Power Efficient Survivable Routing With P-Cycles

by

Kristoffer Fedick

A Thesis

Presented to Lakehead University

in Partial Fulfillment of the Requirement for the Degree of

Master of Science

in

Electrical and Computer Engineering

Thunder Bay, Ontario Canada

Abstract

Power awareness in networking has been a vital area of research in wireless networks but, until recently, has been largely ignored in wired networks. In wireless applications, the amount of power utilized by transmission is of vital importance since it will limit factors such as battery life and transmission range. In wired networks, the power issues of wireless networks do not arise since the wired networks receive their power from the power grid. However, the problem of operational costs and the environmental impact of wired networks have become increasingly important issues in recent years.

This thesis proposes a power efficient routing scheme to address the environmental and operational cost issues. The operational costs of a wired network can be reduced by reducing the amount of power the network utilizes. The proposed power efficient routing scheme utilizes a demand prediction algorithm to determine a set of expected future traffic. The set of expected traffic is then assigned paths in the network using an energy efficient routing algorithm. The paths that are assigned to the predicted traffic are used to assign paths to the real traffic as it enters the network. By continuously updating the set of expected traffic, and the paths that are assigned to the expected traffic, the energy efficient routing algorithm can maintain an energy efficient routing solution over time, and thus, power efficiency is achieved.

The work in this thesis focuses on the energy efficient routing algorithms that are used in the power efficient routing scheme. Three energy efficient routing algorithms are proposed. Each of the algorithms uses p-cycles to plan the routes that traffic will take through the network. Traditionally, p-cycles are groups of links that form a circular path that are used to plan backup routes for demands. In this work, p-cycles are used to plan both the working and backup paths for each demand. In this way, traffic will be routed around straddling links and encompassed nodes so that they can be put into offline mode (turned off). Straddling links are not a part of the cycle but connect two nodes that are a part of the cycle and encompassed nodes are nodes that are not on

a cycle but are part of a path between two nodes that are on the cycle.

In house Matlab simulations were used to compare the bandwidth and energy efficiency performance of the three proposed algorithms with two established benchmark algorithms used for survivability. These two algorithms represent both the best and worst cases for bandwidth efficiency. Through a comparison with the benchmark algorithms, it will be shown that the proposed algorithms provide an energy efficient set of paths for a given set of traffic. This proves the feasibility of the power efficient routing scheme since the proposed algorithms can provide a set of energy efficient paths for the traffic for varying network loads.

Acknowledgments

I would like to thank my reviewers for taking time out of their busy schedules to review my thesis. I would also like to thank Justin Enders for his help in running my simulations.

To my friends and family. Without your love and support, this thesis would not have been possible.

Contents

List of Figures

List of Symbols

ABP - Average length, in number of links, of the backup path.

ALL - Average Link Load. The average amount of bandwidth used by all of the demands assigned to the network.

AWP - Average length, in number of links, of the working path.

b_i - The bandwidth required for demand i.

$B_{c,i}$ - The bandwidth required for demand i to be accepted on cycle c.

$BA_{d,i,l}$ - The amount of bandwidth available on link l for sharing with demand d.

BAD_j - The free bandwidth on link j.

$BB_{d,l}$ - The total backup bandwidth available on link l for demand d.

$BC_{i,l}$ - The amount of bandwidth needed for current demand i on link l.

$BD_{i,j}$ - The bandwidth of demands that use link j as a backup path and have working paths that are disjoint from demand d.

BG_l - The backup bandwidth needed on link l.

BN - The the amount of bandwidth needed by the new demand.

$BPC_{d,i}$ - Indicates if link i is in the backup path of demand d.

BR_c - The total bandwidth required by all demands in the demand set (DS_c) of cycle c.

BW_j - The used bandwidth on link j.

$BWD_{d,j}$ - The bandwidth on link j for demand d that is assigned to demands that have disjoint working paths from demand d.

C_j - The total capacity of link j.

$CL_{c,j,k}$ - Indicates if link k is a common link with cycle j in set UC provided cycle j is not cycle c.

CP_1 - A set of links in the first least cost path found during the Dedicated Backup Path Protection algorithm.

CP_2 - A set of links in the second least cost path found during the Dedicated Backup Path Protection algorithm.

CP_3 - A set containing all the links in CP_1 and CP_2 that are not common between sets CP_1 and CP_2.

CR - A set of all cycles that have been assigned demands.

CY - A set of all p-cycles for the network.

$D_{i,c}$ - Indicates if both the source and destination nodes of a demand are on cycle c.

DM - A set of all demands that need to be assigned paths.

DRM - Demand Rejection Metric. A measurement of the amount of the demands that are a routing algorithm.

DS_c - The demand set for cycle c. It is a set of all demands that cycle c can potentially support.

F_d - A set of cycles that have already been considered for demand d that could not support the demand. Also called the Failed set for cycle d.

FB_l - The amount of free bandwidth on link l.

GC - A set of all cycles that are grown from a current cycle.

$L_{c,i}$ - Indicates if link i is a common link with any cycle in set UC but is not a part of cycle c.

M_l - The number of demands that have link l as a part of their backup path.

N - The number of links in the network.

N_c - The number of demands in set DS_c.

ND - Number of demands in set DM.

NC - Number of cycles in set UC.

NL_c - Number of links in cycle c.

NP_k - The number of links in path k.

$PS_{c,d,k}$ - The path score for path k of demand d that was assigned to cycle c.

RD - The number of rejected demands.

$S1_c$ - Also called a Stage 1 Demand set for cycle c. It is a set of all demands assigned to cycle c during Stage 1 of the Hybrid Shared Algorithm.

$S2_c$ - Also called a Stage 2 Demand set for cycle c. It is a set of all demands assigned to cycle c during Stage 2 of the Hybrid Shared Algorithm.

S_c - The Cycle Score for cycle c.

$SS_{c,d}$ - The Cycle Score of cycle c for current demand d.

TCL_c - The total number of common links that cycle c has with the cycles in set UC. A cycle cannot have a common link with itself so the number of common links cycle c has with itself are not included in this total.

TD - The number of demands in the network.

TDS - The total number of demands that needed to be assigned paths and bandwidth in the network.

UC - A set of all cycles that have demands assigned to them.

$WPC_{d,i}$ - Indicates if link i is in the working path of demand d.

Chapter 1

Introduction

1.1 Power Minimization

Traditionally, less attention is given to power awareness in optical networks compared to copper or wireless networks. Fibre networks use less power than copper wire networks and, unlike many wireless devices, they are plugged into the grid. For this reason, power hasn't been a major concern when designing optical networks [2, 5]. Some have even viewed the topic of power in optical networking controversial [11] in the past. The main driving force of the growth of networks such as the internet has been the number and size of demands, and therefore, customers that can be provided service. The growth of the number of users and network speeds has led to an exponential increase in bandwidth demand [12, 13, 14]. This is because the operating costs of the network (OPEX) are shared by the users and the growth of the number of users caused the price per bit of data to decrease. The lower cost of using networks further fueled their growth, and therefore, the cost of transmission and switching equipment was considered one of the major barriers of growth to large networks like the Internet [4].

Advances in technology have allowed networks to support more and larger demands. As a result, networks have had no trouble meeting the growing demands. Network designers have traditionally relied on advances in technology to increase performance and lower the capitol costs (CAPEX) [6] to keep the costs to the users low [12]. Components are clocked faster, have a higher degree of parallelization, are physically smaller, and have a lower supply voltage than their older counterparts. According to Moore's Law, the number of transistors per

square inch on integrated circuits doubles every year and therefore, the performance of those components shows a similar increase [15, 16]. Advances in IC technologies have allowed supply voltages to be lowered as well. The power used by ICs is measured as: $Power = \frac{CapacitiveLoad \times Voltage^2 \times Frequency}{2}$ [15, 16]. The decrease in supply voltage has a greater influence on power than the frequency increase does so, with new advances in technology, networking devices became more power efficient. However, we are reaching the limit of our ability to miniaturize components and the supply voltage can no longer be decreased. Increasing the frequency of operation will cause an increase in power utilization. Increasing power utilization will also increase the heating of the components, and we have reached the limit of our ability to dissipate the heat from high frequency processors [12]. This phenomenon is called The Power Wall [15, 16]. Because of the power wall, network designers can no longer rely on advances in technology alone to keep networks affordable as demand continues to grow in the future. As a result, the future growth of large networks such as the Internet is in jeopardy [13].

The power density and utilization of a network increases with user demand. The wired networks buy their power from electrical companies and, due to rising electrical costs, power utilization is becoming a significant factor in OPEX [1, 4, 6, 9]. Another source of rising OPEX costs is from the component heating caused by power utilization. As stated above, an increase in power utilization also causes a rise in the heat generated by components. This increase in heating could cause large scale networks like the Internet to require expensive cooling equipment [12]. The cost of installing, powering, and operating such cooling equipment will drastically increase OPEX costs. Thus, by lowering the power utilized by the network, the OPEX costs can be lowered. The lowering of OPEX costs causes the network to be more affordable to its users, and the future networks can be kept more affordable.

Aside from lowering OPEX costs, there are four other reasons for power efficient routing in wired networks:

- Lowering current power inefficiencies. [11]
- Enable greater deployment. [11]
- Benefits in the event of a disaster. [11]
- Lesser environmental impact. [1, 2, 4, 6, 8]

The current methodology of maximizing throughput and minimizing latency during network design is power inefficient. Network components are turned on at full power 24/7 regardless of the traffic load or, in the case of backup resources, if they aren't in use. Add to this the fact that network designers over-provision the resources to account for future rapid growth, and it can be seen that there is a large amount of power waste [11, 5]. This waste is further compounded by the fact that the components will generate heat while active. Heating equipment requires power to operate and, the more heat that needs to be dissipated, the higher the power expended in removing it from the system. Cooling systems can possibly double the power consumption of a network [8].

One of the barriers of Internet deployment in some parts of the world is the scarcity and cost of electricity [11]. High energy costs increase the cost of accessing the Internet to users and high costs cause the demand to stay low from lack of new users. The network providers in an area with high energy costs will have little reason to expand the capacity and reach of their network to meet the demands of new users. Areas with frequent power outages or where there simply isn't enough power to spare for a network with high power consumption would also benefit from power efficiency. The backup power supplies can keep the networks running longer or with less equipment during power outages. Power efficiency also allows a larger and higher capacity network to be built that puts a lower strain on the power grid in an area where a lot of power isn't available. In the event of a disaster, low power equipment is extremely useful. Parts of a network, or the entire network, require backup power sources to keep them operational. Emergency services would benefit from having access to their networks for longer and share data to help handle the disaster situation.

