between stable period statistics and full timeline statistics in studying the blocking rates.

Figure 23 illustrates the blocking rate in both modes, from both stable period and full timeline. We can see that the blocking rate in stable period is modestly higher that in full timeline when $\rho \geq 0.3$. Their values at low ρ values are close and their differences are

indistinguishable. This can be explained that, after the stable period before the end of the simulation, the system utility is moving to zero as connections are leaving the system. The connections arriving at this period are much less likely to be blocked since the network is no longer as congested as in the stable period. This is the reason for the full timeline blocking rate to be lower than that of the stable period.

We also observe that the full timeline no priority mode blocking rate is even higher than the stable period blocking rate in the priority mode when $\rho \geq 0.3$. This means that, the impact on blocking rates from priority is greater than that from period selection. We can conclude that the blocking rate in priority mode is lower than that in no priority mode.

Consider that the blocking rate in no priority mode will converge to the service incompletion rate in priority mode when ρ becomes large. It can be seen that the slight advantage of the blocking rate in priority mode will be compensated by the increasing drop rate when the network is congested. We can conclude that, based on the previous reasoning, that the performance of no priority mode is equal to, if not slightly better than that of the priority mode in service completion rates. In the subsequent contents, we only study the no priority mode both for its possible slight performance advantage, and the the convenience of implementation.

5.3.6 Queue Theory Verification

It has long been a critical challenge for simulation results that there maybe unspotted computational error or bugs in the program, which would forfeit the result. In this section we verify the simulation result by queue theory.

It is very hard to precisely evaluate a large scale networking system, especially those with complicated protocols. Take our network as an example, there are 14 inputting queues sharing 98 directed links. There is 2^{98} states for the links that we need to evaluate. Consequently it is impractical to model the entire network. Nevertheless, we can partially verify the simulation results with queue theory.

If we only allow the inter party connections, the overall network can be modeled as a set of independent M/M/1/1 queues. We can thus verify the simulation result by comparing

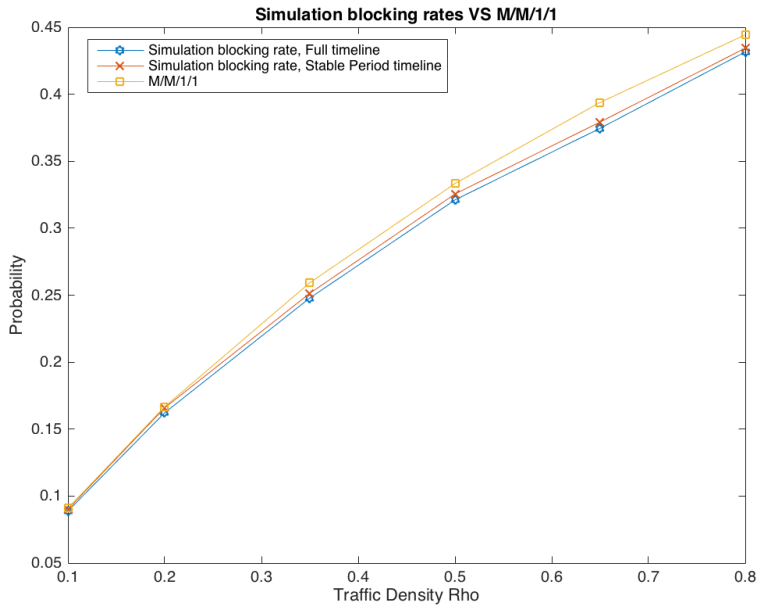

Figure 24: Queue Theory Verification of inter party traffic blocking rates

the simulation results to M/M/1/1 results. The data is collected from both the full and the stable period to provide a better understanding.

As can be seen from Figure 24, both the full timeline blocking rate and the stable period blocking rate appear to be lower than the M/M/1/1 estimate. The stable period blocking rate is closer to the M/M/1/1 theoretical value. The reason is that the M/M/1/1 estimates the system status at a stable state. The full timeline blocking rate takes into account of some time interval when the system is not stable and has a lower blocking rate. Since the stable period blocking rate better fits the nature of M/M/1/1, it is closer to the M/M/1/1 estimate.

The difference between the M/M/1/1 estimate and stable period blocking rate may be caused by some nature noises in the generation of Poisson traffic. For example, the generated arriving time according to exponential distribution in the simulation may not be perfect.

This result could partially verify the functions for inter party traffic routings as the the

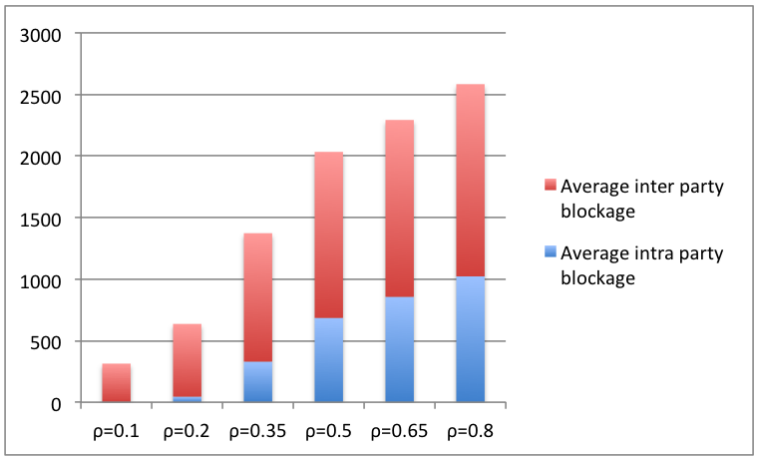

Figure 25: Average number of blocked connections in priority mode

blocking rates in the simulation match the estimate of M/M/1/1. While it is very hard to verify the comprehensive simulation system as a whole, a partial verification provides a decent verification of the results as well as a partial understanding of the deviation introduced in the simulation process.

5.3.7 Analysis of Blocked Connections

In this subsection we analyze the blocked connections in the set of simulations. We focus on the difference between inter party connections and intra party connections. As mentioned above, a intra party connection has more candidate paths than a inter party connection has. Thus we study the blocked connections for either group and compare the results.

Figure 25 and Figure 26 plot the average number of blocked connections for intra and inter party connections in priority and no priority modes respectively. In both figures, the total number of blocked connections rise with the increase of ρ. It shows that the number of blocked connections in either inter or intra party connections increases universally with ρ. We can also observe that there are more inter party connections blocked than intra party connections. This is due to the fact that inter party connection has less candidate paths compared to intra party connections.

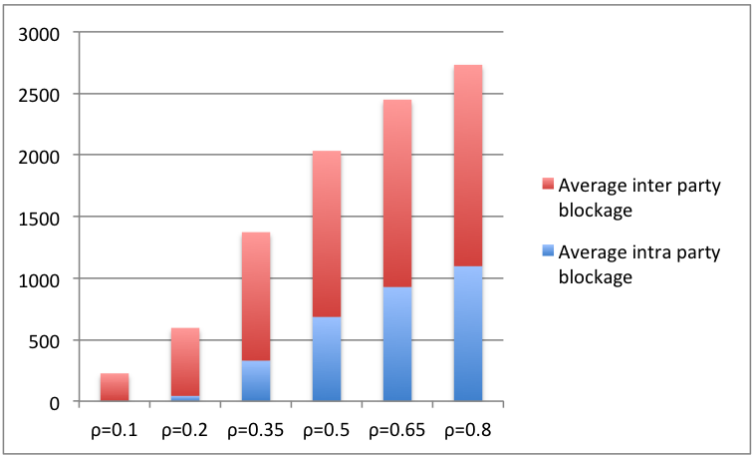

Figure 26: Number of blocked connections in no priority mode

Table 8 summarizes the ratio of blocked inter party connections among all blocked connections, which is further referred as *block ratio* in following contents. It can also be observed that the number of blocked intra-party connections is significant lower than that of inter party connections when ρ is low. It can also be explained by the fact that inter party connection is in nature more likely to be blocked due to limited candidate paths.

Besides, we can observe that at low densities, the number of blocked intra-party connections is negligible compared to inter party connections. This can be explained by that the probability that an intra party connection is blocked at low traffic densities is very small, because It is very rare that an intra party connection has no hop to choose upon its arrival when the network is not congested.

When ρ is large, the number of blocked intra party connections are close to that of inter party connections. That is, the probability for both types of connections becomes close, compared to when the network is less burdened. We can observe that in a congested network, intra party connections no longer have the significant advantage in path options to inter party connections. It can be explained as following. When traffic density is high, the network is congested. At the point of congestion, any new connection has an equal probability to de blocked as it is hard to even find one path. Thus, the numbers of blocked intra

Table 8: Block ratios of inter party connections among all blocked connections in priority and no priority mode.

ρ	Block ratio (priority)	Block ratio (no priority)
0.1	0.9973	0.9957
0.2	0.9289	0.9279
0.35	0.7574	0.7598
0.5	0.6723	0.6633
0.65	0.6266	0.6213
0.8	0.6044	0.5987

party connections and blocked inter party connections become close.

We could conclude that the priority is not playing a significant role as the block ratios between the two modes are very close. It suggests that setting priority does not have significant effect on the block ratio for the two type of connections.

5.4 CONCLUSION

This section introduces and proposes a simulation framework to evaluate the potential of the redundant links of $K_{7,7}$ to handle traffic in a general scenario. It also proposes a priority mechanism to distinguish overflown traffic and serve them at a lower priority.
From the set of observations we could conclude:

1. The service incomplete rate in no priority mode is equal to, if not slightly better than that in priority mode.
2. In performance evaluation, stable period statistics is on average higher than full timeline statistics and are closer to theoretical estimations.

3. Most blocked connections are inter party connections, because they have limited path option compared to intra party connections. Priority has little effect in this issue.
4. The simulation results are partially verified by queue theory.

The results provide a basic comparison of certain parameters among several configuration/modes. These results are base stones for further analysis. For example, when studying blocking rate, we will only analyze the blocking rate in no priority mode. In the next chapter, we introduce multiple slices in a network and the simulation becomes even more complicated.

6.0 PERFORMANCE ANALYSIS FOR MULTIPLE SLICE NETWORKS

A slice is a network model defined by a graph G where the capacities of all links are the same (both directions) and serve one connection. For a detailed explanation and modeling process of it, refer to Appendix A.3.

In the section above, we studied the routing performance of our algorithm in a one slice $K_{7,7}$ network. However, a backbone network conduit may be able to host multiple slices to achieve more throughput. In order to better depict and study the performance for multiple slice networks, we developed new features for the simulation framework mentioned in the previous chapter.

In multiple sliced routing, we need to implement the slice selection function for a connection because it can be routed to a number of slices. We propose two slice selection algorithms: Lowest Slice and Random Slice and compare their effect on system performance metrics.

We also propose a "3-hop" modification to the proposed algorithms. It allows in inter-party connection to be alternatively routed on a 3-hop path, which enables more routing options for inter-party connections.

Further, we compare the cost efficiency between the "3-hop" routing and the widely used K-shortest path routing algorithm for a set of network structures to evaluate the algorithms and the network structure in a comprehensive manner. The cost efficiency is evaluated by the number of links for a blocking rate.

6.1 RELATED WORK

Before we start the details of the simulation experiments in this chapter, we begin by a brief review of several related papers in which simulation is used to collect, verify and evaluate the performance of prototype optical networking techniques proposed by the authors.

Costa [69] uses the *Optical Network Simulator* to evaluate the performance of his resource allocation and optimal decision method designed for optical elastic networks (OEN). Zhao and Tian [72] use a simulation framework to evaluate the performance of auxiliary graph facilitated traffic grooming techniques in space division multiplexed (SDM) OENs. Tan and Gu [73] use simulations to verify the gain in energy saving of an MILP method design for efficient routing in OENs. The result is compared to the performance of K-shortest path algorithms. Min and Zhou [74] simulated their proposed heuristic developed for a surveillant network technique, and compared the simulation result to similar techniques. Ali [75] uses a simulation platform to study the service provisioning time in EON to cope with dynamic traffic.

While the above work simulates the network protocols and mechanisms, simulations can also be used to study optical networks by evaluating signal qualities, which is a typical application in telecom studies. Zhou [70] used simulations to evaluate the gain in bandwidth brought by bandwidth variable transponders (BVTs) in OEN, which is focused mainly in the scope of signal processing. Zhao and Hu [71] use simulations to evaluate the signal quality in their proposed ROADM ring architecture.

The simulation framework in this paper is closest to [75] as both evaluate the performance of a prototype routing mechanism as well as compare the result to its counterpart of the K-shortest path algorithm. For the special task in this paper, the simulation programs are all originally developed, except the function of K-shortest path routing, which is obtained from open sources. In the following sections, we introduce the simulation configurations and present the simulation results.

6.2 SIMULATION PARAMETERS

This section introduces the general parameters for the set of simulations in this section. Section 6.2.1 introduces the new features and configurations of the Matlab program. Section 6.2.2 introduces the slice selection algorithms and a 3-hop modification.

6.2.1 General Program Parameters

We introduce a new network structure which contains multiple identical $K_{7,7}$ slices based on the matlab simulation framework in the previous section. A multiple slice network can be denoted by the number of slices along with the underlying simple graph of the slice, for instance $K_{7,7} - 3$. If the graph is mentioned in the context, the network is denoted by the number of slices. This notation style is used in the figures in this chapter.

In the new framework, the slice utilization rates becomes a vector, consisting of one rate for each slice. The blocking rates and drop rates remain a scalar[1] and they are computed for the entire network. For each network we collect blocking rates and drop rates to evaluate its performance with traffic densities. In each slice of a network, we also collect the slice utilization rate. We collect the statistics for all above parameters (if applicable) from the stable period because the stable period statistics better depict the status a real network which keeps running.

The traffic parameters in the simulations in this chapter are the same. We still assume that the traffic between each source/destination pair is Poisson distributed with service rate $\mu = 0.5$. The traffic density is a set $\rho = \{0.2, 0.35, 0.5, 0.65, 0.8\}$. The arrival and departure time for the traffic between a source/destination pair is generated by a random algorithm according to ρ and μ.

[1]Because a connection can select a slice to be routed, defining the blocking and dropping rates in a slice is meaningless.

6.2.2 Slide Selection Algorithms and 3-hop Modification

This section introduces three new algorithms or modifications that are involved in the set of simulations for this chapter. Two of them are slice selection algorithms that determine a slice for an arriving connection. We also introduce the "3-hop" modification of the routing algorithm which allows more flexible routing for inter party connections.

As mentioned, in multiple slice routing, a connection may be routed to a number of slices. We propose two algorithms to determine the slice it is routed to: lowest slice algorithm and random slice algorithm.

The lowest slice algorithm selects the lowest numbered slice for an arriving connection. For example, if a connection can be routed in slices {2,3,4,5}, slice 2 will be selected and routes this connection. Note that this option is applied to all connections in a simulation run. Once applied, there will be a monotonic decrease in the utilization rate for each slice with the increase of slice number. High numbered slices may be unused if all the connections can be handled by lower numbered slices.