Another important issue that has been gaining more attention recently is the environmental impact of large scale networks [1, 2, 6, 8, 10]. The effects of greenhouse gas emissions have been linked to global warming and climate change. The full effect of climate change due to global warming has been an issue of rising concern. The environmental impact of a network is measured in terms of its carbon footprint. The term "carbon footprint" is used to describe the amount of carbon dioxide (CO_2) emitted during a one year period. CO_2 is the primary greenhouse gas that causes global warming and so reducing CO_2 emission is an important step in combating climate change. The carbon footprint of any system can be either direct, such as from car exhaust, or indirect, such as effects on other direct sources like in power. Many power plants use fossil fuels to generate electricity and emit CO_2 as a byproduct. The amount of electricity required, and therefore, the amount of fossil fuels burned is directly related to the users of the electricity. Because of the relation between users and fossil fuels burned by power plants, the CO_2 emissions are divided up among the customers. This "divided up" CO_2 makes up

part of the carbon footprint of any given company or even individual household. Therefore, reducing the carbon footprint of a network can be achieved by reducing its power usage.

The issues discussed above provide a strong case for more power efficient networking. Besides the benefits to the continued growth and deployment of already existing optical networks, the establishment of new optical networks in areas where power is scarce is possible. The power savings to already existing networks helps reduce their carbon footprint and will also help take some of the demand off of the power grid. However, power awareness in network design introduces a new set of challenges for network designers. It requires changes to the way networks are designed and operated at three levels [11, 12, 3]:

- The individual switch level
- The network level
- In the topology level

As discussed above, network resources are over utilized. Devices are turned on at full power regardless of the traffic load. Traffic load is not constant with time and exhibits burstiness. It has idle periods and nightly variations [2]. When the traffic is idle, the switching components are on at full power as if the traffic is still present. This problem is further compounded when considering backup resources. Backup resources are idle until needed. However, the system will act as if the resources assigned for backup bandwidth are active even when those resources are not (i.e. no working path failures have occurred and the backup resources are not in use). To avoid this issue it has been suggested to put components in offline mode or sleep mode [13, 12, 11, 10, 8].

There are two types of power consumption [10]. The static or "idle"power consumption, which is the power used by components when no traffic is being processed and is independent of the traffic load, and the traffic dependent type which is determined by the power consumption of switching equipment, conversion/regeneration, amplifiers, etc. Energy efficient design aims to reduce the idle power of components by turning them off or putting them into sleep mode. Energy efficient design aims to be more efficient at one given time in the network. This is important to power efficiency since, by aiming to keep energy efficient at all times, the system will be more power efficient as well. Turning components off (offline mode) would obviously provide the best power savings but, in the case of resources assigned for providing backup in case of a failure, would take too long to power

up again to provide adequate protection when a failure occurs. Components in a low power state (sleep mode) would be able to switch back on in a timely manner when a failure occurs. This does not mean that offline mode for components is not useful. During low traffic periods, it is possible to route traffic so that some links and nodes are not being utilized. When not being utilized, the resources for amplification, sending traffic down the link in question, etc. can be powered off. Likewise, nodes that are not being utilized can be switched off as well. Currently, components (line cards, switches, etc.) do not support sleep mode operation. One of the goals of the IEEE 802.3az Task Force is to compose a set of standards to support selective sleep mode for networking components [17, 5]. However, in order to get the most benefit from sleep/offline mode resources, the way traffic is routed has to be altered in order to try to maximize the amount of resources that can be placed into sleep/offline mode. Therefore, changes at the network level have to be made.

Aggregating traffic along similar routes and along as few routes as possible will maximize the number of idle resources available that can be switched into a low power or offline state. Backup routes can also be aggregated together in order to maximize the amount of backup resources available to be put into sleep mode. However, care has to be taken when aggregating backup routes since, with too much aggregation, the risk of connections being lost during a failure increases. For example, consider two demands, each sharing the same backup path and bandwidth. If one demand fails, then the second demand will not have a backup path available and will be disconnected if a link on its working path fails. The same issue can occur when aggregating the backup bandwidth of groups of demands. Higher aggregation will lead to a lower resistance to multiple failures.

Network resources are often assigned to handle twice the expected peak demand in order to cope with future growth [2]. However, since network traffic is bursty and has high and low periods, energy efficiency can be achieved by altering the Internet topology to allow for route adaptation under a wide range of network loads. When network traffic is low, the routes can be planned to allow for more components to be switched into sleep mode or even switched off and when traffic is high, the routes can be planned to allow for more traffic to be accommodated and more components to be switched on. In the case of new networks, the designers can specifically plan the network layout so that it is more compatible with the power efficient paradigm. For example, a network could be made with more links so that there are more paths available for demands between any given node or increase the capacity of certain links in the network. Having more available paths provides more opportunities to aggregate traffic. If certain links have more bandwidth available than others, the paths that traffic is aggregated over can be controlled by the network designers. Larger bandwidth links will have more

traffic placed over them than the lower bandwidth links during aggregation since they have more space for more demands. Both cases would be used to make it easier to aggregate backup traffic over similar links since there would be more potential paths available or more bandwidth available on selected links. Having more potential paths available or having links with more bandwidth strategically placed throughout the network will provide more opportunities for aggregation of bandwidth and also for sleep/offline mode.

1.2 Thesis Contributions

The contribution in this thesis is in the introduction of a power efficient routing scheme that operates in three stages:

1. Prediction Stage
2. Path Finding Stage
3. Operation Stage

During the prediction stage, a prediction algorithm is used to generate a set of expected demands for a given period of time, called the prediction period. An energy efficient set of paths for the predicted traffic is then found during the path finding stage. These paths are stored for use with the real traffic as it enters the network during the operation stage. During the operation stage, the real traffic is assigned paths in the network and the demands for the next prediction period are predicted and assigned paths. This is done so that the next set of demands and paths are ready for use when the current operation stage ends. By continuously updating the set of demand predictions, and the paths for the predicted demands, an energy efficient routing solution can be maintained indefinitely as demand conditions in the network change, and thus, power efficiency can be achieved.

This thesis focuses on the energy efficient algorithms used during the path finding stage. The use of p-cycles to find a set of energy efficient paths for network demands is explored and the feasibility of the power efficient routing scheme is proven by illustrating that energy efficiency can be provided at various levels of network load. Network traffic prediction is introduced in this thesis, however, discovery of an appropriate prediction algorithm is left to a future work.

Three energy efficient routing algorithms are proposed in this thesis:

1. Hybrid Shared
2. Modified Hybrid Shared
3. Power Efficient Growing Cycles

The Hybrid Shared algorithm was designed to increase the number of links and nodes that can be put into offline mode and maximize the number of links and nodes that can be put into sleep mode. The algorithm utilizes p-cycles to plan working and backup paths for traffic so that backup paths for traffic will follow the same routes. This causes the number of links and nodes that have only backup paths routed over them to increase. Links and nodes that are only used for backup bandwidth can be put into sleep mode so the number of sleep mode links and nodes increases. The backup paths are planned by routing traffic over the common links of cycles. Links that are shared by two or more cycles are called common links. However, routing backup paths over as many common links as possible can lead to working paths that are longer than necessary. Since demands cannot share working bandwidth, longer working paths leads to more bandwidth usage. This can also lead to more energy usage because links and nodes that make up a demands working path need to be in online mode. The results will show that in networks with a high degree of connectivity the Hybrid Shared algorithm leads to a higher number of links and nodes in sleep mode. However, the larger number of links and nodes in online mode wind up negating any benefits resulting from the increase in sleep mode links and nodes.

The Modified Hybrid Shared algorithm addresses the bandwidth and energy issues of the Hybrid Shared algorithm. This algorithm does not attempt to maximize the number of links and nodes that can be put into sleep mode. The working paths are the shortest possible so the bandwidth and energy issues of the Hybrid Shared algorithm are avoided. As the size of a backup path increases, the chances a link in the backup path being a common link increases. Therefore, there are still opportunities for links that can be put into sleep mode to arise but the algorithm does not attempt to guarantee that they will arise. The results will show that this algorithm will provide a large energy efficiency with a large bandwidth efficiency loss.

The Power Efficient Growing Cycles algorithm provides an algorithm that does not require a set of predicted demands to operate. If a set of predicted demands does not exist, paths are assigned to the demands as they enter the network. If a set of predicted demands exists, paths are assigned to the predicted demands one by one as if they were entering the network one at a time. The results will show that this algorithm will provide a little energy efficiency with a little bandwidth efficiency loss.

Each of the proposed algorithms is compared to two benchmark algorithms. The Benchmark algorithms are the Dijkstra Shared Backup Paths and Dijkstra Dedicated Backup Paths algorithms. The Dijkstra Shared

Backup Paths algorithm is a twin shortest length paths algorithm that allows backup bandwidth sharing between demands and the Dijkstra Dedicated Backup Paths algorithm is a twin shortest length paths algorithm that does not allow backup bandwidth sharing between demands. Together the benchmark algorithms represent both a best and worst case of bandwidth efficiency as well as a baseline for energy efficiency. The tradeoff of energy/power efficiency is bandwidth efficiency. In house Matlab simulations are used to compare the bandwidth and energy efficiency of the proposed algorithms and the benchmark algorithms using the following metrics.

Resource Usage:

- Average Working Path Length
- Average Backup Path Length
- Average Link Load

Network Performance:

- Demand Rejection

Energy Efficiency:

- Links in Sleep Mode
- Links Offline
- Nodes in Sleep Mode
- Nodes Offline

The results will show the proposed algorithms do provide an energy efficient solution for each level of simulated traffic. Since energy efficiency can be achieved for any given network load, it is possible to achieve power efficiency with the proposed power efficient routing scheme provided an accurate demand prediction algorithm is utilized.

1.3 Thesis Outline

This thesis is outlined as follows. Chapter 2 introduces p-cycles, and network traffic prediction. The chapter opens with an introduction to the p-cycle concept. The discussion on p-cycles provides the background knowledge necessary to understand the concept of using p-cycles for power efficiency discussed in Chapter 3. The chapter closes with a discussion on network traffic prediction, why it is important, and how it can be achieved.

Chapter 3 introduces and discusses previous work in the field of power efficiency and the contributions of this work. The chapter opens with a brief discussion of some previous studies of power efficiency. Following this is a discussion on how the p-cycles concept is modified for this thesis to achieve power efficiency. The chapter closes with a discussion of the three proposed energy efficient routing schemes that use p-cycles and how they can be used for power efficiency.

In Chapter 4, the proposed energy efficient algorithms are compared with a set of benchmark algorithms through the use of in house Matlab simulations. The chapter opens with an explanation of the benchmark algorithms and their operation. Following the introduction to the benchmark algorithms is a discussion of the performance metrics used in the comparison of the algorithms and a discussion of the three network topologies used in the simulations. The chapter closes with a presentation of the results and a discussion on the behaviour of each of the algorithms.

Chapter 5 discusses some future avenues of research and a proposal for future research in the area of power efficiency with p-cycles. This thesis concludes in Chapter6 with a discussion of the results of the simulations.

Chapter 2

Theory

2.1 P-Cycles

P-Cycles were discovered by Grover and Stamatelakis [20, 30] and are a way of providing "ring like speed with mesh like efficiency" [19, 22, 25, 30]. P-Cycles are closed loops of pre-configured backup capacity. They are similar to self healing rings but they protect their straddling links as well as their on cycle links. On cycle links are are a part of the cycle itself and straddling links connect two nodes that are on the cycle but not a part of the cycle (Figure 2.1.1).

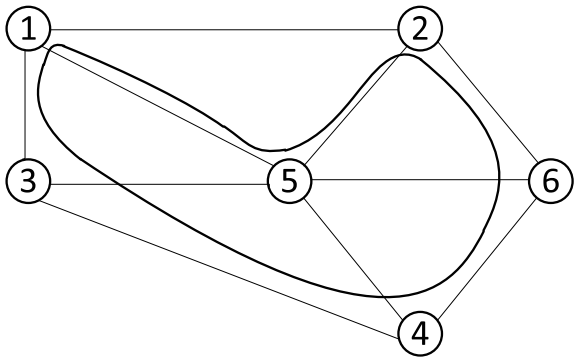

Figure 2.1.1: Example of on cycle and straddling links: Links 1-3, 3-4, 4-6, 2-6, 2-5, and 1-5 are on cycle links and links 3-5, 4-5, 5-6 and 1-2 are straddling links of the cycle shown.

Consider the cycle shown in Figure 2.1.2 A, if an on-cycle link fails (dotted link 2-3 in Figure 2.1.2 B) then one backup path can be provided by the cycle (dashed line; 2-6-5-3). If a straddling link fails (dotted link 6-3

shown in Figure 2.1.2 C) then two backup paths are available (shown in dashed lines). If a link that is not an on-cycle or straddling link fails (dotted link 5-4 shown in Figure 2.1.2 D), then no paths are available for restoration.

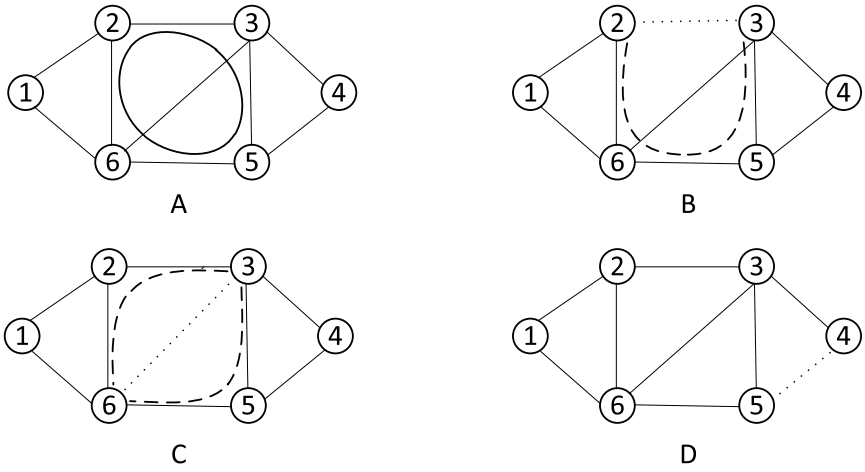

Figure 2.1.2: Example of failures in p-cycle networks: (A) The cycle being considered. (B) On-cycle link failure. (C) Straddling link failure. (D) Failure of link that is not an on-cycle or straddling link.

When using p-cycles for link protection, it is possible to achieve recovery times between 50 and 150 ms [20, 24]. This is because only the two nodes on either side of the failing link need to do any switching since the recovery path for the failed link is pre-configured prior to the failure. Since the two nodes on either side of a failure are the only nodes performing any switching action; the need for complicated signaling schemes to switch from working to backup path is eliminated. Since cycles are able to protect both on-cycle and straddling links, and protection capacity on a cycle is shared by more than one link (either on cycle or straddling), it is possible to achieve 100% restorability from any single failure with less than 100% redundancy [19, 20, 22, 24, 27]. For example, three p-cycles are used to protect the network shown in Figure 2.1.3 (A) for the two demands shown in Figure 2.1.3 (B). If any of the links in the working path for demand 1 fails, the three cycles provide backup paths for each as shown with dashed lines in Figure 2.1.3 (C). These backup paths are pre-configured prior to any failure in the working path. This means that the paths are determined and the resources necessary for the working links to be restored is reserved before transmission begins. If both demands 1 and 2 are being protected, they can share resources over links 3-5 and 5-6 on the backup paths shown in Figure 2.1.3 (D). The sharing of backup resources over links 3-5 and 5-6 make it possible to protect demand 2 and, over link 2-3, demand 1 with less than 100% redundancy but only in the case of a single failure.

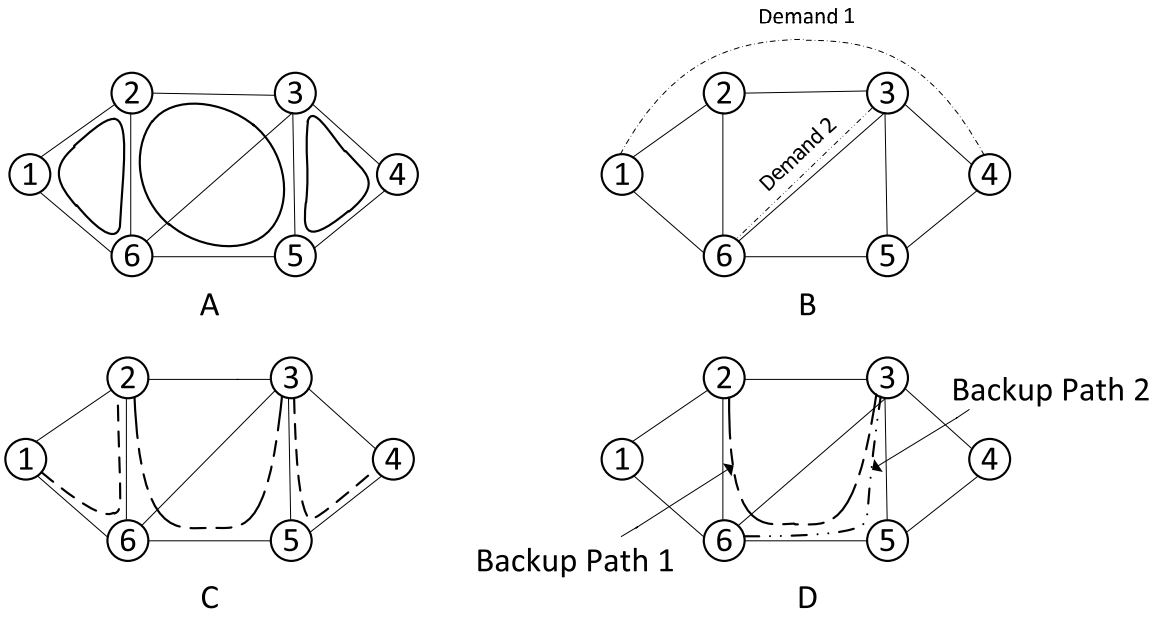

Figure 2.1.3: Example of p-cycle link protection: (A) A test network with 3 p-cycles for link protection. (B) Two example demands. (C) Backup paths in the case of link failure events on single demand. (D) Backup paths in case of link failure on both test demands.

P-Cycles can be used to protect [25, 29]:

- Individual links - An individual link in a path between two nodes. The working path for Demand 1 in Figure 2.1.3 (B) is made up of the three individual links, 1-2, 2-3, and 3-4. The three cycles shown in Figure 2.1.3 (B) protect each of the links in the working path for Demand 1.
- Entire paths/flows - The path is a group of links that make up the route the data will take between a source and destination node. The path for Demand 1 in Figure 2.1.3 (B) will be 1-2, 2-3, and 3-4. A path for a demand is also known as a flow and it is possible for a path/flow to be made up of a single link.
- Flow segments - A path/flow can be broken down into multiple pieces. These pieces can be made up of any number of links. For example: Demand 1 in Figure 2.1.3 (B) can be broken down into 2 or three flow segments. These possible segments are shown in Table below:

Flow 1	1-2	
Flow 2	2-3	3-4

Set 1

Flow 1	1-2	2-3
Flow 2	3-4	

Set 2

Flow 1	1-2
Flow 2	2-3
Flow 3	3-4

Set 3

Table 2.1: Three possible sets of flow segments for Demand 1 in Figure 2.1.3 (B)

As described above; p-cycles protect individual links that are either on-cycle or straddling links. However, if protection of a multilink path is desired, the link protection concept has to be modified to account for paths or path segments. Instead of individual links being protected, the entire path can be protected by a single cycle or parts of the path (also called a flow) can be protected by multiple cycles. Path and flow protecting p-cycles are similar to link protecting p-cycles except that flows and paths are dealt with instead of links. The straddling links become straddling spans and the on-cycle links become on-cycle paths or on-cycle flows. A span is a set of links that make up either a path or a flow. For example, in Figure 2.1.4 two cycles protect the network. The path between nodes 1 and 6 (dashed line) is divided up into two flows. One flow between nodes 1 and 3 is an on cycle flow and protected by Cycle 1 and the other flow between nodes 3 and 6 is protected by Cycle 2. The path between nodes 2 and 4 (dotted line) can be protected by cycle 2, and therefore, is path protected. Node 5 is encompassed by Cycle 2 and entirely protected from failure. Path 3-5-6 is seen as a straddling span.

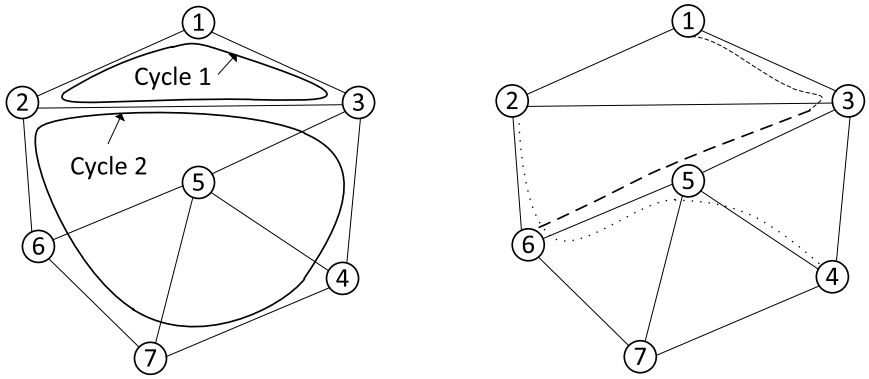

Figure 2.1.4: Example of Flow/Path protecting Cycles: Cycles 1 and two are shown on the left. On the right, two example paths.