Random slice routing, alternatively, distributes connections evenly among the slices. For a connection, each slice that is able to offer a path to it has an equal probability to be selected. For example, consider the same case as mentioned in lowest slice routing. Random slice routing grants each slice a probability of 25%; so, throw the dice to determine the slice number. Random slice routing does not have a specific preference for certain slices and tends to distribute connections evenly.

We propose a "3-hop" modification to Algorithm 1. This modification allows an inter-party connection to be routed on a path of length 3, instead of the direct connecting path only. It offers more flexible routing for inter-party connections and is expected to reduce the number of blocked connections.

With this 3-hop modification, an inter party connection can also be rearranged. It may be rearranged from a length 3 path to another length 3 path by shifting two links along the path, like rearranging the intra party connections. In this process, the length of path may become 3 from 1 after rearrangement. It may also happen that it becomes 1 from 3 because the path contains a length 2 directed cycle which gets truncated.

6.3 SIMULATION STUDY FOR SLICE ROUTING METHODS

This section studies the performance of lowest slice routing and random slice routing for several multiple slice $K_{7,7}$ networks. We conduct a comprehensive study of slice utility rates, blocking/drop rates, and the blocking analysis to compare their performance.

6.3.1 Lowest Slice Routing

In this simulation we study the performance of lowest slice routing in multiple $K_{7,7}$ slice networks by analyzing the slice utility rates, blocking rates, and drop rates, which are obtained from simulations. We use three test networks in the simulation which have 3,4,5 $K_{7,7}$ slices respectively. As drop rates are among our interests, priorities are given for traffic in this simulation.

Figure 27 plots the blocking rates and drop rates for the test networks. The upper figure records the blocking and drop rates in linear scale, while the lower figure plots all blocking rates in a log scale[2]. The abbreviation "BR" represents blocking rate and "DR" represents the drop rate. The number of slices to distinguish the test networks is written after the abbreviations in the notation.

It can be observed that the 4-slice blocking rate line is globally lower than the 3-slice blocking rate. And the 5-slice blocking rate is lower than that of the 4-slice. We can observe that the blocking rate at the same traffic density decreases with the increase of the number of slices in the network.

We can also observe that the blocking rate grows with ρ. We also observe that the benefit for an extra slice in reducing the blocking rate is very obvious, which is shown by the "gap" between two neighboring lines[3]. These observations are consistent with the expectation that more slices would handle more traffic and thus the blocking rate is reduced.

While the observations for blocking rates are reasonable, its counterpart for the drop rates are odd. We can observe that drop rates are significantly higher than blocking rates for each network at each traffic density, from the upper part of Figure 27, and the drop rates

[2]Zero values are not plotted to prevent misleading lines.

[3]The "BR-5" line is not plotted in this figure since most blocking rates in the 5-slice network is zero.

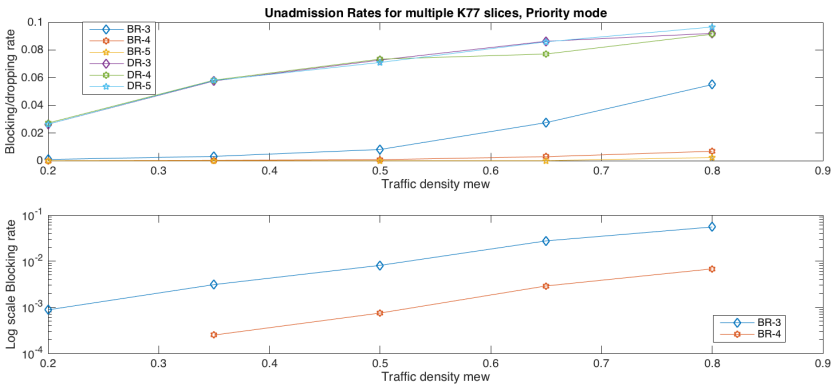

Figure 27: Blocking/Drop rate comparison in priority mode

in all the networks are close. They are approximately constant over the number of slices in the network and grow slightly with the increase of ρ. To better study this odd observation, we further collect the drop counts and slice utilization rate as shown in Figure 28.

Figure 28 is a scatter plot for drop counts and slice utilization rate for each slice in the test network without specifying which test network a slice belongs to. It is shown that there is a positive correlation between the number of drops and the utilization rate for a slice. The drop count rises with the slice utilization rate. This suggests that a highly utilized slice tends to produce significantly more drop than the modestly utilized slices.

Figure 29 records the slice utilization rate for each slice in each of the test networks (3, 4 and 5 slices networks). In each network, we observe that a slice is always significantly more utilized than the slice numbered next to it, which is caused by the preference of the lowest number slice routing. To our surprise the lowest numbered slice keeps a utilization rate at above 40% at even the lowest ρ in the simulations. Combining the observation from Figure 28 that lower numbered slices are highly utilized even at low densities, we can deduce that the lower numbered slices contribute a significant amount of drops compared to other slices in the network. That is, in every simulation no matter the number of slices in a network, there exists a few highly utilized slices due to lowest slice routing, which produces large amount of drops. It explains the fact that drop rates do not have strong correlation with

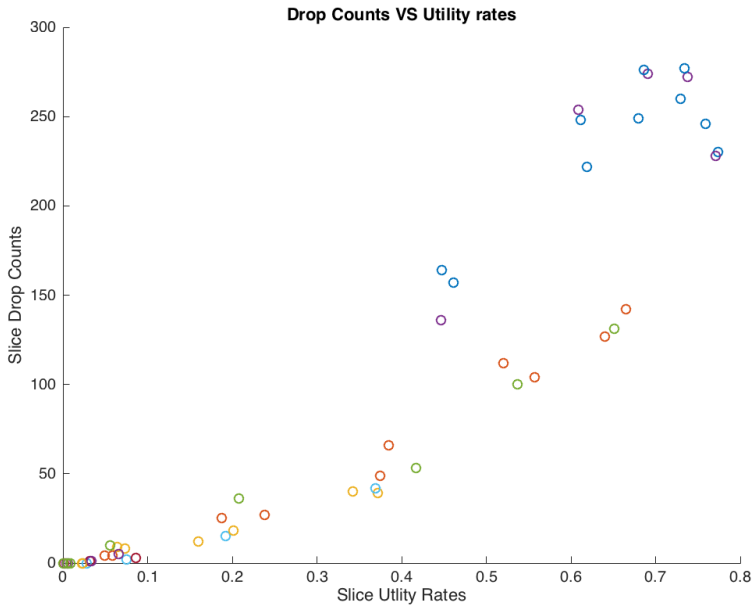

Figure 28: Scatter plot of drop counts

the number of slices.

At high traffic densities, the high numbered slices may also be congested and contribute significant drops. With the rise of ρ there will be more highly congested slides and they will produce more drops. As can be observed from Figure 29, the traffic density needs to be extra-high to make the subsequent slices highly utilized, and traffic density ρ must increase greatly to produce more droppings in these slices. These facts explain the observation that the drop rates slight increases with ρ.

It can be concluded that in this simulation the drop rate is the major part of the uncompleted traffic. For each network, the first slice contributes the majority of drops as it is the most highly used slice in the network due to lowest layer routing.

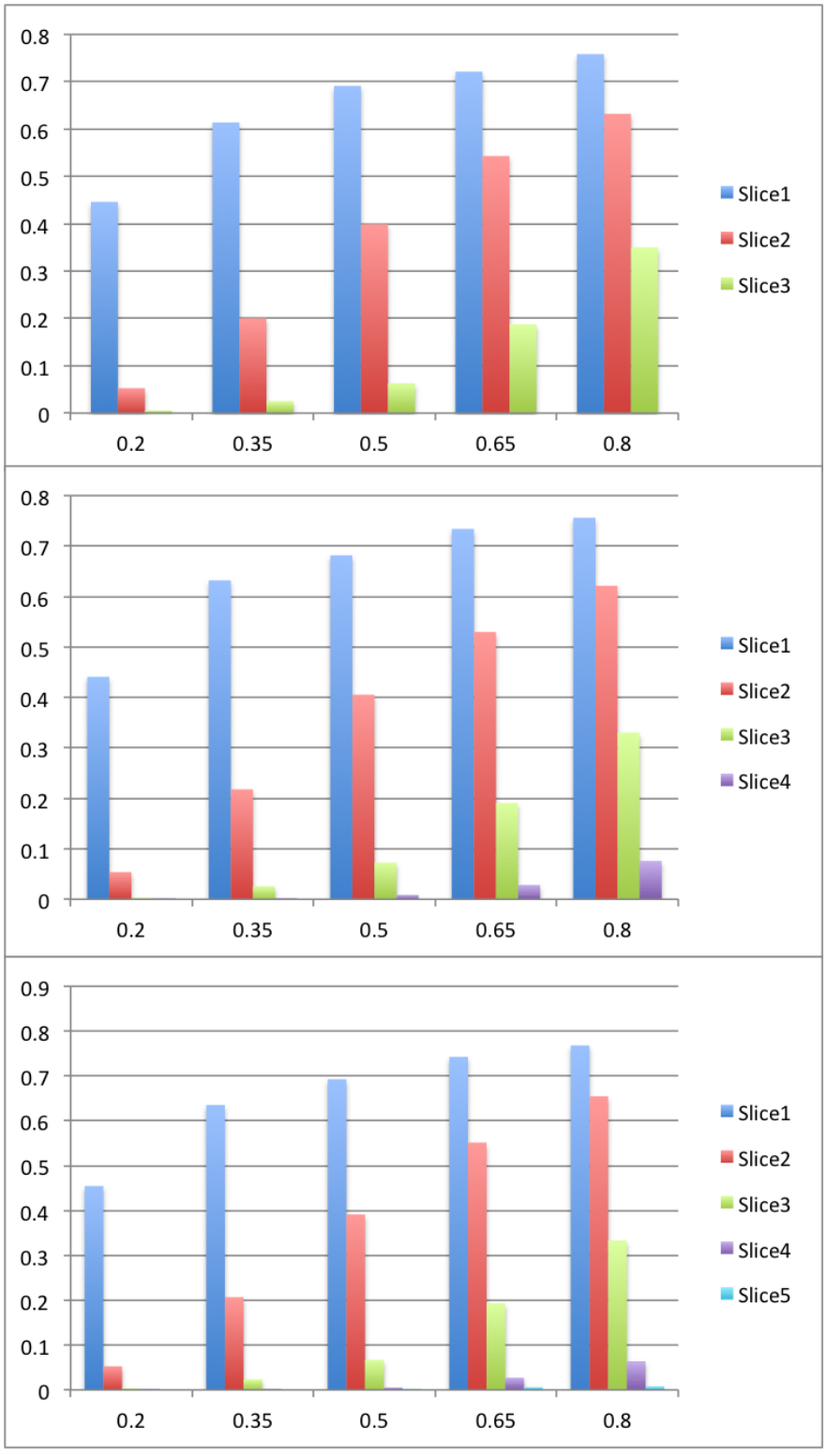

Figure 29: Slices utilization rates in multiple $K_{7,7}$ slices by lowest slice routing

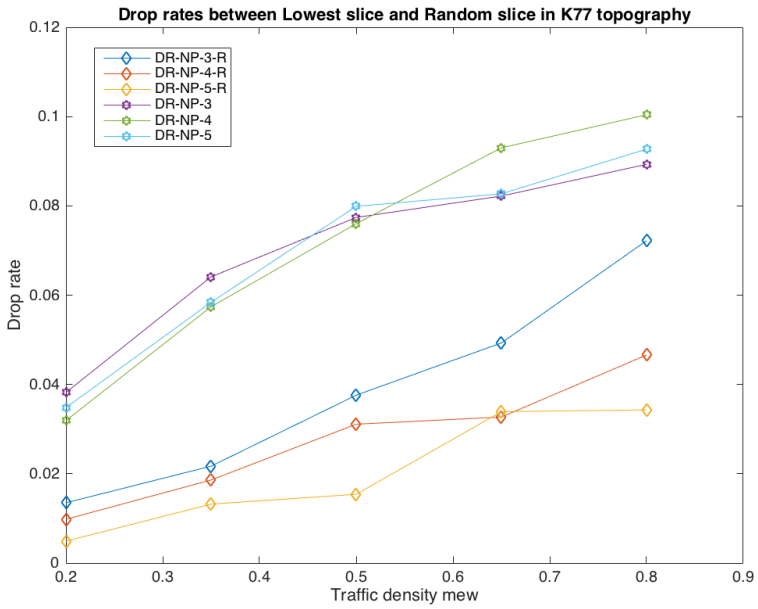

Figure 30: Drop rates comparison lowest slice and random slice in $K_{7,7}$

6.3.2 Random Slice Routing

As mentioned above, the lowest slice routing method has a critical problem that drop dates are significant, because traffic tends to aggregate at low numbered layers. In order to improve QoS by reducing the drop rate, we propose another random slice routing scheme, in which we randomly choose slices instead of preferring low numbered slices. We evaluate its performance by comparing drop rates, blocking rates in comparison to the lowest slice routing scheme.

Figure 30 plots the drop rates in both lowest slice routing and random slice routing. The random slice routing data have an "-R" appended in their notations to be distinguished from lowest slice routing data. The test networks are $K_{7,7}$ networks with 3, 4, or 5 slices which are same as above. As can be seen for each test network, the drop rate in random slice routing is significantly lower than its counterpart in lowest slice routing. So, we can conclude that, by allocating the traffic to the slices evenly, we keep the slice utilization rate

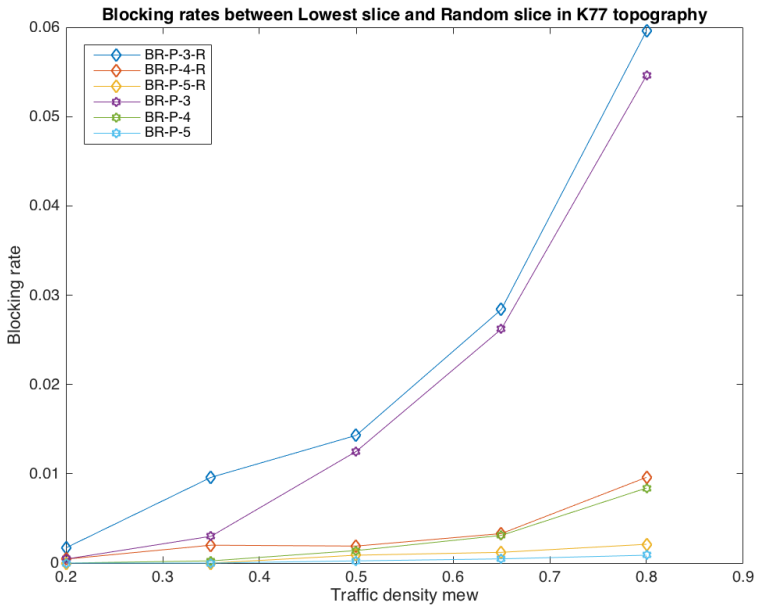

Figure 31: Blocking rates comparison lowest slice and random slice in $K_{7,7}$

for each slice as low as possible and achieved lower drop rate for each network compared to lowest slice routing.