Flow/path protection has the advantage of being able to protect nodes or groups of nodes that are encompassed by the cycle from failure. In Figure 2.1.5 (A), if link protection is used, the shaded node (node 7) will not be protected by the cycle since it is not an on-cycle node. The dotted line links (2-7, 3-7, 5-7, and 6-7) will not be protected either because they will pass through node 7 to form straddling spans and not straddling links. If the cycle were to pass through each node in the network, as shown in Figure 2.1.5 (B), then a single cycle can protect all nodes and links in the network from any single failure. For flow and path protection, the cycle shown in Figure 2.1.5 (A) is adequate for protecting all links and nodes because paths between nodes 2, 3, 5, and 6 that flow through node 7 are straddling spans. Node 7 is also protected from a failure since, any path or flow that passes through node 7, will be protected by the cycle. The cycle does not have to pass through every link in the

network to provide protection for every link and node in the network in the case of path and flow protection.

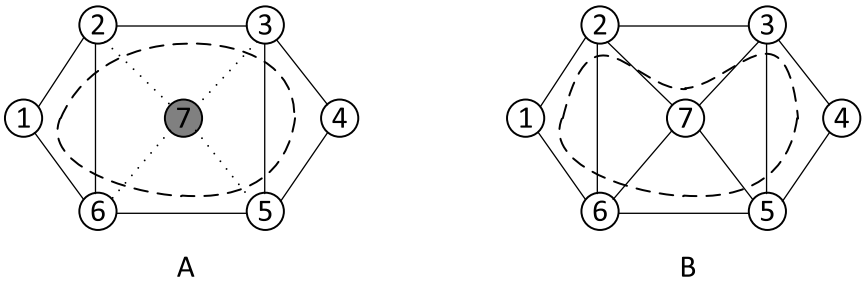

Figure 2.1.5: Protection of Encompassed Nodes: P-cycles are shown in dashed lines. (A) With link protection, the shaded node and dotted links cannot be protected by the cycle. (B) The Hamiltonian cycle is able to protect all links and nodes.

Another advantage of flow/path protecting p-cycles is that they can better coordinate resource sharing between demands. An example is shown in Figure 2.1.6. A network with two cycles to protect it is shown in Figure 2.1.6 (A). Figure 2.1.6 (B) shows two demands. One between nodes 1 and 6 (dashed line), and another between nodes 2 and 4 (dotted line). If link 1-3 were to fail, cycle 1 (1-2-3-1) would restore traffic for link 1-3. If node 5 were to fail, cycle 2 (2-3-4-7-6-2) would restore traffic between nodes 3 and 6 and the traffic between nodes 2 and 4 and the bandwidth for the backup path can be shared between the upper and lower cycles over link 2-3. If links 1-3 and node 5 were to both fail, the traffic would be restored by both cycles. The backup paths for the case of links 1-3 and node 5 in the example network is shown in Figure 2.1.6 (C). The dashed lines show the two backup paths necessary to restore traffic to the demand between nodes 1-6. The dotted line shows the backup path necessary to restore traffic between nodes 2 and 4. The demand between nodes 1 and 6 is broken up into two paths (two dashed paths shown in Figure 2.1.6 (C)). These paths cannot share bandwidth since they are part of the same demand. However, the dotted path on cycle 2 and the dashed path on cycle 1 can share their bandwidth over link 2-3. In this way, cycles are able to share backup bandwidth between each other. Alternatively, the dotted and dashed paths on cycle 2 can share bandwidth over link 2-3.

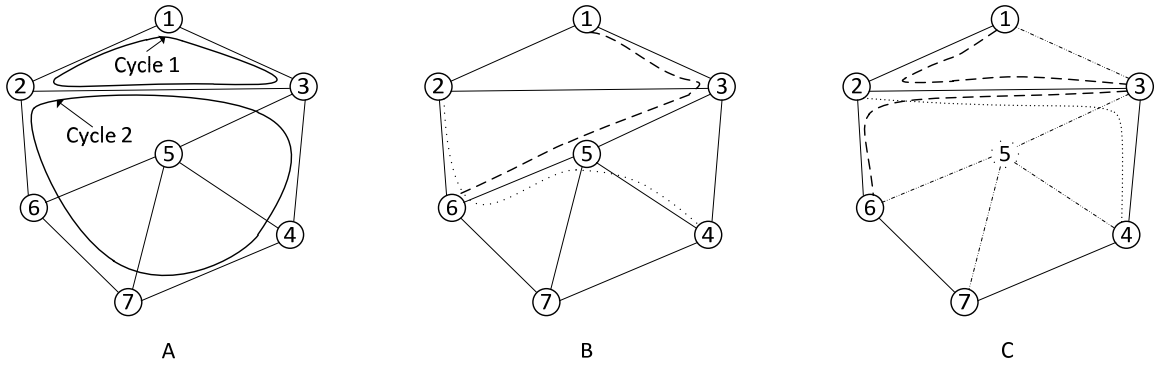

Figure 2.1.6: Resource sharing among demands and a failure event: (A) A network with its two cycles. (B) The working paths for two demands: 1-6, and 2-4. (C) Cycles can protect against single link and node failures.

There are two types of p-cycles: Hamiltonian and non-Hamiltonian [20, 21, 29, 30]. Hamiltonian cycles are cycles that traverse every node in the network once. Non-Hamiltonian p-cycles, which are referred to as just "p-cycles" in the literature, can be any size but do not go through every node in the network (Figure 2.1.7).

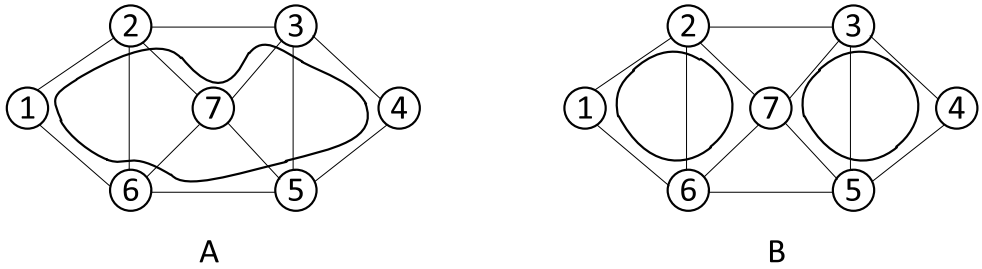

Figure 2.1.7: Types of P-Cycles: (A) A Hamiltonian cycle. (B) A non-Hamiltonian.

Hamiltonian cycles are useful for protecting every link in the network simultaneously. However, their redundancy is not as good as non-Hamiltonian cycles since they are less resistant to link failures. Hamiltonian cycles are good for making backup path computation quicker since there are fewer of them than there are non-Hamiltonian cycles. However, non-Hamiltonian cycles allow for better protection from failures since more than one cycle is being used for planning backup paths. Figure 2.1.8 shows a network with both Hamiltonian cycles and non Hamiltonian cycles. With Hamiltonian p-cycles, the network can be protected by a single cycle (Figure 2.1.8 (A)). When a single failure occurs, the cycle can provide a recovery path for it (Figure 2.1.8 (C)). However, when more than one failure occurs, the cycle can provide a path for only one of the failures (Figure 2.1.8 (E)). For non-Hamiltonian p-cycles (Figure 2.1.8 (B)) the network cannot be protected by only one cycle. However, when a failure occurs, only one of the cycles is necessary to provide a backup path for the failed link (Figure

2.1.8 (D)). When more than one failure occurs, there will be more than one cycle available to provide backup paths and so they can all be restored (Figure 2.1.8 (F)).

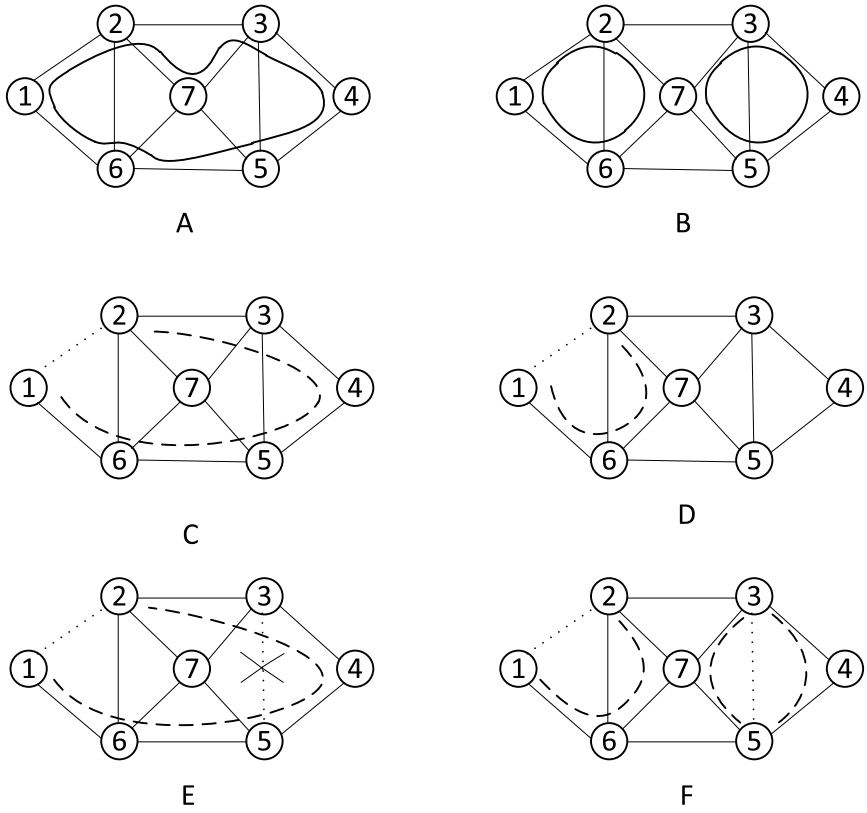

Figure 2.1.8: P-Cycles when a failure occurs for Hamiltonian and regular p-cycles: (A) A Hamiltonian cycle protecting the network. (B) Non-Hamiltonian cycles protecting network. (C) Restoration of a failure with a Hamiltonian cycle. (E) Multiple failures with a Hamiltonian cycle. (D) Restoration of a failure with non-Hamiltonian cycles with link 3-5 not restored. (F) Multiple failures with non-Hamiltonian cycles.

2.1.1 Cycle Discovery

The first step in survivable network design with p-cycles is cycle discovery. Cycle discovery can be performed online or offline. In order to keep path computation fast, cycle discovery is usually performed offline. However, if a purely dynamic solution is desired, cycles can be discovered online as well [19, 23]. Offline calculation will find every possible cycle in the network before any cycles are needed while online calculation will find cycles only when they are needed. Online calculation is used when the network traffic is not known in advance (through historical data and forecasting) and so the p-cycles are assigned to demands dynamically. One way to dynamically assign

p-cycles to demands is described in [19]. The current cycles, which are cycles that are already assigned to a demand, are checked to see if one can provide a backup path for the new demand. If one isn't found, a new cycle is selected. This concept is expanded upon in the Power Efficient Growing Cycles algorithm by allowing the current cycles to be grown to accommodate the new demand. For example, three demands are shown in Figure 2.1.9 (A). When Demand 1 enters the network, the cycle shown in Figure 2.1.9 (B) is used to provide backup paths. One of the nodes for Demand 2 is an on cycle node for the cycle shown in Figure 2.1.9 (B) so it is grown when Demand 2 enters the network. The grown cycle is shown in Figure 2.1.9 (C). The cycle is grown by removing the dotted link and adding the dashed links. When Demand 3 enters the network, it is protected by the new cycle shown in Figure 2.1.9 (D) since neither its source or destination node is on the current cycle shown in Figure 2.1.9 (C).

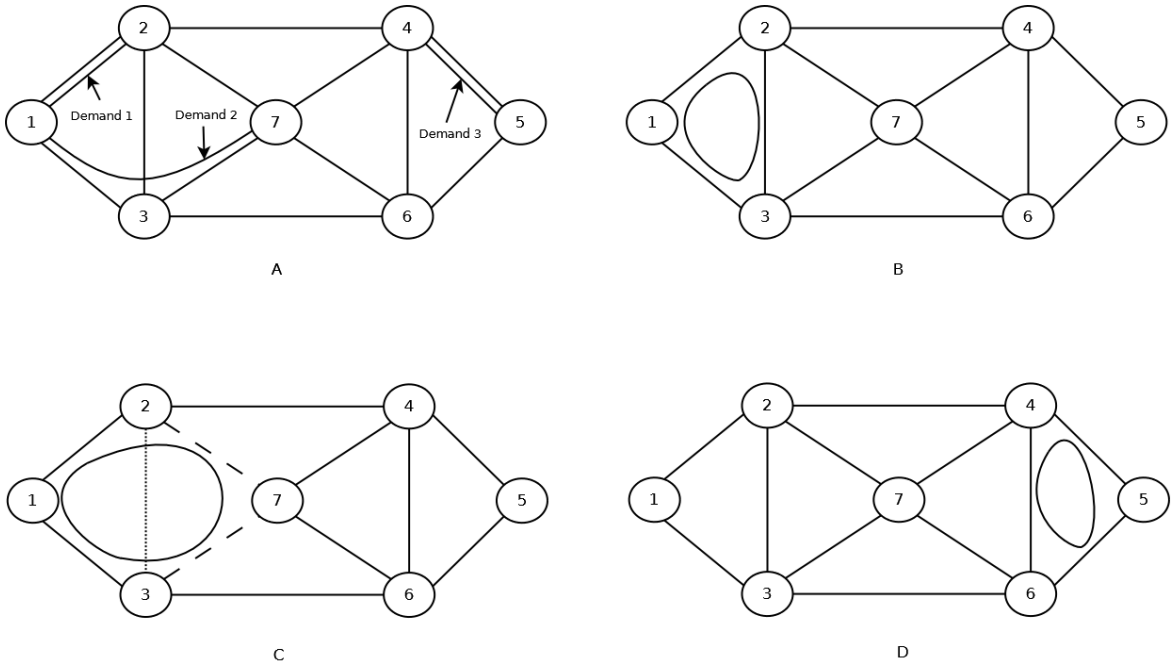

Figure 2.1.9: Growing cycles: (A) Three example demands. (B) Cycle used to protect Demand 1. (C) Result of growing the cycle shown in (B) so that it can protect Demand 2. (D) A new demand is used for Demand 3 rather than growing the cycle in (C).

Three common ways to find p-cycles are [18]:

- Circuit Vector Space Method
- Backtracking Algorithms
- Straddling Link Method

The Circuit Vector Space Method is the simplest of the three methods. It involves finding a simple set of base cycles and then combining them to find the rest of the possible cycles in the network. Two cycles can be combined if they have one or more common links (links that are part of both cycles). The cycles are then stored in a look-up table for later use. The two cycles shown in Figure 2.1.10 (A) have the common links 3-7 and 6-7 so they can be combined to form a new cycle. The new cycle will be made up of all of the links in both cycles minus the common links. The new cycle will be made up of the dashed links shown in Figure 2.1.10 (B) and will not include the dotted links. The resultant new cycle is shown in Figure 2.1.10 (C).

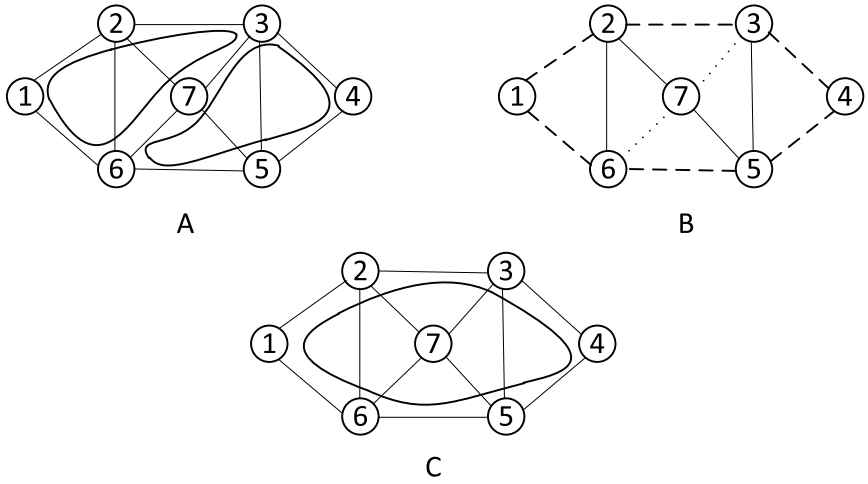

Figure 2.1.10: Combining two cycles: (A) Two cycles to be combined with common links 3-7 and 6-7. (B) Dashed links will make up new cycle while dotted links are excluded. (C) Result of combining the two cycles shown in (A).

This method is the slowest to run since it will find every possible cycle in the network so it is run offline. Figure 2.1.11 (A) shows an example network and all of its base cycles. Combination of the base cycles by growing yields the two cycles shown in Figure 2.1.11 (B). Finally, combining either of the cycles in Figure 2.1.11 (B) with either base cycle (1-2-4-1) or (4-5-6-4) will yield the cycle shown in Figure 2.1.11 (C). Thus, after combining the base cycles with each other to form new cycles, more cycles can be found by combining the newly found cycles with more base cycles.

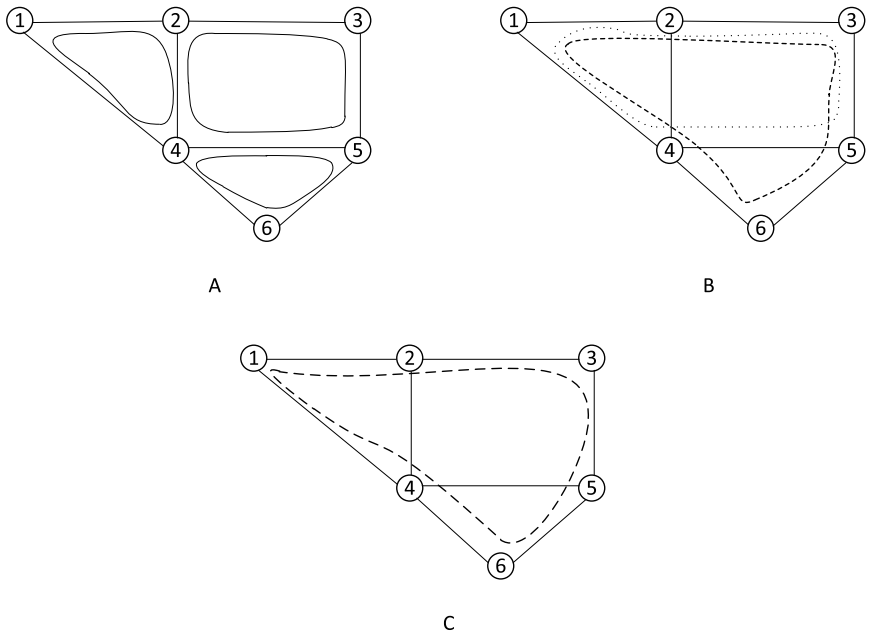

Figure 2.1.11: Circuit Vector Space Method Example: (A) An example network and all of its base cycles. (B) Combination of base cycles yields two new cycles. (C) Cycle that is formed from combinations of discovered cycles in (B) with base cycles in (A).

The Backtracking Algorithms are modified versions of a shortest path algorithm. It starts by finding the shortest path from any given node itself (i.e. source and destination node is the same). Once the destination node is found the algorithm backs up by one link and looks for another way to complete the path. This is done until there are no more unique paths available (i.e. starting from the first link in the searching process and no paths are available to the destination). A new node is selected and the process repeats until all nodes are searched. For example, in Figure 2.1.12 (A), a circular path through the network is found using a path finding algorithm. The algorithm then backs up by one link and looks for another way to complete the path that does not use the link just used (Figure 2.1.12 (B)). One isn't found so it backs up by one more link (Figure 2.1.12 (C)) and then looks for another path that does not use the link just used (dotted link) to complete the cycle (Figure 2.1.12 (D)).