It is also observed that with the increase in slice number, the drop rate slightly decreases in random slice routing. We can deduce that adding more slices will reduce the slice utilization rate at each slice; however the new added slices would produce drops. This observation implies that the number of drops saved by the utilization rate reduction is more than the number of drops introduced by the new slices. Thus we can conclude that using random slice routing can achieve lower drop rates and they can be potentially further decreased by introducing more slices.

Figure 31 plots the blocking rates. The notations are similar to above. It can be observed that for each slice number, the blocking rates in the random slice scheme is slightly higher than its counterpart in the lowest routing mode. It can be concluded that lowest slice routing is slightly better in traffic performance by aggregating them to the lower numbered

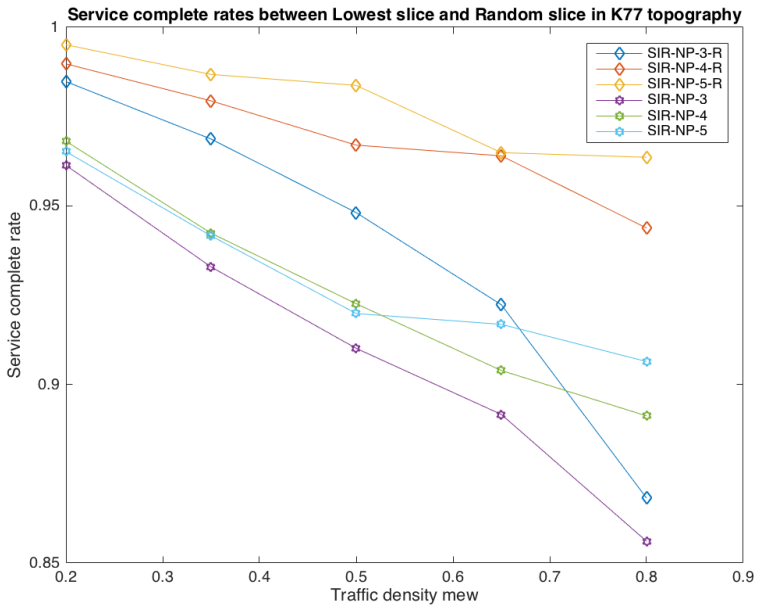

Figure 32: Service complete rates comparison lowest slice and random slice in $K_{7,7}$

slices such that subsequent arriving connections have a higher probability to be routed on a high number slice.

Figure 32 plots the service completion rates, which is essentially the portion of traffic that is not dropped nor blocked. It can be observed that for each network, using random slice routing achieves a higher service completion rate than using lowest slice routing. Combining the result from Figure 30 and Figure 31, we can explain the higher service complete rate for random slice routing because the gain in reduced drop rates overwhelms the loss by increased blocking rate.

We can conclude that the drop rate in a slice has a positive correlation between the slice utilization rate. Their relationship is a curve such that highly utilized slices may produce more excessive drops than mildly used slices. Random slice routing achieves better service completion rate by evenly distributing the connections to reduce the number of dropped connection by preventing highly utilized slices. This is another counter example to the

Benes Conjecture because routing the traffic to the busiest part of network may reduce the performance.

6.4 3-HOP ROUTING MODIFICATION

The previous section addressed the drop rate problem. This section proposes a method to improve blocking rates. It can be observed that Algorithm 1 offers very few path options for inter party connections which are more likely to be blocked. For example, an inter party connection only has one path in each slice, compared to seven in $K_{7,7}$ for an intra party connection. Thus, we propose the 3-hop routing modification to allow more path options for inter party connections in order to improve the blocking rates.

We collect results from 4 sets of simulations for a 3-slice $K_{7,7}$ network with lowest slice routing:

1. Priority mode. Blocking rate (denoted as "BR-P") and drop rate (as "DR-P") are collected.
2. NO priority. Blocking rate (as "BR-NP") is collected.
3. Priority mode with 3-hop modification. Blocking rate (as "BR-P-L3") and drop rate (as "DR-P-L3") are collected.
4. NO priority with 3-hop modification. Blocking rate (as "BR-NP-L3") is collected.

In Figure 33, we observe that 3-hop routing significantly reduces the blocking rate especially when $\rho \leq 0.5$. The blocking rates are close to zero when $\rho \leq 0.5$ and become significant when $\rho \geq 0.65$, in both priority mode and no priority mode. Note that the blocking rate without 3-hop modification reaches the $\rho = 0.65$ value of 3-hop modification at $\rho = 0.35$ and keeps growing with ρ. This demonstrates that the 3-hop routing method has a strong potential in reducing the blocking rate. For the same blocking rate, the routing algorithm would handle nearly twice the traffic with 3-hop modification compared to the original algorithm without it. We can also observe that the service completion rate in no priority mode is higher than its counterparts in priority mode.

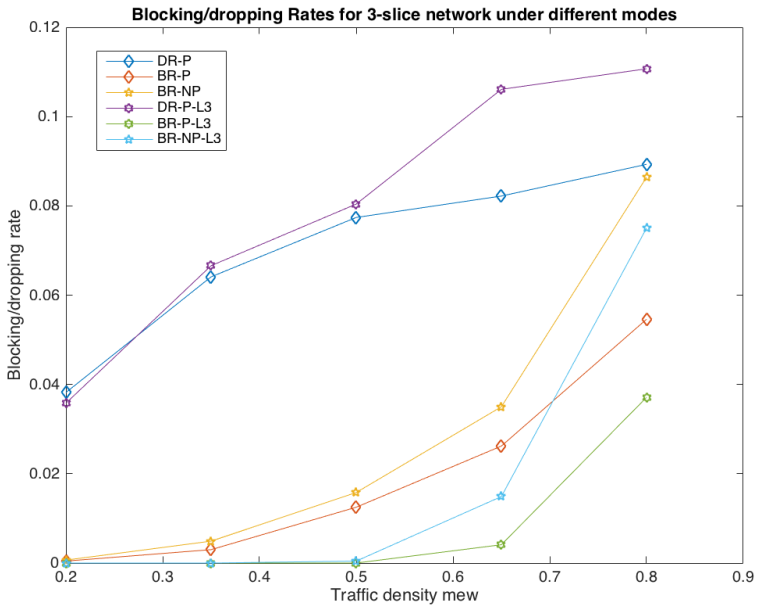

Figure 33: Blocking/Dropping rates in 3-slice $K_{7,7}$ with 3 hop routing

We also noticed that the 3-hop routing does not reduce the drop rate; however it elevates the drop rate significantly at high traffic densities. Even when $\rho \leq 0.5$, the difference is observable. To further investigate into the details of this effect, we recorded the number of dropped connections, for each traffic type, with or without 3-hop routing, in Figure 34.

In Figure 34, we observed that when $\rho \geq 0.35$, the number of intra party connections dropped with 3-hop routing is more than that without 3-hop routing. We can also observe this phenomenon for inter party connections. It looks to be a universal rise of dropped connections in each type.

We may begin the explanation with the fact that 3-hop routing reduces the blocking rate significantly. In other words, it increases the utilization rate as more traffic gets into the network. Thus, the drop counts will increase with the improved slice utilization rates. This explains the universal rise in drop count at high traffic densities. We can conclude that 3-hop routing can universally reduce the blocking rate significantly. It may raise the drop

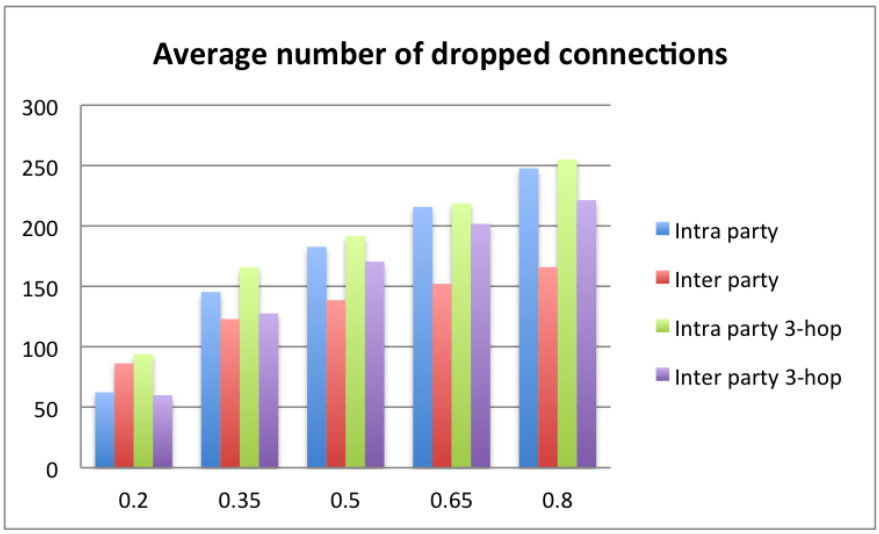

Figure 34: Average number of dropped connections

rate in priority mode at high traffic densities.

6.5 COMPARISON WITH K-SHORTEST PATH ALGORITHM

To further evaluate the performance of this bipartite complete routing scheme, we compare its performance and cost efficiency with the most commonly used algorithm in optical network design—the *K-shortest path and first fit* algorithm. It returns K shortest paths for a certain source-destination pair, and an available path will be picked. The term *First fit* means to set the connection to the lowest numbered slice.

The K-shortest path algorithm is implemented directly on the NSFNet without virtual circuits. In this case, each slice has the original NSFNet topology while each link has one standardized channel in either direction.

Unlike the K-shortest path, the bipartite complete routing algorithm needs to be deployed in a virtual $K_{7,7}$ network, which needs to be deployed in the virtual layer for NSFNet. As

recorded in Table 5.1, a $K_{7,7}$ slice for NSFNet needs 250 directed links, while NSFNet slice (no virtual circuits) uses 42 directed links.

The cases with traffic priorities are no longer studied in this section because we have seen that no priority mode could at least achieve an equal service completion rate to the priority mode throughout previous experiments. Another reason is that the K-shortest path and first fit algorithm do not set priorities and we want to keep a consistent standard.

We compare the performance versus the cost for each algorithm. The performance is evaluated by blocking rate and the cost is evaluated by the number of links in the network. The cost is the total number of links in all slices in the test network. It is the product of the number of links in one slice, and the number of slices. In this way, each $K_{7,7}$ slice is as costly as approximately 6 original NSFNet slices.

6.5.1 Performance Comparison

We begin with a set of simulations evaluating the blocking rates, with the following schemes , the result of which are recorded in Figure 35:

1. 3-slice $K_{7,7}$ network, no priority, with 3-hop routing (denoted as "BR-NP-L3-3") or without (denoted as "BR-NP-3").
2. 4-slice $K_{7,7}$ network, no priority, with 3-hop routing (denoted as "BR-NP-L3-4") or without (denoted as "BR-NP-4").
3. 12-slice, 13-slice, 14-slice, and 15-slice original NSFNet network, no priority, K-shortest path and First Fit (denoted as "KshortFF-12", "KshortFF-13", "KshortFF-14", and "KshortFF-15" respectively).

Before we begin to analyze the result in Figure 35, we would like to report the following facts which are not plotted for a better display. These facts provide a good reference for the number of slices to handle traffic without blocking.

1. A 5-slice $K_{7,7}$ network, no priority, with 3-hop routing ("BR-NP-L3-5") has all zero blocking rates collected in each simulation run. No exception observed.
2. A 18-slice original NSFNet network, no priority, routed by K-shortest path algorithm, has all zero blocking rates collected in each simulation run. No exception observed.

Next, we present Figure 35 to study the performance. Among the group of blocking rates from $K_{7,7}$ networks without 3-hop routing, the 3-slice blocking rate is higher than the 12-slice with K-shortest path routing. The difference in blocking rates at low densities are more significant than that in high densities between the 3-slice and 4-slice $K_{7,7}$ networks. The 4-slice blocking rate is positioned well between the 12-slice and 13-slice network with K-shortest path routing.

This may suggest that, to achieve similar blocking rates, each $K_{7,7}$ slice is similar to approximately more than 3 original NSFNet slices at low and middle traffic densities. The incoherence is that, a 3-slice $K_{7,7}$ network seems to be equivalent to 13 original slices, while the gap between the blocking rates from a 3-slice $K_{7,7}$ network and a 4-slice $K_{7,7}$ network are about the size of a gap of 1 NSFNet slice. This may due to the fact that at high traffic the network is congested and the marginal effect of adding one virtual slice is not as significant as that in low densities.

We also observe that among the group of blocking rates from $K_{7,7}$ networks with 3-hop routing, the blocking rates from the 4-slice $K_{7,7}$ network outperforms all the K-shortest path blocking rates plotted. Similarly, we can estimate that each $K_{7,7}$ network with 3-hop routing would be similar in blocking rate performance to 4 original slices.

While more work is needed for a better approximation to a more precise equivalent number, our work is sufficient to make the statement that $K_{7,7}$ slices, combined with its routing method, is NOT efficient in resource use for achieving the same blocking rate compared to K-shortest path routing, even when the 3-hop modification is used. Our work estimates that a $K_{7,7}$ slice at its best is equivalent to 4 NSFNet slices while achieving similar blocking rate, while using equivalently the resource for 6 NSFNet slices.

It can be concluded that the cost of virtual circuits to make NSFNet into a $K_{7,7}$ virtual topology offsets the benefit of the routing methods. The cost of virtual circuits is related to the physical layer topology. We speculate cost-efficiency would be better if the cost for virtual circuits can be reduced by building the virtual $K_{7,7}$ topology on other physical layer topologies. This problem is further studied and discussed in the next subsection.

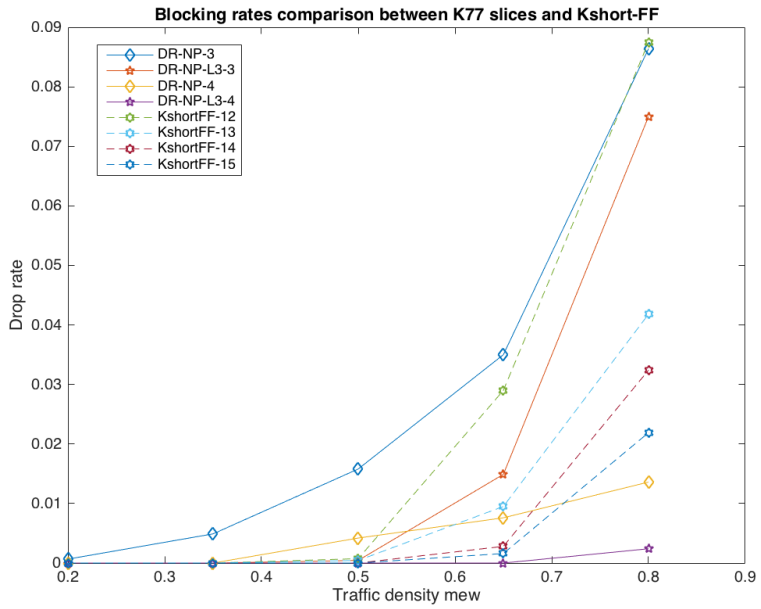

Figure 35: Comparison of $K_{7,7}$ slices routing and K-shortest path routing

6.5.2 Cost Efficiency Evaluation with "Transitional Networks" of NSFNet

The previous subsection suggests that insufficient connectedness in the physical topology may raise the cost of virtual circuits, which may offset the cost-efficiency. Thus we propose to study the effect of physical layer topologies towards the cost of virtual circuits in evaluating the cost-efficiency. We begin with a set of simulations reaching the conclusion that, if the physical layer topology is $K_{7,7}$, our routing method with 3-hop routing out performs K-shortest path routing. Then we evaluate some transitional graphs, which are super graphs to NSFNet, to evaluate their performance in blocking rate with consideration of virtual circuit costs.