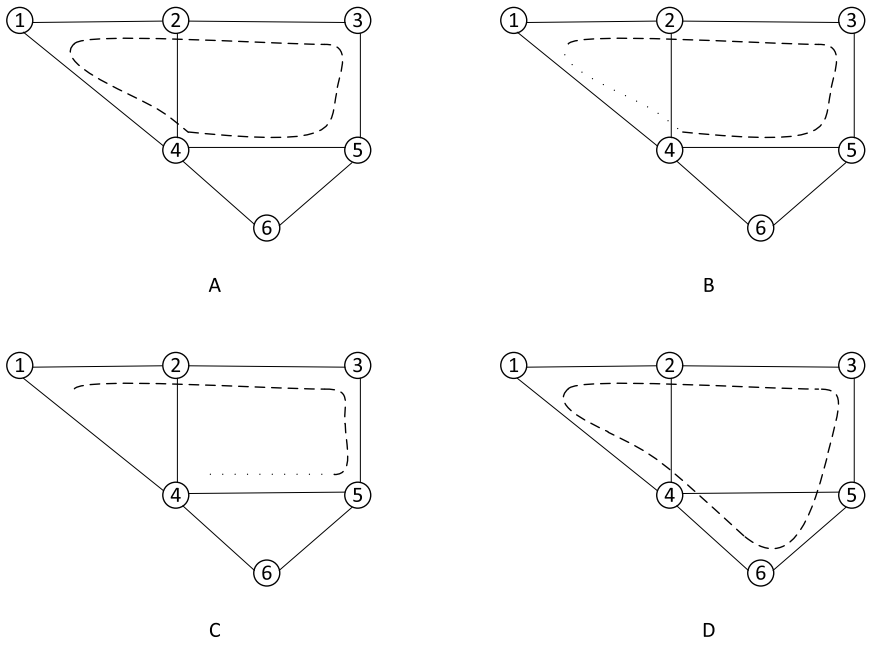

Figure 2.1.12: Example of Backtracking Algorithms: (A) A circular path through the network is found. (B) The algorithm then backs up by one link and tries to find a new path. (C) A new path isn't found so the algorithm backs up by one more link. (D) Algorithm looks for another path.

The Straddling Link method attempts to find straddling links and then form the cycles using a twin disjoint paths algorithm. When considering a source and destination pair, a cycle can be bisected into two halves. These halves are the two possible paths between the source and destination nodes that the cycle can provide. By finding two disjoint paths between a source and destination node, a cycle can be formed. However, in many applications, it is desirable to have a cycle with a straddling span because a cycle can provide two potential backup paths to any working paths that flow over its straddling span. The straddling link method, when possible, tries to find cycles with straddling links. If the source and destination nodes have a degree of three or more than a shortest path algorithm is used to find a direct path between the source and destination nodes. This path becomes the straddling span. Two more disjoint paths are then found to build the cycle. There are some instances where the source node, destination node, or both will have a degree of only two. In that case, there will only be two possible paths between the source and destination nodes and no straddling span will be possible. When only two paths are possible between source and destination nodes; finding the two disjoint paths between the two nodes will yield the two necessary halves of a new cycle. For example, in Figure 2.1.13, node 1 and node 5 want to communicate. In Figure 2.1.13 (A), node 1 has a degree of 2 and node 5 has a degree of 3. The degree of a node is the number of links connected to it. In this case, two disjoint paths are found between nodes 1 and 5. The

cycle is then formed from the two paths. In Figure 2.1.13 (B), node 2 and node 4 want to communicate. Both nodes have a degree of 3 and so three paths should be available between the two. The shortest path between the source and destination nodes (the straddling span of the cycle) is found first. Next, two more disjoint paths are found to make up the two halves of the cycle being constructed. The new cycle is shown in Figure 2.1.13 (C). However, it is possible for two nodes, each with a degree of two or more, to have only two possible paths between them like in the case of nodes 2 and 5 in the network of Figure 2.1.13 (note: a p-cycle cannot pass through the same node twice). In this case, the twin disjoint paths are found and those two paths are then used to create the cycle.

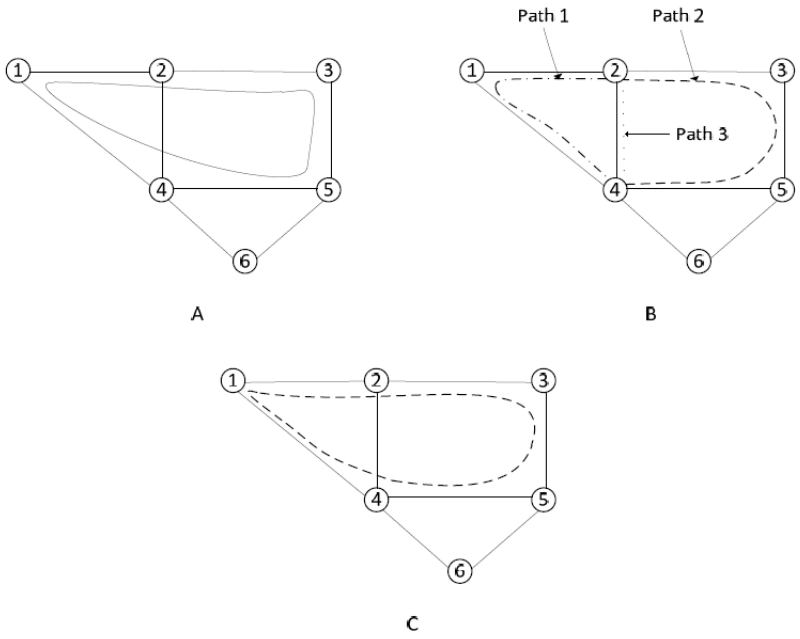

Figure 2.1.13: Straddling Link Method: (A) Cycle formed from two disjoint paths between nodes 1 and 5. (B) Three possible disjoint paths between nodes 2 and 4. (C) Path 1 and Path 2 are combined to form a new cycle.

Unlike the Circuit Vector Space Method and the Backtracking Algorithms, which find all the possible p-cycles in a network at one time, the Straddling link method can be run on demand. This means that a cycle can be found when a new one is needed for a demand rather than having to find the entire list of possible cycles prior to network operation. In the Backtracking Algorithms and the Circuit Vector Space Method, the number of cycles found increases exponentially as the number of links and nodes in a network increases [18, 25]. Because of this increase in the number of cycles found, it is necessary to select a set of best candidate cycles that can be used so the list of cycles used by the network to plan backup routes is smaller, and the routing decision takes less time.

Best candidate cycles are selected using an efficiency metric [19, 25, 26, 30]. Two common efficiency metrics are the *topological score* (TS) and the *apriori efficiency* (AE) [25, 30]. $TS(j) = \sum_{i \epsilon S} x_{i,j}$Where S is a set of links, $x_{i,j} = 1$ if link i is part of cycle j, $x_{i,j} = 2$ if link i straddles cycle j, and $x_{i,j} = 0$ otherwise. The apriori efficiency metric adds a distance cost to the topological score. $AE(j) = \frac{TS(j)}{\sum_{(i \epsilon S | x_{ij}=1)} C_i}$ Where Ci is the cost or distance of span i. However, since the efficiency metric for selecting the best candidate cycles will be dependent on the desired operation of the network, choice of an appropriate efficiency metric is left up to the network designer.

2.1.2 P-Cycle Implementation

Once the cycles are discovered it is time to decide how to utilize them. There are two approaches to p-cycle deployment for protection [20, 21, 25]:

- Non-joint approach
- Joint approach

In the non-joint approach, the working paths for all of the demands are found first. Then, after every demand is assigned a working path, the p-cycles are used to determine the best set of backup paths for the demands. In the joint approach, the working and backup paths for each demand are found at the same time. The backup paths are also determined using the p-cycles in the joint approach. The non-joint approach usually provides a less optimal solution than the joint approach. This is because the joint approach allows the algorithm to ensure that as many working paths as possible are over straddling links. Straddling links provide more backup paths than on cycle links and so more backup paths are available to demands. An example network with three wavelengths available on all links and the working paths for two demands is shown in Figure 2.1.14 (A). In the non-joint approach, the demands are assigned wavelengths independently of each other and their backup paths. Working paths are free to take whichever wavelength(s) that they want without considering the effect they will have on other demands. In Figure 2.1.14 (A), the demands have taken different wavelengths. The possible backup paths that the cycles shown in Figure 2.1.14 (B) can provide for Demand 1 are shown in Figure 2.1.14 (C) and the backup paths for Demand 2 are shown in Figure 2.1.14 (D). Path 2 has wavelengths 1 and 3 available to it while Path 1 has all three available. Path 3 has wavelengths 2 and 3 available to it. If Path 1 is chosen for the backup path to Demand 1 then, in order to share backup resources with Path 3, Wavelength 3 would have to be used. This causes the links in Cycle 2 to utilize three wavelengths. The more efficient way to assign the bandwidth

would be to assign both Demands 1 and 2 to the same wavelength on their working paths as shown in Figure 2.1.14 (E). The joint approach will consider the working paths at the same time. Since they are disjoint, they can be assigned the same wavelength over their working links as shown in Figure 2.1.14 (E) (Wavelength 2 was chosen for this example). Backup paths 2 and 3 are free to take either wavelength 1 or 3 as shown in Figure 2.1.14 (F) (for this example, Wavelength 3 was chosen), causing Cycle 2 to be utilizing only two wavelengths. One wavelength is entirely unused over all the links on Cycle 2.

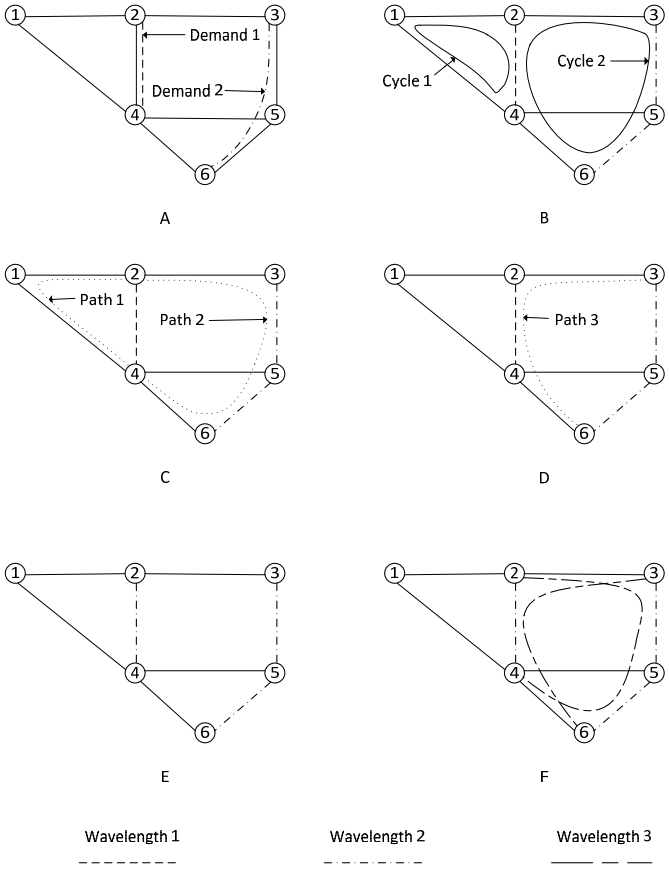

Figure 2.1.14: Example of joint and non-joint cycle assignment: (A) An example network with three wavelengths available on all links, the working paths for two example demands, and the wavelengths chosen for the working paths with the non-joint approach. (B) Two cycles used to provide backup paths for the network. (C) Two backup paths available for Demand 1 using cycles 1 and 2. (D) Backup path available to Demand 2 using Cycle 2. Cycle 1 cannot provide a backup path for Demand 2. (E) Working paths for the two example demands with their assigned wavelengths with the joint approach. (F) Backup paths for Demands 1 and 2 with the joint approach.

P-cycles are usually used for minimizing link cost [25, 30] or for efficient utilization of network resources [20, 30, 31]. However, they can be used for other optimization problems. One such problem is in optimal placement of wavelength converters and also planning where to do signal regeneration [20, 24, 28]. In a network

without wavelength conversion/regeneration, it is necessary to have clear paths (paths that utilize the same wavelength through every link in the path) between source and destination in order to transmit data. This leads to inefficiencies since paths, that could be available if conversion was allowed, would be considered unusable. The path size is also limited since attenuation/degradation of the signal becomes a problem and regeneration is necessary. In Figure 2.1.15 (A), four demands, their working paths, and their required wavelengths are shown. Demand 1 is from node 1 to node 7, Demand 2 is from node 6 to node 2, Demand 3 is from node 6 to node 2, and Demand 4 is from node 6 to node 4. Demands 1, 3, and 4 have clear paths between their source and destination nodes. However, since Demand 1 is already using wavelength 1 over link 2-7, Demand 2 does not have a clear path between the source and destination nodes and cannot transmit. If the signal is converted at node 7 to another wavelength as shown in Figure 2.1.15 (B), the path for Demand 2 can be completed. An example of regeneration is shown in Figure 2.1.15 (C). A demand between nodes 3 and 7 enters the network. However, node 7 is not in range of node 3 and the signal degrades after node 6 (shown as a dotted line). Therefore, it is necessary for node 6 to decode the signal in order to send it to node 7 (regenerate the signal) to complete the path (Figure 2.1.15 (D)).

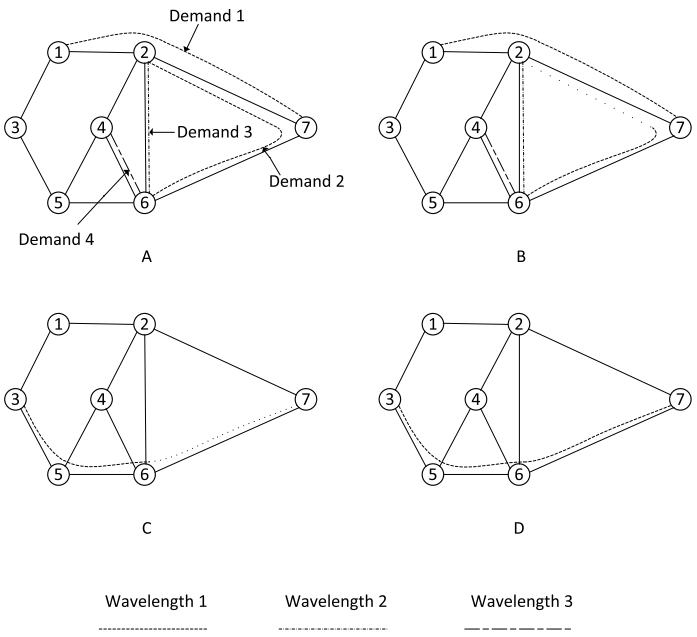

Figure 2.1.15: Conversion and Regeneration: (A) An example network and four demands: 6-4, 6-2, 6-2, and 1-7. (B) Conversion is necessary in order to transmit the data over link 2-7. (C) node 7 out of range of node 3. (D) Conversion of signal at node 6 so nodes 3 and 7 can communicate.

However, wavelength conversion and regeneration requires conversion of the signal from optical to electrical

and back to optical again (OEO conversion). OEO conversion is expensive both equipment and power wise so most networks have partial wavelength conversion (wavelength converters at selected nodes rather than every node). In networks with partial wavelength conversion, the problem is where to place the converters and how many to use. The placement and number of converters are dependent on how the network plans its connections. P-Cycles can solve this problem in two ways [20, 24, 28]. The working and backup paths can be assigned the same wavelength(s) and conversion is unnecessary in the case of a failure. Converters are then only needed when the working path and p-cycle use different wavelengths. Furthermore, because the working and backup paths are the same wavelength(s), the assignment of resources is more uniform, less converters are needed, and the chances of wavelength blocking from broken up paths lowers. The other way to solve the converter problem is to have the working path use one wavelength and then convert only at the access points of the cycle. A node that is attached to a link that is a common link between two or more cycles is called an access point. For example, node 2 in Figure 2.1.16 (A) is an access point between Cycles 1 and 2. Having the working path use one wavelength and converting only at access points of a cycle allows the cycle to support multiple demands at the cost of having a little more converters. In the case of both schemes, paths would be planned so that regenerators are placed where converters are needed. Since conversion and regeneration are performed by the same device, power and component costs are minimized. The example network shown in Figure 2.1.16 (A) has three cycles, each with a different wavelength. Three demands are shown in Figure 2.1.16 (B). Demand 1 is protected by Cycle 1, Demand 2 is protected by Cycle 2, and Demand 3 is protected by Cycle 3. The demands can take any wavelength they want but, by taking the same wavelength as the cycle that will protect them, it is not necessary to convert to the protecting cycles wavelength in the event of a failure. No wavelength conversions are necessary in this case. However, there are cases where a demand will cross from one cycle to another as shown in Figure 2.1.16 (C), in these cases, the working path is broken into sections. Each section is protected by a different cycle and is converted to that cycles wavelength at the appropriate access point. Demand 3 in Figure 2.1.16 (C) is broken into two sections. Section 3-1 is protected by Cycle 1 and converted to Wavelength 1 at the access point between cycles 1 and 2 (Node 2), Section 3-2 is protected by Cycle 2 and is transmitted on Wavelength 2. Demand 4 in Figure 2.1.16 (C) is broken into two sections. Section 4-1 is protected by Cycle 2 and is transmitted on Wavelength 2, Section 4-2 is protected by Cycle 3 and converted to Wavelength 3 at the access point between cycles 2 and 3 (Node 6). To cover the entire network in Figure 2.1.16, it is necessary to have converters at nodes 2, 4, 6, and 8. This is more desirable than having them at every node.

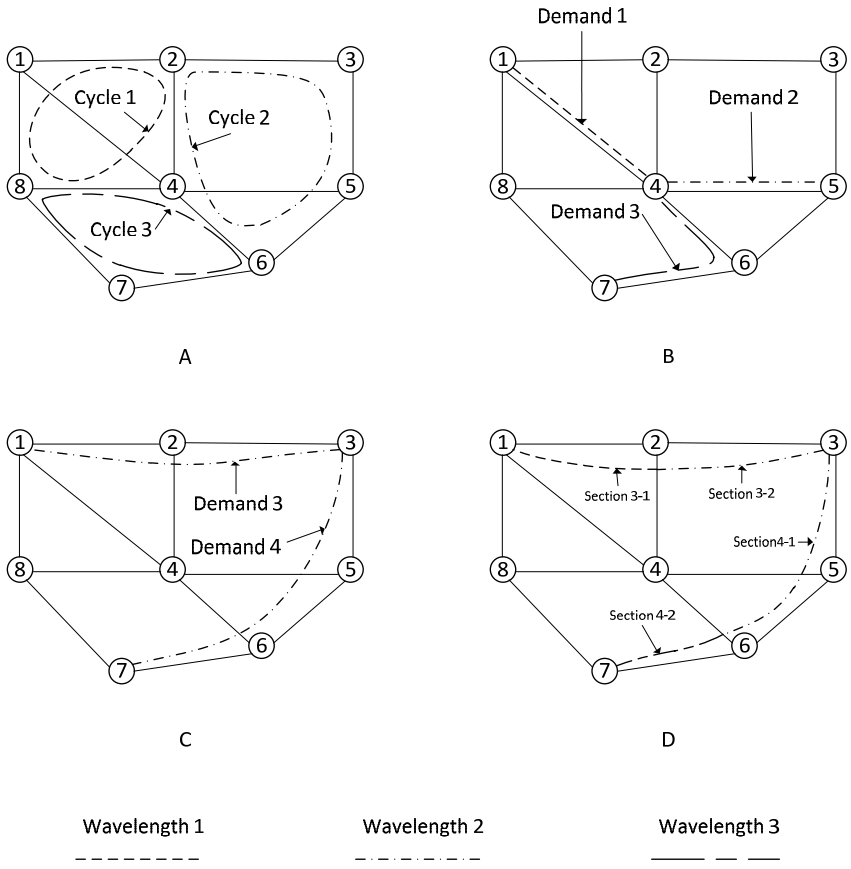

Figure 2.1.16: Example of the two schemes for minimizing converters needed in a network: (A) An example network with three cycles, each of a different wavelength. (B) The working paths for three demands and the wavelengths they will be transmitted on. (C) Working paths for two demands. (D) Conversion of the working paths for Demands 3 and 4 at nodes 2 and 6.

2.2 Network Traffic Prediction

Network traffic prediction is a subject that has been gaining increasing attention. Prior knowledge of the behaviour of network traffic can be very useful in network planning. If the traffic is known before the resources are needed, the best configuration of the network for the traffic can be planned before the traffic arrives. This is very useful since the discovery of an optimal solution is difficult in a purely dynamic case where only the demands that are in the network are known.

Traditionally, network traffic prediction used a Poisson process. However, it was discovered that this process

did not provide a good estimate of real traffic [34]. As a result, traffic measurement studies were performed to determine the nature of network traffic. The most prominent of the studies was performed by Leland and Wilson [36]. In this study, measurements of packet traffic on the Bellcore network were taken over a period of three years. The packet measurements were then studied to determine the behaviour of network traffic. The results of this study showed that network traffic is bursty and that it is self similar in nature. The Poisson process model, which assumes that the aggregate traffic becomes less bursty as the number of traffic sources increases, is inaccurate [36]. Further study has shown that this self similar, "bursty", nature is also present in traffic at the call level [2, 34]. This behaviour is due to the actions of the transmitter (i.e. the users) and not the nature of the network itself [35].

A process is self similar if its future values or outcomes are dependent on past measured values or outcomes. In other words, outcomes measured over a certain period of time behaves the same as the outcomes measured over a different time scale (appropriately scaled) [35]. A self similar process can be either short term dependent or long term dependent. Short term dependent processes are dependent on recent behaviour (recently occurred or short term outcomes), and long term dependent processes are dependent on both recent and long term behaviour (long term outcomes). Network traffic is long term dependent [33, 34, 35, 36, 37]. It has high and low periods and night time variations [2, 5]. These variations repeat themselves over a larger time scale so that daily, monthly, or even yearly patterns can be predicted. The only limit of prediction algorithms is the size of the samples needed in order to project the future behaviour of the network traffic and the choice of prediction method. The choice of prediction method will affect the prediction interval, the error, and the computational cost [32]. The interval is important since, if it is too small, the network will not predict the traffic properly and, if it is too large, it will require more complex computation than necessary. The error is affected by the prediction interval the algorithm uses (some algorithms will have error in them since there is no such thing as a perfect prediction algorithm), and the amount of change that occurs in the system. Networks are constantly growing and the number of customers is constantly increasing. This results in a constant change in the behaviour of network traffic. This change can also be predicted by modeling the new customer demands with time, but that methodology will be very complex since more than one model has to be developed to predict the network traffic over the long term. Therefore, prediction algorithms usually include an over-estimate or margin of error to allow for future changes in the traffic behaviour.