In addition, our routing algorithm for the bipartite complete graph along with the 3-hop modification is essentially a topology aware routing algorithm which finds routing paths through a shortcut in which only partial network status is needed, which is granted by the structure of bipartite complete graphs. The advantage of the topology aware routing can be

evaluated by the cost efficiency towards the K-shortest path algorithm. To summarize, this study will provide us an quantitive reference the effect of "sufficiently" connected physical layer topology and the advantage of topology towards a cost-efficient routing.

Suppose the physical layer topology is $K_{7,7}$ and thus no virtual circuit is needed. We can observe that intra party connections have at most 7 paths to choose while inter party connections has up to 43 paths to choose, with 3-hop modification. In K-shortest path routing, each connection have up to K choices; while we set $K = 5$ in our simulation. Based on this fact we could expect that K-shortest-FF algorithm may be beaten.

We further verify this expectation by simulations, the result of which is plotted in Figure 36. Notation "BR-NP-L3" represents the blocking rate with no priority and 3-hop routing in $K_{7,7}$ topology using our proposed algorithm. The notations "KshortFF-K77" means applying the K-shortest path algorithm in a $K_{7,7}$ network. The number of slices in the network is appended to this notation. At the end of a notation, the number of shortest paths computed (K) is appended.

For each network at each traffic density, the 3-hop modification beats the K-shortest path algorithm by producing significantly lower blocking rates. We obtained lower blocking rates than the experiments with $K = 7$ while the experiments with $K = 7$ report lower blocking rates than the experiments with $K = 5$. This confirms our expectation that our algorithm has performance advantage over K-shortest path algorithm if there is no cost for virtual circuits.

From the inefficient solution in Figure 35 to the performance advantage in Figure 36 we can confirm that the cost of virtual circuits plays a critical role to determine the cost-efficiency. It raises an interesting question to find the rendezvous point of the performance advantage and virtual circuit cost. We study the cost efficiency for a series of *transitional graphs*. They are like middle point between NSFNet and $K_{7,7}$ graphs. In other words, they are super graphs of NSFNet and share part of the structure from $K_{7,7}$[4].

This series of graphs can be generated by adding several of the virtual circuits listed in Table 5.1 as physical layer links. We begin by picking up several virtual circuits in the

[4]A super graph of NSFNet is not a subgraph of $K_{7,7}$ because there are triangles in NSFNet in which an edge may connect two vertices of the same color.

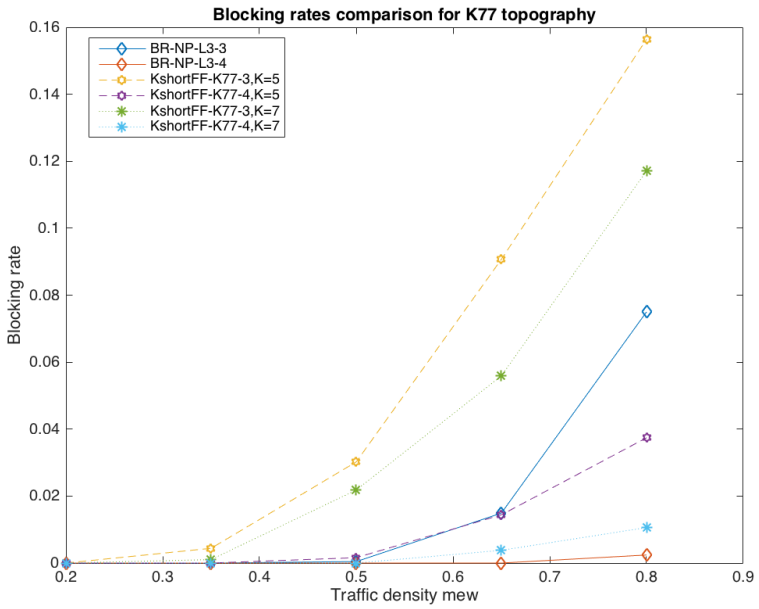

Figure 36: 3-hop modification vs KshortFF in physical layer $K_{7,7}$ topology

table. For each virtual circuit, we connect the source and destination vertices by a physical layer link. That is, this virtual circuit is alternatively reshaped as a physical link in part of the physical layer network. The new physical links may also be used to set up other virtual circuits. These virtual circuits selected to be alternatively implemented by physical links are referred as *selected virtual circuits*. This method not only saves virtual circuits but also provides a better connected physical layer topology to deploy other virtual circuits.

We study 5 transitional graphs which are generated by adding 5, 10, 15, 20, 25 virtual circuits in Table 5.1 to the physical layer network. The selected virtual circuits are randomly picked up from the pool. Note the term "# physical links" in the table is NOT the total number of links in the physical layer. Rather, it is the number of physical links labeled purple in Figure 15, which is the number of links directly used for inter party links in the virtual topology[5] and is referred as l_p. The table records the cost of virtual circuits to set

[5]The non-purple links are not used as physical link but rather as part of virtual circuits. Thus they are not accounted in the number of physical links

Table 9: Link cost of transitional graphs for NSFNet

Topography	# of Selected Virtual circuits	# physical links (l_p)	Virtual circuit cost (l_v)
NSF	0	18	107
NSF+5R1	5	23	79
NSF+10R1	10	28	61
NSF+15R1	15	33	52
NSF+20R1	20	35	37
NSF+25R1	25	43	16

up a virtual $K_{7,7}$ slice for each of the graphs, which is referred as l_v. It can be obtained by solving the RWA problem to set up a virtual $K_{7,7}$ topology for the transitional graphs.

The transitional graphs are shown as follows in the form of an adjacency matrix. The vertex labels are the same as in Figure 15. The new selected virtual circuits are plotted in

bold font. The adjacency matrix for "NSF+5R1" is:

$$\begin{bmatrix}
0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & \mathbf{1} & 0 & 0 & 1 \\
0 & 0 & \mathbf{1} & 0 & 1 & 0 & 0 & 1 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & \mathbf{1} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & \mathbf{1} & 0 & 1 & 0 & 0 & 0 & \mathbf{1} & 0 \\
0 & 0 & 0 & 1 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 1 & \mathbf{1} & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0
\end{bmatrix}$$

The adjacency matrix for "NSF+10R1" is:

$$\begin{bmatrix}
0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & \mathbf{1} & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 & \mathbf{1} & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & \mathbf{1} & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & \mathbf{1} & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} \\
1 & 0 & 0 & \mathbf{1} & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & \mathbf{1} & 0 \\
0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & \mathbf{1} \\
\mathbf{1} & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & \mathbf{1} & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 1 & \mathbf{1} & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 1 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & \mathbf{1} & 0 & \mathbf{1} & 0 & 0 & 1 & 1 & 0
\end{bmatrix}$$

The adjacency matrix for "NSF+15R1" is:

$$\begin{bmatrix}
0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & \mathbf{1} & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & \mathbf{1} & 0 \\
1 & 1 & 0 & \mathbf{1} & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & \mathbf{1} & 0 & 1 & 0 & 0 & \mathbf{1} & 0 & \mathbf{1} & 1 & 0 & 0 & \mathbf{1} \\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\
0 & \mathbf{1} & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & \mathbf{1} & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & \mathbf{1} & \mathbf{1} & 0 & 0 & \mathbf{1} \\
1 & 0 & 0 & \mathbf{1} & 0 & 0 & 1 & 0 & 1 & 0 & 0 & \mathbf{1} & 0 & 0 \\
0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & \mathbf{1} & 1 & 1 & 0 \\
0 & 0 & 0 & \mathbf{1} & 0 & 1 & \mathbf{1} & 0 & 1 & 0 & 0 & \mathbf{1} & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & \mathbf{1} & 1 & \mathbf{1} & 1 & 0 & 0 & 1 \\
0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & \mathbf{1} & 0 & 1 & \mathbf{1} & 0 & 0 & 0 & 0 & 1 & 1 & 0
\end{bmatrix}$$

The adjacency matrix for "NSF+20R1" is:

$$\begin{bmatrix}
0 & 1 & 1 & 0 & \mathbf{1} & 0 & 0 & 1 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & 0 & 0 & \mathbf{1} & \mathbf{1} & 0 \\
1 & 1 & 0 & \mathbf{1} & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\
0 & 1 & \mathbf{1} & 0 & 1 & 0 & 0 & 0 & 0 & \mathbf{1} & 1 & 0 & 0 & \mathbf{1} \\
\mathbf{1} & 0 & 0 & 1 & 0 & 1 & 1 & 0 & \mathbf{1} & 0 & 0 & \mathbf{1} & 0 & 0 \\
0 & \mathbf{1} & 1 & 0 & 1 & 0 & 0 & \mathbf{1} & 0 & 1 & \mathbf{1} & 0 & 0 & 1 \\
0 & \mathbf{1} & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & \mathbf{1} & 0 & 0 & \mathbf{1} \\
1 & 0 & 0 & 0 & 0 & \mathbf{1} & 1 & 0 & 1 & 0 & 0 & \mathbf{1} & \mathbf{1} & 0 \\
0 & \mathbf{1} & 0 & 0 & \mathbf{1} & 0 & 0 & 1 & 0 & 1 & \mathbf{1} & 1 & 1 & 0 \\
0 & 0 & 0 & \mathbf{1} & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{1} & 0 & 0 & 1 & 0 & \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & 0 & 0 & 1 & 1 & 0 \\
0 & \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & 0 & 0 & \mathbf{1} & 1 & 0 & 1 & 0 & 0 & 1 \\
0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 1 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & \mathbf{1} & 0 & 1 & \mathbf{1} & 0 & 0 & 0 & 0 & 1 & 1 & 0
\end{bmatrix}$$

The adjacency matrix for "NSF+25R1" is:

$$\begin{bmatrix}
0 & 1 & 1 & 0 & \mathbf{1} & 0 & 0 & 1 & 0 & \mathbf{1} & 0 & 0 & 0 & \mathbf{1} \\
1 & 0 & 1 & 1 & 0 & \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & 0 & 0 & \mathbf{1} & \mathbf{1} & 0 \\
1 & 1 & 0 & \mathbf{1} & 0 & 1 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & \mathbf{1} & 0 & 1 & 0 & 0 & 0 & 0 & \mathbf{1} & 1 & 0 & 0 & \mathbf{1} \\
\mathbf{1} & 0 & 0 & 1 & 0 & 1 & 1 & 0 & \mathbf{1} & 0 & 0 & \mathbf{1} & \mathbf{1} & 0 \\
0 & \mathbf{1} & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & \mathbf{1} & 0 & 0 & 1 \\
0 & \mathbf{1} & 0 & 0 & 1 & 0 & 0 & 1 & 0 & \mathbf{1} & \mathbf{1} & 0 & 0 & \mathbf{1} \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & \mathbf{1} & \mathbf{1} & 0 \\
0 & \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & 0 & 0 & 1 & 0 & 1 & \mathbf{1} & 1 & 1 & \mathbf{1} \\
\mathbf{1} & 0 & 0 & \mathbf{1} & 0 & 1 & \mathbf{1} & 0 & 1 & 0 & 0 & \mathbf{1} & \mathbf{1} & 0 \\
0 & 0 & 0 & 1 & 0 & \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & 0 & 0 & 1 & 1 & 0 \\
0 & \mathbf{1} & 0 & 0 & \mathbf{1} & 0 & 0 & \mathbf{1} & 1 & \mathbf{1} & 1 & 0 & 0 & 1 \\
0 & \mathbf{1} & 0 & 0 & \mathbf{1} & 0 & 0 & \mathbf{1} & 1 & \mathbf{1} & 1 & 0 & 0 & 1 \\
\mathbf{1} & 0 & 0 & \mathbf{1} & 0 & 1 & \mathbf{1} & 0 & \mathbf{1} & 0 & 0 & 1 & 1 & 0
\end{bmatrix}$$

We plot the blocking rate in log scale (y-axis) vs the total number of links in the network (x-axis), while ρ is set as a constant value. The total number of links can be computed by $N_s \times (l_p + l_v)$, where N_s is the number of $K_{7,7}$ slices in the network. The sum of l_p and l_v is essentially the total number of links necessary for establishing a $K_{7,7}$ slice.

Figure 37 and Figure 38 each have 6 subgraphs, in which the blocking rate vs virtual circuit cost is plotted for our algorithm (solid lines), and the K-shortest path algorithm (dashed lines) respectively. The transitional graph for each subplot is briefly recorded in the notation of the dashed line. For each plot of our algorithm, the set of blocking rates are same while total link cost decreases with the increase of the number of links in transitional graphs. So we can observe the line of the virtual $K_{7,7}$ line is leaning left with the scale increase of links in physical layer topologies.

The blocking rates for the K-shortest path algorithm are obtained from the simulation while the link cost is the number of links in the physical layer. We observed that the lines also have a trend of moving left in the subplots when more physical layer links are added.

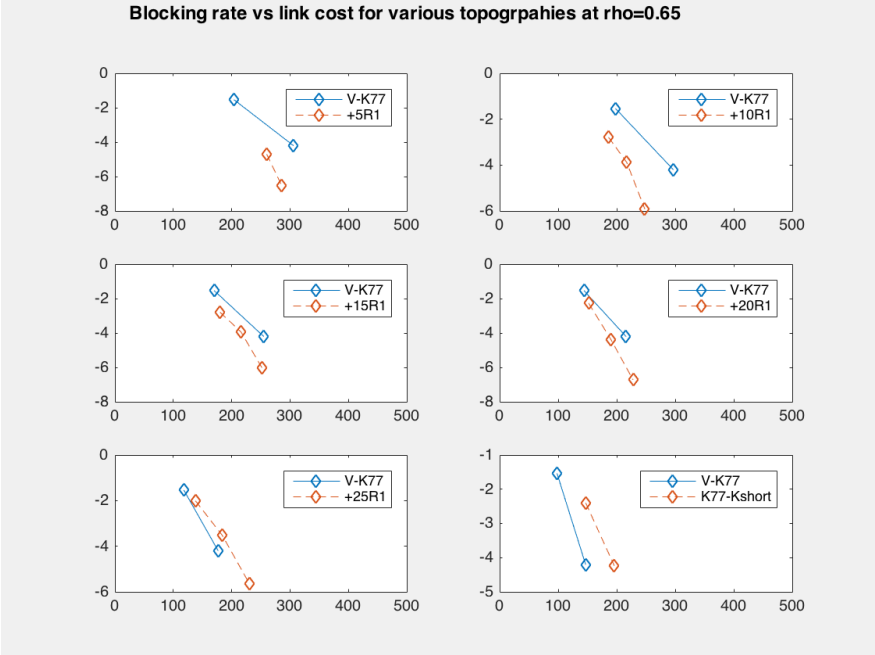

Figure 37: Blocking rate vs link cost at $\rho = 0.65$

This indicates that we can achieve lower blocking rates and reduce the overall link cost by improving the connectedness of the physical layer network while guaranteeing the same blocking rate.

By comparing the trends of both lines among the transitional graphs, we can see that the line for our algorithm moves faster than the K-shortest-path lines. For *NSFNet*, the line is at the right of that of the K-shortest-path algorithm, which means that it uses more links while achieving the same blocking rate. For *NSF+25R1*, which is a super graph of *NSFNet* by 25 more links, the line of our algorithm is at the left of its counterpart of K-shortest-path algorithm. This means our algorithm can achieve the same blocking rate at a lower cost for a better connected network. This matches our observation in the previous chapter.

The rendezvous point at the two lines can be estimated at somewhere between graph *NSF+20R1* and *NSF+25R1* when $\rho = 0.65$, and at *NSF+25R1* when $\rho = 0.8$. Despite that we need to add more than 20 lines to *NSFNet* to achieve this result, our bipartite complete

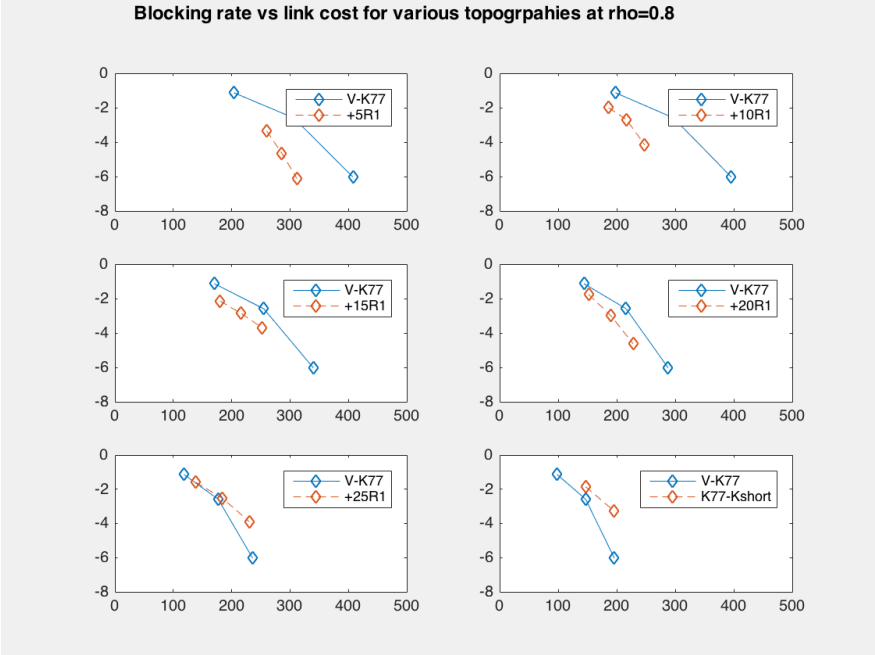

Figure 38: Blocking rate vs link cost at $\rho = 0.8$

routing algorithm still demonstrates its advantage at sufficiently connected graphs over the K-shortest path algorithm.

We could conclude that, if we develop a sufficiently connected graph based on NSFNet, implementing a virtual bipartite complete graph with Algorithm 1 has better blocking rate than the K-shortest-path algorithms at the same link cost. It demonstrates that applying the topology aware routing method has better cost-efficiency than the general routing method when the physical layer topography is close to a bipartite complete graph.

6.5.3 Cost Efficiency Evaluation of SprintNet

In this subsection we conduct the same experiment as above section, but with the SprintNet structure, which has 11 nodes distributed nationwide connected by 17 links. Its average degree per node is 1.54, which is very close to NSFNet (1.5). Similarly, a virtual $K_{5,6}$ topology can be set up in the SprintNet network by adding virtual circuits. It's topology and the

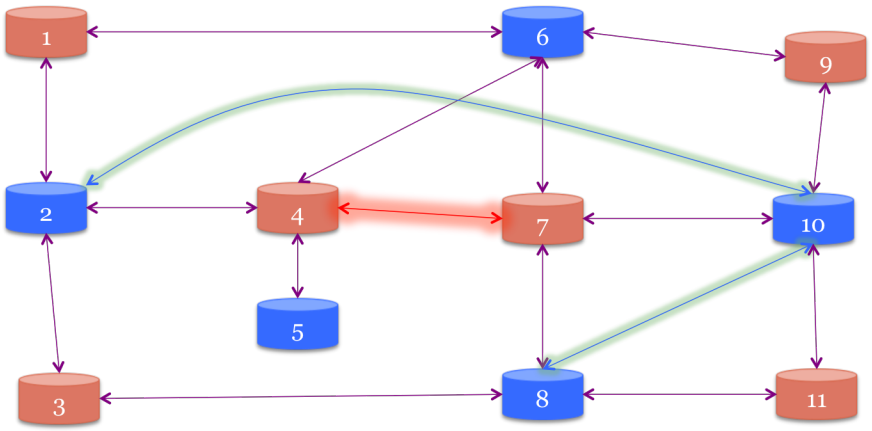

Figure 39: SprintNet Topology and Node parties

node parties in corresponding $K_{5,6}$ topology is illustrated by Figure 39. The 11 nodes in the network are divided into two parties. The blue party has five nodes while the red party has six nodes. Among the 17 links 14 of them are inter-party links, and similar to NSFNet, 3 of them are intra party links. Since graph $K_{5,6}$ has 30 links, 16 virtual circuits need to be set up. We can achieve a good solution of their paths along the network using the MILP methods mentioned in Section 2.3.

Table 10 records the virtual circuit paths and their colors. A total of six colors are needed as the virtual circuits need five and the physically connected links need one. It is about two thirds of its counterpart of NSFNet. It costs 55 links (two ways) to set up the virtual $K_{5,6}$ topology in SprintNet.

We compare the cost efficiency of bipartite complete routing with 3-hop modification and the K-shortest path algorithm for SprintNet. The bipartite complete virtual topology is constructed according to Table 10. We set $K = 7$ for the K-shortest path algorithm.

Figure 40 plots the blocking rate and the number of links needed for both algorithms. It shows that bipartite complete routing uses slightly more links to achieve the same blocking rate. Similarly, the cost of virtual circuits offsets the structure advantage.

We propose 3 transitional graphs, which are super graphs of the SprintNet, to estimate

Table 10: Virtual Circuits Setup for SprintNet

VC	Path	Color	VC	Path	Color	VC	Path	Color
$2 \leftrightarrow 7$	[2,10,7]	2	$2 \leftrightarrow 9$	[2,10,9]	4	$2 \leftrightarrow 11$	[2,10,11]	5
$1 \leftrightarrow 5$	[1,2,4,5]	2	$2 \leftrightarrow 5$	[3,2,4,5]	4	$5 \leftrightarrow 7$	[5,4,6,7]	5
$5 \leftrightarrow 9$	[5,4,6,9]	3	$5 \leftrightarrow 11$	[5,4,7,10,11]	1	$3 \leftrightarrow 6$	[3,2,1,6]	1
$6 \leftrightarrow 11$	[6,7,8,11]	1	$1 \leftrightarrow 8$	[1,6,7,8]	3	$4 \leftrightarrow 8$	[4,7,8]	2
$8 \leftrightarrow 9$	[8,10,9]	2	$1 \leftrightarrow 10$	[1,2,10]	3	$3 \leftrightarrow 10$	[3,8,10]	1
$4 \leftrightarrow 10$	[4,2,10]	1						

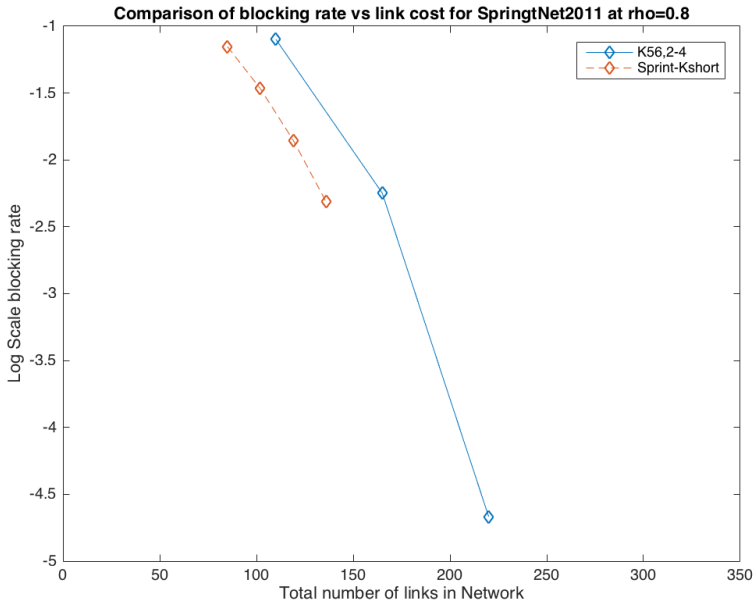

Figure 40: Cost Efficiency for SprintNet topology

Table 11: Virtual Circuits Setup for middle stage graphs of SprintNet

Graphs	Virtual Circuits			
SprintNet+4R1	$2 \leftrightarrow 9$	$2 \leftrightarrow 11$	$3 \leftrightarrow 6$	$1 \leftrightarrow 8$
SprintNet+8R1	$2 \leftrightarrow 9$	$2 \leftrightarrow 11$	$5 \leftrightarrow 7$	$5 \leftrightarrow 9$
	$5 \leftrightarrow 11$	$3 \leftrightarrow 6$	$8 \leftrightarrow 9$	$3 \leftrightarrow 10$
SprintNet+8R1	$2 \leftrightarrow 7$	$2 \leftrightarrow 9$	$2 \leftrightarrow 11$	$1 \leftrightarrow 5$
	$5 \leftrightarrow 7$	$5 \leftrightarrow 9$	$3 \leftrightarrow 6$	$1 \leftrightarrow 8$
	$4 \leftrightarrow 8$	$8 \leftrightarrow 9$	$3 \leftrightarrow 10$	$4 \leftrightarrow 10$

the rendezvous point of the cost efficiency of two algorithms. The transitional graphs are generated by adding 4, 8, 12 virtual links in Table 10 in the physical layer. For instance, by adding link $(2, 7)$ to the physical layer graph, we don't need to implement virtual circuit $2 \leftrightarrow 7$ anymore to reach $K_{5,6}$. They are denoted as *SprintNet+4R1*, *SprintNet+8R1*, and *SprintNet+12R1* respectively.

For the convenience of presentation, we only list the virtual circuits in Table 11 that are directly added to the physical layer for each transitional graph as the number of virtual circuits for the SprintNet is not huge. We plot the blocking rate vs link cost at $\rho = 0.65$ and $\rho = 0.8$ in Figure 41 and Figure 42 respectively. Similarly the topology in each subplot is denoted by an abbreviation beginning with a "+" symbol. The solid line represents the blocking rate using virtual $K_{5,6}$ topology and bipartite complete routing with 3-hop modifications. The dashed line represents the blocking rates for the K-shortest path algorithm.

We can observe in Figure 41 that the rendezvous point is estimated at around *SprintNet+8R1*, in which the two lines cross over. The subplot for *SprintNet+8R1* in the figure shows that the K-shortest path algorithm could achieve better blocking rates when there are fewer than 150 links. However, our bipartite complete routing method achieves lower blocking rates as we have more links to use. For graph *SprintNet+8R1*, which adds more virtual circuits to the physical layer, our routing method achieves the same blocking rate

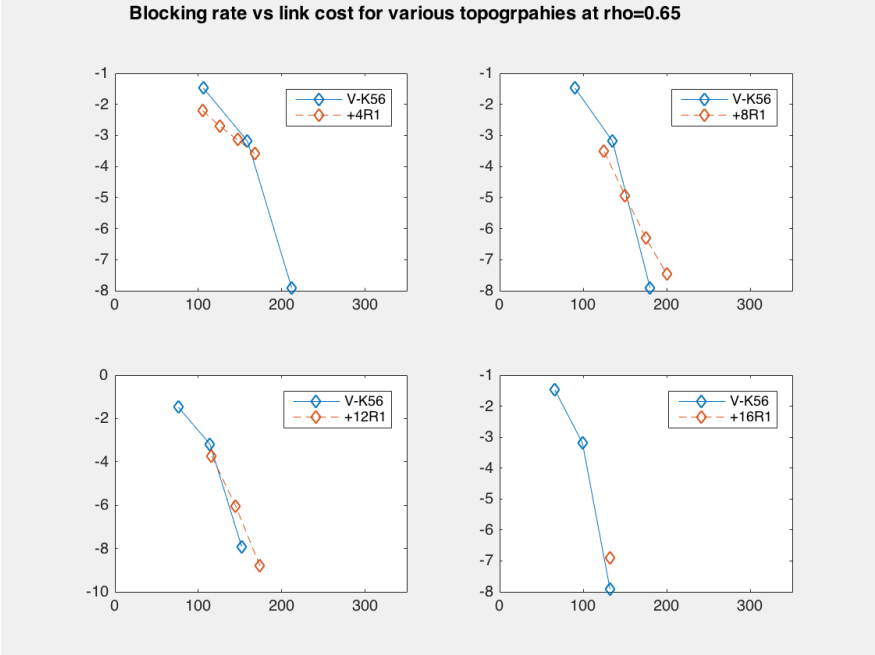

Figure 41: Blocking rate vs link cost at $\rho = 0.65$ for SprintNet

using fewer links on average compared to the K-shortest path algorithm.

We can observe that the rendezvous point shifted toward better connected graphs when $\rho = 0.8$ as shown in Figure 42. The similar crossover observed in *SprintNet+8R1* when $\rho = 0.6$ is now observed in *SprintNet+12R1* instead. The bipartite complete graph becomes overall more efficient in *SprintNet+16R1*, which is virtually a bipartite complete structure[6]. It implies that the K-shortest-path algorithm is likely to achieve a slightly better blocking rate per link at higher traffic densities.

In Figure 41 and Figure 42 we observed similar patterns in Figure 37 and Figure 38. When the physical layer network is less connected, the K-shortest path algorithm achieves a better cost efficiency. By improving them, we can reduce the overall link cost for the virtual circuits. Despite that the K-shortest path algorithm would also perform better with the improvement, both sets of experiments show that the bipartite complete routing algorithm

[6]To be more precise, it is a super graph of $K_{5,6}$ as it has three links that are adjacent to same color vertices.

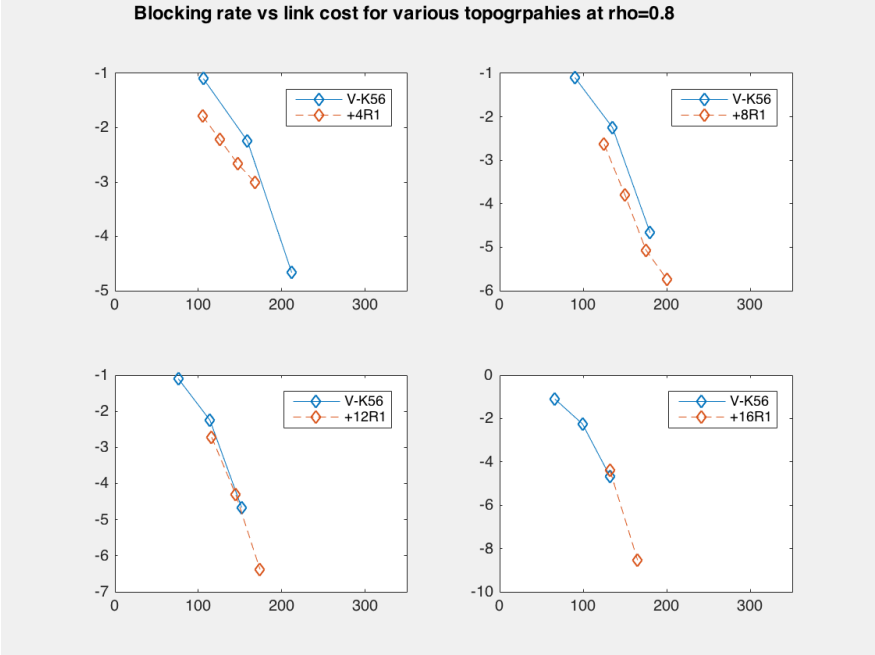

Figure 42: Blocking rate vs link cost at $\rho = 0.8$ for SprintNet

gains more cost efficiency than the K-shortest path algorithm. It achieves a better per link blocking rate than the K-shortest path algorithm does at a middle stage graph. In addition, we observed that the rendezvous point of their performance tends to move to better connected physical layer networks when ρ is large.

6.5.4 Conclusion

We compared the cost efficiency of the bipartite complete routing method and the K-shortest path routing algorithm in previous two subsections. The cost efficiency is evaluated by the number of links needed to achieve the blocking rate. This experiment is conducted for NSFNet and SprintNet. A set of transitional graphs are generated for each network to better evaluate the trade-off between the power of the bipartite complete topology aware routing algorithm and the cost for virtual links.

We can conclude that the bipartite complete routing method has a natural advantage over

the K-shortest path algorithm by being distributable and having better routing capability. However, the links needed to set up the virtual circuits have significant cost that might offset the advantage in both network structures due to their insufficient connectedness.

Nevertheless, the cost for virtual circuits can be reduced if we improve the connectedness of the network. The study of both networks demonstrated that the bipartite complete routing method can achieve the same blocking rate with fewer links than the K-shortest path algorithm does if we add approximately half of the virtual circuits needed for a virtually bipartite complete topology. It is also found that the bipartite complete routing method could achieve a slightly better cost efficiency when ρ is small than when ρ is large.

Based on these results, we could conclude that the bipartite complete routing method has a card to play to be the core of a distributed network framework when the connectedness of the network is sufficient. Note that it can be implemented in a distributed system and provides a quick response, which would be very beneficial in real implementations.

7.0 CONCLUSION

This dissertation has presented a systematic design paradigm for a general WDM mesh network, which includes:

1. Identify that bipartite complete graphs are RNB and WSNB, which depends on the routing algorithm.
2. Propose a RNB routing algorithm and a WSNB routing algorithm for bipartite complete graphs. The WSNB problem is a counter example to the Benes Conjecture, which proposed routing traffic to the busiest part of the network.
3. Propose implementation frameworks to deploy the non-blocking graphs to NSFNet with system diagrams and current available devices. Both frameworks could bypass IP routing.
4. Evaluate the traffic handling power of the redundant links in bipartite complete graphs for Poisson traffic.
5. Evaluate and compare cost efficiency of the bipartite complete routing framework against prevailing algorithms.

The non-blocking design paradigm proposed in this paper has four components: non-blocking graphs, algorithm, system infrastructure, and performance analysis. The discussion and conclusion for each of them are separately presented in following sections in this chapter.

7.1 NON-BLOCKING GRAPHS

This paper presents two novel theorems for non-blocking connected mesh network structures. The first theorem (Theorem 4.2.28) indicates that bipartite complete graphs are WSNB un-

der Algorithm 2. The WSNB property has never been discussed for meshed networks in the related literature. The second theorem (Theorem 4.2.31) indicates that they are also RNB under Algorithm 1, which is a reduced version of Algorithm 2, as long as proper rearrangement methods are provided.

This result shows that we don't need a complete graph or nearly complete graph to obtain non-blocking properties. While a complete graph needs $\frac{n(n-1)}{2}$ links, where $n = |V|$, a bipartite complete graph at most uses approximately $\frac{n^2}{4}$ links, which is only approximately half of that of a complete graph.

Among the results, we would like to specify that $K_{1,t}$ has non-blocking properties and has the least number of links. In another words, it is the non-blocking graph with least number of links among the graphs with $t+1$ vertices. This extreme case implies that $K_{1,t}$ is the RNB graph with least number of links given the number of vertices. We also observe that $K_{r,t}$ has redundant links for every traffic pattern when $r, t \geq 2$. This implies that we need an extra number of links to guarantee the same non-blocking property if we want to design the network in a more distributed manner.

7.2 ALGORITHMS

This dissertation introduced two non-blocking guaranteed topography aware routing algorithms for bipartite complete graphs $K_{r,t}$. Taking advantage of the graph structure of $K_{r,t}$, we can design topography aware routing algorithms by first dividing the connections in a traffic pattern into intra party connections and inter party connections. Both algorithms route inter party connections directly and add one hop for intra party connections. The two algorithms are different in the way they select the hop vertex for intra party connections and we are surprised to see that they lead to different non-blocking properties.

In our investigation of a proper hop selection method to guarantee non-blocking, we observed that routing intra party connections to the most busiest hop vertex that can still handle it, as stated in the Benes Conjecture, will lead to a need rearrangement scenario

which forfeits the WSNB property. Thus, the WSNB routing algorithm could be a counter example for the Benes Conjecture. A generalized statement of this fact is proposed and proven in Proposition 2.4.6.

Both algorithms only return the shortest path (s) between the source and destination. They are not implemented to find every path possible at this time, in another words, therefore, not perfect algorithms. This suggests that, by taking advantage of the graph structure feature, imperfect routing algorithms are sufficient to guarantee non-blocking. We also observe that by perfect routing, we found that a subgraph of $K_{2,4}$ is also RNB. We proved link removal tolerance in $K_{2,t}$ in Lemma 4.2.34.

The impact of this algorithm toward non-blocking graph structures is great. It can be inferred from our observation that a better routing algorithm could make smaller scale graphs non-blocking. However, the algorithm is usually more complicated and requires global status of the network. Status propagation packets are liable to delays and error which may hamper the outcome of the routing algorithm. In our work we prefer to a simple online algorithm as long as it could guarantee non-blocking and preclude the complex implementation issues. It is a very interesting question to investigate the performance vs number of status trade off for the algorithm.

7.3 SYSTEM INFRASTRUCTURE

This dissertation discussed how to establish a bipartite complete graph based on NSFNet. We computed a virtual circuit scheme which needs 9 channels, either in wavelengths or in spatial fibers. We further discussed the system infrastructure for a node in such a system. We found that we can bypass the IP routers if we could label the vertices in the network because the routing algorithm takes advantage of the topography of $K_{r,t}$. We can also bypass the wavelength converters in the system if we use the spatial framework.

7.4 PERFORMANCE ANALYSIS FOR SPARE CAPACITY

We observed that some bipartite complete graphs have redundant links which may provide extra capacity to handle traffic. To evaluate this spare capacity we developed a simulation framework using Matlab. We found the following facts from a series of simulations:

1. Priority mode has similar service completion rate than no priority mode. With 3-hop modification, priority mode has lower service completion rate than no priority mode.
2. The number of drops occuring in a slice is positively correlated to the slice utilization rate.
3. Inter party connections are more likely to be dropped due to limited path options. Enabling 3-hop routing would achieve similar drop statistics to intra party connections.
4. The simulation result can be partially verified by queue theory.

Based on these facts, we further investigated multiple slice networks and compared the cost-efficiency of our design framework with the K-shortest-path algorithm. We concluded that our algorithm has advantages over the K-shortest-path algorithm; however the virtual circuit cost to make NSFNet into $K_{7,7}$ offsets this advantage.

By further evaluations we found out that we can add a significant number of links to NSFNet, and SprintNet, to reduce the cost of virtual circuits and achieve the rendezvous point such that our algorithm begins to obtain the same blocking rate using fewer links than the K-shortest path algorithm does. This demonstrates the potential in efficient routing of our algorithm by taking advantage of the topography. It also suggests that a better topography based graph and routing method pair needs to be found to achieve better performance.

7.5 FUTURE WORK

This non-blocking paradigm proposed by this paper is brand new and can be extended in several directions. For the space capacity evaluation section, for example, we can extend the cost efficiency analysis to multiple bipartite complete graphs and their transitional graphs for

a more comprehensive evaluation of the traffic potential in $K_{r,t}$. It is equivalently promising to study the message exchange process of the routing algorithm and evaluate its effect on the promptness and correctness of the routing process and the network orchestration.

In the graph theory scope, it is also very promising to study a better RNB graph and routing algorithm pair such that we can achieve the RNB property with a limited number of virtual circuits for a real network. Note that bipartite complete graphs have the WSNB property. We may speculate that there will be RNB graphs at a smaller scale if we use a better algorithm. If there is one such algorithm, it would be very interesting to evaluate the degree of distribution of the new algorithm.

In addition, it is very interesting to investigate new topography features that could be used to improve the routing method while guaranteeing a non-blocking property, or some other similar properties. A new topography feature, if there exists, would be the corner stone of the better algorithm just mentioned.

To further improve the performance of the bipartite complete routing, we can modify the routing protocol such that it avoids cycles in the physical layer. An intra party connection takes a length-2 path in the virtual layer. If the two virtual links in the virtual path contain cycles in the physical layer, in which the traffic comes back and forth, but not noticed by the virtual layer, telecom resources are wasted. It is very promising to improve the virtual layer routing algorithm which would prevent or eliminate the cycles and thus make better use of physical layer links. This technique requires extra work to properly manage the virtual layer and physical layer links.

Last but not the least, the fault-tolerance property of a network structure is worth in depth discussion. According to Remark 4.2.32, we can deduce that $K_{r,t}$ is not fault-tolerant to link failures if we use either Algorithm 1 or Algorithm 2. Thus, it is worthy to investigate node failure tolerances, and, if we extend it, the relationship between non-blocking network structures, and fault tolerance properties.

APPENDIX

PERMUTATION TRAFFIC SET

This chapter introduces the deduction process to model real network traffic by permutation traffic sets and demonstrates why the traffic patterns in a permutation traffic set can be mapped to permutations. We begin with a general formulation of a network and traffic and then present step by step analysis and discussions, which then lead to the topic of permutation traffic sets.

A.1 A GENERAL NETWORK

A network is essentially a set of nodes connected by links. A node is similar to a vertex in a graph and provides internet access to its subscribers. Links connect nodes and carry traffic. The structure of a network is the same as a graph if we consider nodes as vertices and links as edges. The terms are used as the following criteria: when referring a network in the telecom concept, we use "node" and "link"; when referring in math concept, we use "vertices" and "edge".

Figure 43 illustrates a simple example network. Two nodes in the network are explicitly plotted as S and D while the remaining nodes and links are abstractly represented by the cloud labeled "internet topography". It implies all the nodes plus S and D are somehow connected by the links.

Consider a traffic stream from S_1 to D_1 in Figure 43. The traffic is first sent to node S.

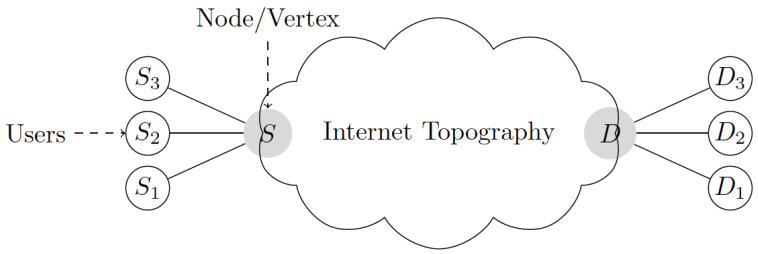

Figure 43: Illustration of a General Telecommunication Network

The routing algorithm computes an available path from S to D to handle the traffic. The network then routes the traffic along the path. Node D, as the destination node, forwards the data directly to subscriber D_1.

In telecom networks, each link has a link capacity (a certain amount of telecom resource), which is usually represented by bandwidth. Bandwidth can be represented by an interval (w_l, w_u), $w_l, w_u \geq 0$, or a real value $w = w_u - w_l$ for the size of the interval. Transmission of a traffic stream consumes bandwidth. Denote the bandwidth cost for traffic stream i is a positive real value w_i.

It raises a tricky question to allocate the bandwidth when there are multiple different traffic streams sharing the same link. To simplify this problem, a standard channel with bandwidth w_c is introduced. The bandwidth of a particular link can be divided into several channels, each of which has identical bandwidth w_c. Thus, the number of channels can be alternatively used to represent the link resources and the cost for traffic streams.

We define that each channel works in one direction while a link works bidirectional (full duplex). We further assume that a link has the same bandwidth in either direction. A link can thus be seen as several pair of channels working in different directions. The total capacity of the link can be represented by the number of pairs multiples of w_c.

Remark A.1.1. *In the following contents in this paper, we may use a vector (v_j, v_j) to represent both a link or a channel in the link. When referring a link it is undirected. Otherwise*

it is directed. To avoid ambiguity, the meaning of the vector will be specified in the context, if not directly in the sentence.

A telecom network can be modeled as a simple graph with weighed vertices and weighted links as in Definition A.1.2.

Definition A.1.2. *A general network can be modeled as a tuple* $N = \{G(V, E), t(v), c(e)\}$ *where* G *is a simple graph,* V *represents the set of vertices or nodes in the network, and* E *is the set of edges in the network. Function* $t(v)$ *is the number of users at vertex* v *and function* $c(e)$ *represents the number channel pairs in a link* $e \in E(G)$*.*

Graph G is called an *underlying simple graph* of network N and represents the network structure. The term G or $G(N)$ is used to refer this underlying simple graph in following contents. Function $t(v)$ represents the users at a node; function $c(e)$ represents the bandwidth of a link in the network. Based on this network model, we can proceed to discuss the traffic in this network.

A.2 TRAFFIC MODEL

Users in real networks subscribes different services from the internet provider, and have diverse traffic. To simplify the problem, we propose two assumptions for the traffic.

1. Each user is identical and lease one pair of standard to transmit/receive data via network.
2. At a particular instant one user only send traffic to at maximum one other user in the network. Similarly, each user can only receive from one another user. A user can send and receive simultaneously.

These assumptions appear strong however they do not lose generality. We explain the generality issues in Remark A.2.1.

Remark A.2.1. *A challenge to assumption 1 may rise as there may be some users asking for more bandwidth than others in the real scenario. WLOG, a "large" user can be alternatively*

modeled as multiple "standard" users such that the total channels/bandwidth are equivalent.

A challenge to assumption 2 may rise that a user could split its traffic to multiple destinations simultaneously. The traffic of an identical user may not fit into this model. However the aggregated traffic of a node may be modeled, as multiple users sending to multiple destinations according to its statistic.

Consider an easy case in which all the destination users are served by one node. Let the network be the one plotted in Figure 43 where users S_1, S_2, S_3 are served by node S. Each of them is splitting its traffic somehow to D_1, D_2, D_3 respectively. As all the destinations are served by one node D, no matter how the traffic is split, Node S needs to reserve 3 channels along the path from S to D in the network, which is equivalent to that there are three users S_i, each of which is leasing a line to D_i respectively, $i \in \{1, 2, 3\}$.

Even if the destination users are served by different nodes, the number of channels to be reserved for each destination can be decided according to the amount of the aggregated traffic to each destination. Similarly, we can reserve the channels as it looks like a number of users are dedicatedly leasing these channels. The granularity issues do exist however its effect is limited and thus not further discussed.

Thus, we conclude that these two assumptions are reasonable in analyzing traffic in this network model.

At this point, we can define connections, traffic patterns and related concepts. Fortunately, we can apply Definition 2.4.2, 2.4.3, and 2.4.5 here without losing generality. Note that the requirement of "an applicable traffic pattern" (Proposition 2.4.6) needs to be rewritten as:

Proposition A.2.2. *A traffic pattern X is applicable to network $N = \{G(V, E), t(v), c(e)\}$ if and only if for any $v \in G$ there is $\eta_s(X, v) \leq t(v)$, and $\eta_d(X, v) \leq t(v)$.*

We present an example of the connections in a traffic pattern in a general network with a simple network structure.

Example A.2.3. *Consider the 4-vertices example network in Figure 45. An attempt of node A to send traffic to node C can be represented by a connection vector (A, C).*

Suppose, at a specific instant in the example network, A is sending to B, B is sending to

C, and C is sending to A. Then the traffic pattern at this instant is a collection of them and can be written as $X = \{(A, B), (B, C), (C, A)\}$.

The term *traffic pattern* is widely used in computer science and engineering literature and is defined in a variety of ways. It may be a description of several statistical features for a flow model. For example, in [55], a traffic pattern represents an Markovian model, which depicts how traffic status and change rates. In addition, specifically in the study of mobile ad-hoc network studies [56], a traffic pattern is denoted by a transmission protocol, which represents the traffic characteristic. What's more, in some miscellaneous network design studies [57], a traffic pattern is defined as a function for the aggregated traffic volume of time. For example, a traffic pattern may be a periodical function representing the traffic volume in 24 hours. Last but not the least, a traffic pattern can be defined as a matrix as in [58], representing the traffic volume between each pairwise set of vertices in the network, which is used for traffic engineering and network design analysis.

Our definition is closest to the last definition above. Instead of a matrix, we define a traffic pattern in the form of a collection of connection vectors, as in Definition 2.4.2. It is sufficient to describe the dynamic traffic scenarios in a general network defined in Definition A.1.2.

In the stochastic process point of view, each traffic pattern can be interpreted as a *state* of the network. The change of the state is triggered by the arrival or disconnection of connections. For example, consider a simple two nodes network and each node serves one user. There are only four applicable traffic patterns shown as following. Their state transition relationship is shown in Figure 44.

- $X_0 = \emptyset$
- $X_1 = (v_1, v_2)$
- $X_2 = (v_2, v_1)$
- $X_3 = (v_1, v_2), (v_2, v_1)$

We can see that X_0 becomes X_1 with the arrival of of connection vector (v_1, v_2). Similarly, X_3 becomes X_2 after the disconnection of (v_2, v_1). We can interpret each traffic pattern as a state representing the traffic scenario at an instant. Note that a non-applicable traffic

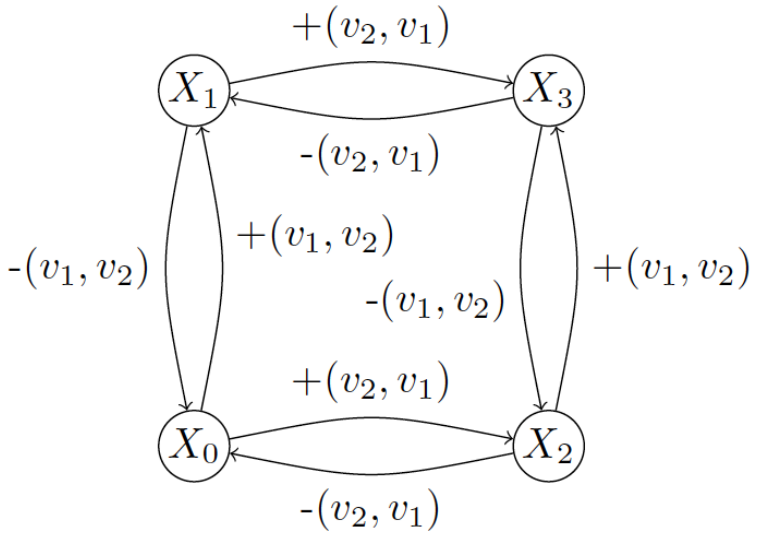

Figure 44: Traffic pattern as network states

pattern to a network is essentially an unreachable state, and thus meaningless.

A traffic pattern depicts a collection of all traffic at an instant in the network. By interpreting them as different states, the term "dynamic traffic" can be depicted as frequent state changes while the network can be modeled as a finite state machine, the states of which are the applicable traffic patterns. Thus we can study the solution at different states (traffic patterns) respectively to evaluate and solve dynamic traffic problems.

However, the complexity to compute either the non-blocking property is unacceptable for a general traffic set. In order to compute the "easiest" property–RNB, we need to compute the solution for each applicable traffic pattern. Let n denote the total number of users and we have $n = \sum_v t(v)$. There are a total of $n!$ traffic patterns in the network, if we consider each user as distinct. If we consider users served by one node are identical, the total number of traffic patterns is $\frac{n!}{\prod_v t(v)!}$, which is less than $n!$ but remains astronomical. Note that we even have not put the complexity of the routing algorithm into account!

We offer an example network as plotted in Figure 45 to illustrate the scale of traffic

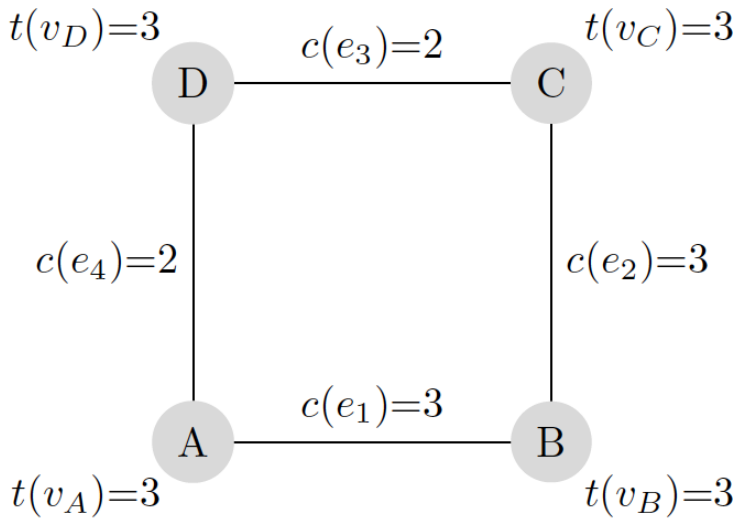

Figure 45: An example network

patterns. The network has four nodes connected with four links. The underlying simple graph of the network is essentially C_4. The nodes weighted by the number of users $t(v)$, and edges are weighted by the number of channels $c(e)$. There are 3 users in each node and each node is adjacent to at least four pairs of channels.

The number of traffic patterns in this "simple" network is $\frac{12!}{4\times 3!}$, the numerical value of which is close to 20 million, which is surprisingly huge compared to the network scale. Alternatively, we can approach non-blocking properties for the underlying simple graph G instead of N. Upon obtaining non-blocking underlying simple graphs, we can build non-blocking networks. In the next section we proceed to "dissemble" the network into slices while keep the graph structure, and study the non-blocking property alternatively for a slice for reduced complexity.

A.3 NETWORK SLICES AND PERMUTATION TRAFFIC SET

A slice, N_i, is a fragment of the original network N, which inherits the underlying simple graph, namely[1] $N_i = \{G, t_i(v), c_i(e)\}$, where $t_i(v) = 1, c_i(e) = 1$. In a slice a node only has one user and a link only has one pair of channels in either direction.

It is noteworthy to mention that this *slice* concept is similar to that in EON studies[59, 60, 61, 62]. Both of them represent a set of independently managed channels in data transmission. In our paper, a slice contains a set of links other than wavebands in a single link as proposed in EON studies. What's more, our paper assumes all channels in the same slice have identical bandwidth, which is contradictory to EON slices in which the channel capacity are subject to change adaptively for traffic.

We offer Figure 46 to demonstrate how to "slice" the network in Figure 45. The term A_i means the i^{th} user at node A. All the links in the slices have one pair of channels in either direction. The first two slices fully inherit the underlying graph. Due to the fact that the links $e_1 = (C, D)$ and $e_2 = (A, D)$ only have two channels, which are already set to the first two slices respectively, there is no link in neither e_1 nor e_2 in slice 3. Note that user D_3 may be blocked, if all the channels in e_1 and e_2 in the first two slices are used.

A slice can be seen as a specific network such that the RNB (Definition 2.4.10) and WSNB (Definition 2.4.9) properties defined by network can be rigorously applied to a slice. For convenience, we can denote a slice with a simple graph G, because the values of function $t(v)$, and $c(e)$ are fixed at one. Then we have:

Proposition A.3.1. *If a slice*[2] *G has RNB/WSNB property, network $N = \{G, t(v), c(e)\}$ with $t(v) = n$ for all $v \in G$ and $c(e) = n$ for all $e \in G$ also has RNB/WSNB property.*

Proposition A.3.1 suggests that we can build up a non-blocking network alternatively by studying non-blocking property for slices (essentially simple graphs). Studying non-blocking slices are more cost efficient because the total number of traffic patterns for a slice is $||V(G)||!$, which is less than the number of traffic patterns of the original network. Thus we can shift

[1] Sometime the underlying simple graph of a slice may be a subgraph of that of N, which depends on the value of t and c. See Figure 46.

[2] We can also say a simple graph G instead as they are virtually equivalent.

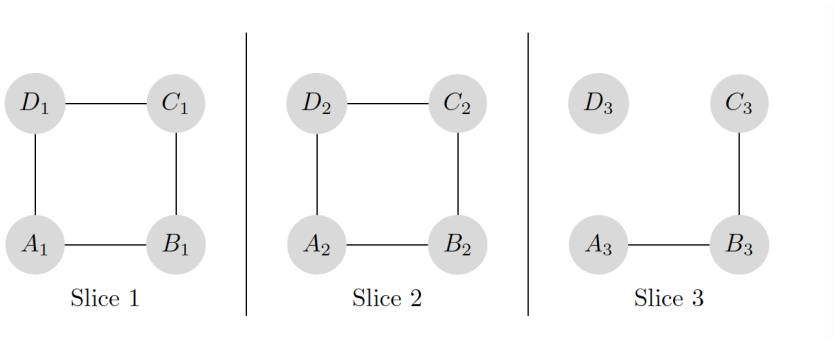

Figure 46: Complicated network into slices

our study to non-blocking slices and investigate its traffic set.

A traffic pattern X applicable in a slice can be represented by a permutation. As $\eta_d(X, v) \leq 1$, $\eta_s(X, v) \leq 1$ for all applicable traffic patterns X and all vertices v, the source and destination vertex pairs can be observed as a reshuffle, which can then be formulated as a permutation. We offer Example A.3.2 to illustrate the formulation process.

Example A.3.2. *Consider $X = \{(A, D), (D, B), (B, A)\}$ in C_4. Its connection vectors are plotted in Figure 47 as the arrow points to the destination vertices.*

If we organize the source/destination vertices according to the order of the connections, particularly in this example, we have the source vertices as an array (A, D, B), and the destination vertices (D, B, A)[3]*.*

Consider the source vertices array as a sequence before reshuffle, and the destination vertices array as the sequence after reshuffle. Then we can summarize a permutation $\pi = (ADB)$ from it. This permutation represents the traffic pattern without ambiguity.

Despite that traffic pattern $X_1 = \{(A, D), (D, B)\}$ can't be precisely represented as a permutation as above as B is not involved as source and A is not involved as destination in X_1, we can find traffic pattern X that has all the connections in X_1 while X can be represented by a permutation. Because the routing paths of X_1 are a subset of that of X.

[3]The connections can be organized in any order. The permutation would remain the same regardless of this order.

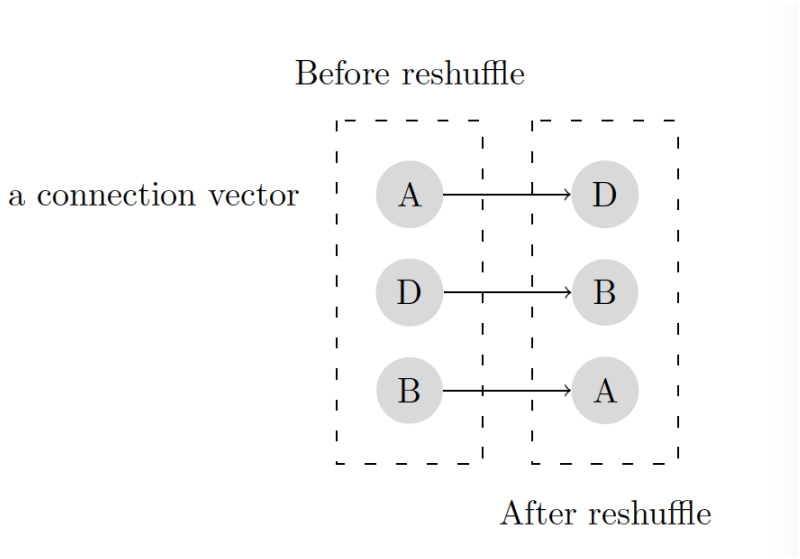

Figure 47: From a traffic pattern to a permutation

Knowing X can be routed means X_1 can be routed. Thus, in the scope of studying the routing paths of the traffic patterns, these two traffic patterns can both be represented by X, which can further be formulated as π.

The traffic pattern for multiples slices can't be represented by a permutation. For example, the traffic pattern for a 3-slice network may be $X_m = (A, B), (A, B), (A, C)$, which can't be represented by a reshuffle as $\eta_s(X, A) > 1$.

This section further introduces the process to model and analyze traffic in a general network and discusses non-blocking property bases on them. The non-blocking property is defined as an enumeration of routing solutions for different traffic patterns in a network. By introducing a slice concept the difficulty of enumerating traffic patterns is reduced. Thus the brute force enumeration of traffic pattern to obtain RNB properties for small scale graphs becomes practical.

BIBLIOGRAPHY

[1] Xiao MA, 2012. *Using Graph Enumeration and Topography Reasoning to Analyze Blocking in WDM Networks Without Wavelength Interchange*, Master Thesis, University of Pittsburgh.

[2] Thompson, Richard A. (2011) *Two Conditions under which WDM Networks are Rearrangeably Nonblocking without Wavelength Interchangers.* In: 2nd Int'l Conference on the Network of the Future, November 28-30, 2011, Paris, France. (Submitted)

[3] Jonathan L. Gross, Jay Yellen, Ping Zhang, 2013. *Handbook of Graph Theory, Second Edition*, CRC Press

[4] Jonathan L. Gross, Jay Yellen, 2006.*Graph Theory and Its Applications, Second Edition*, Chapman & Hall/CRC

[5] P. Raghavan and C. D. Thompson, *Randomized rounding: A technique for provably good algorithms and algorithmic proofs.* Combinatorica, vol. 7, no. 4, pp. 365-734,1987

[6] Biswanath Mukherjee, 2006. *Optical WDM Networks*, Springer

[7] G.A.Dirac, 1960. *In abstrakten Graphen vorhandene vollständige 4-Graphen und ihre Unterteilungen*, Math. Nachr.22

[8] H.A.Jung, 1970. *Eine Verallgemeinerung des n-fachen Zusammenhangs für Graphen* Math.ann.

[9] D.G.Larman, P.Mani, 1970 *On the existence of certain configurations within graphs and the 1-skeletions of polytopes* Proc. London Math

[10] O. Ore, 1960 *A note on hamiltonian circuits*, Amer. Math. Monthly 67

[11] O. Ore, 1963 *Hamiltonian connected graphs*, J. Math. Pures. Appl. 42

[12] B. Jackson, 1980 *Hamiltonian cycles in regular 2-connected graphs*, J. Combin. Theory Ser B 29

[13] J. Moon and L. Moser, 1963 *On hamiltonian cycles in bipartite graphs*, Isr. J. Math. 1

[14] G. H. Fan, 1984 *New sufficient condition for cycles in graphs*, J. Combin. Theory Ser B 37

[15] D. Bauer, H. J. Broersma, H. J. Veldman, R. Li, 1989 *A generalization of a result of Haggkvist and Nicoghossian* J. Combin. Theory Ser. B

[16] V. Chavátal and P. Erdös, 1972 *A note on hamiltonian circuits*, Discrete Math. 2

[17] D. R. Woodall, 1978 *A sufficient condition for hamilton circuits*, J. Combin. Theory Ser. B 25, no.2

[18] P. Fraisse, 1986 *A new sufficient condition for hamiltonian graphs*, J. Graph Theory

[19] G. Chartrand, C. E. Wall, 1973 *On the hamiltonian index of a graph*, Studia Sci. Math Hungar

[20] H. Fleischner, 1974 *The square of every two-connected graph is hamilton*, J. Combin. Theory Ser. B

[21] J.-C. Bermond, 1978 *Hamiltonian graphs* in Selected topics in Graph Theory, L. Beineke and R. Wilson, ed., Academic Press, London

[22] H. A. Jung, 1978 *On maximal circuits in finite graphs*, Annals of Discrete Math.

[23] D. Bauer, A. Morgana, E. Schmeichel, and H. J. Veldman, 1989. *Long cycles in graphs with large degree sums* Discrete Math. 79, no. 1

[24] H. J. Broersma and H. J. Veldman, 1990. *Resrtictions on induced subgraphs ensuring hamiltonicity or pancyclicity of $K_{1,3}$-free graphs*, Contemporary Methods in Graph Theory, R. Bodendiek, ed., BI-Wiss,-Verl., Mannheim-Wien-Zurich

[25] J. A. Bondy, 1977 *Pancyclic graphs*, J. Combin Theory Ser. B

[26] G. Hendry, 1991 *Extending cycles in bipartite graphs* J. Combin. Theory Ser B

[27] J. Komlós, G. N. Sárközy, and E. Szemerédi, 1996 *On the square of a hamiltonian cycle in dense graphs*, Random Structures and Algorithms 9

[28] G. Hendry, 1990 emphExtending cycles in graphs Discrete Math. 85

[29] H. S. Kierstead, G. N. Sárközy, and E. Szemerédi, 1996 *On the square of a hamiltonian cycle in dense graphs*, Random Structures and Algorithms

[30] R. J. Faudree, R.J. Gould, A. Kostochka, L. Lesniak, I. Schiermeyer, and A. Saigo, 2003 *Degree Conditions for k-ordered hamiltonian graphs*

[31] R. Ramaswami and G. H. Sasaki, 1997 *Multiwavelength optical networks with limited wavelength conversion* IEEE INFOCOM

[32] H. Zhang, J. P. Jue, and B. Mukherjee, 2000 *Wavelength Conversions in Quasi-Phase Matched $LiNbO_3$ Waveguide Based on Double-Pass Cascaded $\chi^{(}2)$ SFG + DFG Interactions* OFC '05 Anaheim, CA,pp. OWK6

[33] M. Berkelaar, "lpSolve: Readme file," 1997 *Documentation for lpSolve program*

[34] K. M. Chan and T. S. Yum 1994 *Analysis of least congested path routing in WDM lightwave networks*Proceedings, IEEE INFOCOM, Toranto, Canada.

[35] D. W. Matula, G. Marble, and J. D. Issacson 1972 *Graph coloring algorithms* Graph Theory and Computing

[36] H. Zhang, J. P. Jue, and B. Mukherjee, 2000 *A Review of Routing and Wavelength Assignment Approaches for Wavelength-Routed Optical WDM Networks* Optical Networks Magazine

[37] R.A.Barry and S. Subramaniam, 1997 *The MAX-SUM wavelength assignment algorithm for WDM ring networks* Proceedings OFC, Dallas, TX

[38] X. Zhang and C. Qiao 1998 *Wavelength assignment for dynamic traffic in multi-fiber WDM networks* Proceedings, 7th International Conference on Computer Communications and Networks, Lafayette, LA

[39] Yin, Yawei, et al. *Fragmentation-aware routing, modulation and spectrum assignment algorithms in elastic optical networks.* Optical Fiber Communication Conference and Exposition and the National Fiber Optic Engineers Conference (OFC/NFOEC), 2013. IEEE, 2013.

[40] Zhu, Zuqing, et al. *Dynamic service provisioning in elastic optical networks with hybrid single-/multi-path routing.* Journal of Lightwave Technology 31.1 (2013): 15-22.

[41] Shen, Gangxiang, and Rodney S. Tucker. *Energy-minimized design for IP over WDM networks.* Journal of Optical Communications and Networking 1.1 (2009): 176-186.

[42] Lee, Chankyun, and June-Koo Kevin Rhee. *Traffic grooming for IP-over-WDM networks: Energy and delay perspectives.* Journal of Optical Communications and Networking 6.2 (2014): 96-103.

[43] Shen, Gangxiang, Yunlei Lui, and Sanjay Kumar Bose. *Follow the Sun, Follow the Wind? Lightpath Virtual Topology Reconfiguration in IP Over WDM Network.* Journal of Lightwave Technology 32.11 (2014): 2094-2105.

[44] Zhang, Jiawei, et al. emphEnergy-efficient traffic grooming in sliceable-transponder-equipped IP-over-elastic optical networks [Invited]. Journal of Optical Communications and Networking 7.1 (2015): A142-A152.

[45] Lin, Tachun, et al. *Unified mathematical programming frameworks for survivable logical topology routing in IP-over-WDM optical networks.* Journal of Optical Communications and Networking 6.2 (2014): 190-203.

[46] Jager, Monika, et al. *Topology optimization of IP and WDM transport networks with respect to content delivery functions.* Telecommunication, Media and Internet Techno-Economics (CTTE), 2015 Conference of. IEEE, 2015.

[47] Dey, Suman Kr, and Aneek Adhya. *IP-over-WDM network design methodology to improve efficiency in overall expenditure due to cost and energy consumption.* Journal of Optical Communications and Networking 7.6 (2015): 563-577.

[48] Zhang, Xiaoning, Haoran Wang, and Zian Zhang. "Survivable green IP over WDM networks against double-link failures." Computer Networks 59 (2014): 62-76.

[49] Karasan, Ezhan, and Ender Ayanoglu. *Effects of wavelength routing and selection algorithms on wavelength conversion gain in WDM optical networks.* IEEE/ACM Transactions on networking 6.2 (1998): 186-196.

[50] Ramaswami, Rajiv, and Kumar N. Sivarajan. *Routing and wavelength assignment in all-optical networks.* IEEE/ACM Transactions on Networking (TON) 3.5 (1995): 489-500.

[51] Banerjee, Dhritiman, and Biswanath Mukherjee. *Wavelength-routed optical networks: Linear formulation, resource budgeting tradeoffs, and a reconfiguration study.* IEEE/ACM Transactions on Networking (TON) 8.5 (2000): 598-607.

[52] Mukherjee, Biswanath, et al. *Some principles for designing a wide-area WDM optical network.* IEEE/ACM transactions on networking 4.5 (1996): 684-696.

[53] V.E. Bene. *Mathematical Theory of Connecting Networks and Telephone Traffic. Front Cover.* Academic Press, Jan 1, 1965

[54] Jinno, Masahiko, et al. *Spectrum-efficient and scalable elastic optical path network: architecture, benefits, and enabling technologies.* IEEE Communications Magazine 47.11 (2009).

[55] Mun, Youngsong, and Hee Yong Youn. *Performance analysis of finite buffered multistage interconnection networks.* IEEE Transactions on Computers 43.2 (1994): 153-162.

[56] Maakar, Sunil Kr, and Yudhvir Singh. *Traffic Pattern based Performance Comparison of Two Proactive MANET Routing Protocols using Manhattan Grid Mobility Model.* International Journal of Computer Applications 114.14 (2015).

[57] Marsan, M. Ajmone, et al. *Optimal energy savings in cellular access networks.* 2009 IEEE International Conference on Communications Workshops. IEEE, 2009.

[58] Sen, Priyanka Pradeep, and Vaishail Sahare. *Design of Traffic Pattern Discovery System in Mobile Ad Hoc Network.* Communication Systems and Network Technologies (CSNT), 2015 Fifth International Conference on. IEEE, 2015.

[59] Gerstel, Ori, et al.*Elastic optical networking: A new dawn for the optical layer?.* IEEE Communications Magazine 50.2 (2012): s12-s20. APA

[60] Tomkos, Ioannis, et al.*A tutorial on the flexible optical networking paradigm: State of the art, trends, and research challenges.* Proceedings of the IEEE 102.9 (2014): 1317-1337.

[61] Jinno, Masahiko, et al. *Multiflow optical transponder for efficient multilayer optical networking.* IEEE Communications Magazine 50.5 (2012): 56-65.

[62] Zhang, Guoying, et al.*A survey on OFDM-based elastic core optical networking.* IEEE Communications Surveys & Tutorials 15.1 (2013): 65-87.

[63] Qin, Xiangdong, and Yuanyuan Yang. *Nonblocking WDM switching networks with full and limited wavelength conversion.* IEEE Transactions on Communications 50, no. 12 (2002): 2032-2041.

[64] Rasala, April, and Gordon Wilfong. *Strictly nonblocking WDM cross-connects.* SIAM Journal on Computing 35, no. 2 (2005): 449-485.

[65] Zhou, Chunling, and Yuanyuan Yang. *Wide-sense nonblocking multicast in a class of regular optical WDM networks.* IEEE Transactions on Communications 50, no. 1 (2002): 126-134.

[66] Barry, Richard A., and Pierre A. Humblet. *Models of blocking probability in all-optical networks with and without wavelength changers.* In INFOCOM'95. Fourteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Bringing Information to People. Proceedings. IEEE, vol. 2, pp. 402-412. IEEE, 1995.

[67] Hsu, Ching-Fang, Te-Lung Liu, and Nen-Fu Huang. *Performance analysis of deflection routing in optical burst-switched networks.* In INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, vol. 1, pp. 66-73. IEEE, 2002.

[68] Barry, Richard A., and Pierre A. Humblet. *On the number of wavelengths and switches in all-optical networks.* IEEE Transactions on Communications 42, no. 234 (1994): 583-591.

[69] Costa, Lucas R., and Andr C. Drummond. *New distance-adaptive modulation scheme for elastic optical networks.* IEEE Communications Letters 21, no. 2 (2017): 282-285.

[70] Zhou, Xingyu, Qunbi Zhuge, Meng Qiu, Meng Xiang, Fangyuan Zhang, Baojian Wu, Kun Qiu, and David V. Plant. *On the capacity improvement achieved by bandwidth-variable transceivers in mesh optical networks with cascaded ROADMs.* Optics Express 25, no. 5 (2017): 4773-4782.

[71] Zhao, Li, Weisheng Hu, and Xiaojian Zhang. *Architecture and performance of grouped ROADM rings with shared optical amplifier and grouped add/drop ports for hybrid data center network.* Optical Switching and Networking 23 (2017): 1-4.

[72] Zhao, Yongli, Rui Tian, Xiaosong Yu, Jiawei Zhang, and Jie Zhang. *An auxiliary graph based dynamic traffic grooming algorithm in spatial division multiplexing enabled elastic optical networks with multi-core fibers.* Optical Fiber Technology 34 (2017): 52-58.

[73] Tan, Yanxia, Rentao Gu, and Yuefeng Ji. *Energy-efficient routing, modulation and spectrum allocation in elastic optical networks.* Optical Fiber Technology 36 (2017): 297-305.

[74] Ju, Min, Fen Zhou, Shilin Xiao, and Haitao Wu. *Leveraging Spectrum Sharing and Defragmentation to p-Cycle Design in Elastic Optical Networks.* IEEE Communications Letters 21, no. 3 (2017): 508-511.

[75] Shakeri, Ali, Miquel Garrich, Anderson Bravalheri, Davide Careglio, Josep Sol-Pareta, and Andrea Fumagalli. *Traffic allocation strategies in WSS-based dynamic optical networks.* Journal of Optical Communications and Networking 9, no. 4 (2017): B112-B123.