There are many different prediction algorithms for self similar networks and many possible combinations of

those algorithms [32, 38, 39, 40]. For simplicity, only the main techniques will be discussed here. There are four main algorithms for predicting self similar processes:

- FARIMA/ARIMA [32, 38, 40]
- Artificial Neural Networks (ANN) [32, 40]
- Wavelet based predictors [32]
- Genetic Algorithms [39]

The Fractional AutoRegressive Integrated Moving Average (FARIMA) model creates a time series out of past values that is used to project future values. The larger the time series is, the more accurate the model. The FARIMA model is capable of capturing long term dependence, and is more accurate than the ARIMA model, but is much more complex than ARIMA. The ARIMA model can be seen as a special case of the FARIMA model that can only capture short term dependence. It is useful in cases where predictions in the short term are necessary and the long term dependent model isn't necessary. The ARIMA and FARIMA models have the advantage of being easily adjusted to match changes in the behaviour of the process by adjusting the time series used to form the predictions. These adjustments are made by measuring the difference in the predicted value and the actual value as it occurs and then adjusting the formula accordingly. Obviously, this cannot be done with every prediction but the error can be monitored until it reaches a certain threshold value and then the polynomial can be adjusted to account for it.

Artificial Neural Networks are modeled with an input layer, an output layer, and some intermediate layers. The intermediate layers are connected together by "neurons". Each layer can be connected to any other layer using these neurons and, as a result, the neural network can capture the behaviour of complex phenomenon [40]. In order to predict any process, the neural network needs to be trained using historical data. This is done by feeding historical data into the input of the system and using the output to adjust each layer of the neural network to obtain the known output values. This process takes a number of steps to perform and can be time consuming. Another disadvantage is that, if the behaviour of the process changes, the network has to be re-trained.

Wavelet based predictors take the original time series and decompose it into varying frequency components via the Wavelet transform. These components are then used to predict the future behaviour of the time series. The high frequency components are used to predict short term behaviour and the low frequency components predict the long term behaviour.

The Genetic algorithms follow a similar route as evolution to predict future behaviour. The algorithm produces a set of possible solutions called a population. Each solution is called an individual and individuals are grouped together into chromosomes. The solution space is searched to find the best solution by assigning each chromosome a fitness value. The criterion for setting the fitness value is set by the designer. The current population is evolved to create a new population with a higher fitness through two operations: Crossover, where two individuals are mated together in order to exchange genetic information, and Mutation, which is a random change to the chromosome which enables new avenues of the solution space to be included in a given set of solutions. Each generation is created from the last using the evolution operators and the process terminates when a new generation is the optimal solution.

A study of these prediction algorithms and how they apply to network traffic prediction is left to a future work. However, where needed, it is assumed that a suitable network prediction scheme is in place to provide a list of expected demands.

Chapter 3

Power Efficient Routing With P-Cycles

3.1 Background and Previous Work

As discussed in Section 1.1, power efficiency has become an important issue in modern networks. Lowering power usage can reduce operational costs, lower the strain on the power grid, and can lower the environmental impact of the network. Most work has focused on finding the minimum power solution [1, 2, 4, 7, 9, 10, 13, 14]. However, this comes at a cost of increased bandwidth usage and loss of the shortest length path. Therefore, a middle ground should be found that will provide both power and bandwidth efficiency rather than one or the other.

Power efficiency can be achieved by aggregating traffic onto similar paths. Aggregation of traffic makes it possible to minimize power consumption by allowing resources to be switched off or into a low power state. Links and nodes have three modes:

- Online mode - When a node or a link is in use for transmitting data it is said to be in "online mode".
- Offline mode - When a node has no traffic passing through it, and is not a source or destination node for traffic, it can be powered down. Similarly, if a link has no traffic flowing through it, the resources necessary for operating that link (lasers, switching architecture, etc.) can be powered down. When a node is powered down or the resources necessary for operating a link are powered down, the node or link is said to be in "offline mode".

- Sleep mode - When only backup routes pass through a node, that node can be put into a low power state. Similarly, when a link only has backup routes over it, the resources necessary for operating that link (lasers, switching architecture, etc.) can be put into a low power state. When a node is in a low power state or the resources necessary for operating a link are in a low power state, the node or link is said to be in "sleep mode".

Obviously, the offline mode is more power efficient. However, survivability is also important so backup resources also have to be considered. Unless 1+1 protection is used, nodes and links that have only backup paths routed over them can be put into a low power or sleep state until needed. Standardization efforts by the IEEE, discussed in Section 1.1, aim to introduce a selective sleep mode for internal components of routers. So components such as line cards and cross connects can be switched off or into a sleep (low power) state to conserve power. If backup paths are routed over the similar paths like the working resources; nodes and links that only have backup paths routed through them begin to emerge. This is called path aggregation. The example network in Figure 3.1.1 (A) shows a network without aggregation. In this network, there are four links that are always online (1-2, 2-4, 1-3, and 3-7), one link in sleep mode (4-7), and six offline links (2-3, 3-6, 4-6, 4-5, 5-6, and 6-7). There are five online nodes (1, 2, 3, 4, and 7), no sleep mode nodes, and two offline nodes (5 and 6). Figure 3.1.1 (B) shows an example network with aggregation of backup resources. In this case, there are three links that are always online (1-2, 2-4, and 2-7), three links in sleep mode (1-3, 3-7, and 4-7), and six offline links (2-3, 3-6, 4-6, 4-5, 5-6, and 6-7). There are four online nodes (1, 2, 4, and 7), one sleep mode node (3), and two offline nodes (5 and 6). With aggregation, more links and nodes that only have backup bandwidth routed over them are present in the network. More links and nodes in sleep mode lead to power savings.

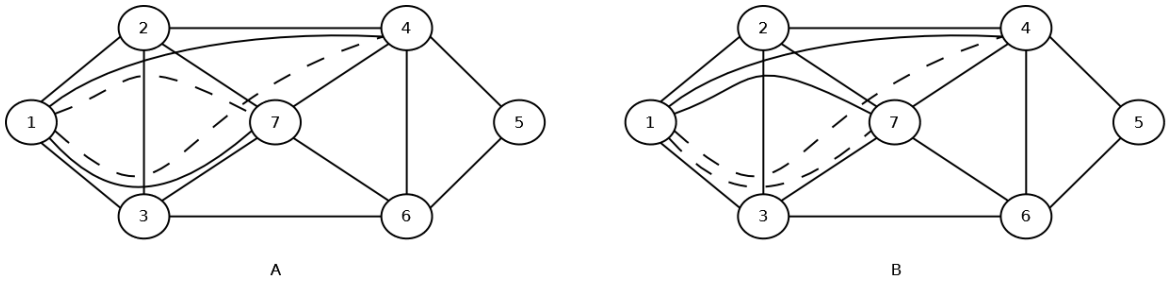

Figure 3.1.1: Example of sleep/offline mode: (A) An example of a network without aggregation. (B) An example network with aggregation of backup resources.

Previous works focused on using wavelength grooming or employed a shortest path discovery algorithm

to achieve aggregation of working/backup paths. In [1], ILP formulations were developed to minimize power consumption in networks with dedicated path protection and have sleep mode support, and for networks that do not support sleep mode. In the case of networks with sleep mode support, the algorithm would find the solution that provided paths for all demands that utilized the least number of links for working paths. Links that are already in use for working paths for other demands are preferred when setting up a working path for a new demand. In this way, the working paths for demands will follow similar routes through a network. This will keep the number of links that have working traffic over them to a minimum and maximize the number of potential links that can be put into sleep mode or offline mode. Similarly, backup paths from each demand are routed over the same links wherever possible. This will keep the number of potential links that can be put into offline mode and at a maximum. Maximizing the number of links that can be put into offline mode is desirable over maximizing the number of links that can be put into sleep mode because offline mode is more power efficient than sleep mode. The number of links in sleep mode and offline mode is maximized but at a cost of paths with a greater number of links (i.e. longer paths). In the case of networks without sleep mode support, the algorithm would find the solution that provides paths to all demands and also utilizes the least number of links. This work aimed to provide optimum power efficiency but did not consider capacity usage.

In [10], ILP formulations were developed to minimize the total energy consumption as well as the total capacity usage. The algorithm would find paths for all demands that will minimize the number of working and backup wavelengths. The working paths can be routed over the same or different routes and the backup routes are shared wherever possible. This work considers both energy and capacity usage but it does not factor in power consumption. It finds the optimal energy use by considering the optimal configuration for a set of demands at one given point in time but does not consider the effect of changing demands. For example, if the network is configured for optimal energy efficiency, as long as the demands stay exactly the same, the network will also be configured for optimal power efficiency. However, power is the average energy expended over time. If some of the demands enter or leave the network or their bandwidth requirement changes, the configuration necessary for minimum energy efficiency will change. If changes in network demands are not taken into account, the network will not be operating in the most energy efficient way at all times, and therefore, it will not be operating in the most power efficient way.

In [2], [9], and [14], the traffic grooming concept was studied to show its effects on power efficiency. In [2] and [14], studies were performed to find the effect on selectively switching sub components (such as line

cards, electrical cross connects, etc) into sleep/offline mode. Grooming is the grouping of traffic bound for the same or a similar destination (i.e. the paths share a number of links in common). Grooming can occur at the waveband level, the wavelength level, and the sub-wavelength level. Sub-wavelengths are grouped under the same wavelength, wavelengths are grouped under the same wavebands, and wavebands are grouped under the same fibre. When traffic is groomed, the network requires fewer resources to transmit data than in networks that do not use grooming. For example, consider the multi-layer optical cross connect architecture shown in Figure 3.1.2. This architecture is similar to that found in [42] and [43] but with an added electrical cross connect for conversion and regeneration like the architectures found in [44]. A network without grooming is shown in Figure 3.1.2 (A). The two signals shown use separate wavelengths during transmission and it is necessary to switch the signals at the wavelength cross connect (WXC). A network with grooming, as shown in Figure 3.1.2 (B) and (C), will switch the two wavelengths onto the same waveband at an intermediary node (shown in Figure 3.1.2 (B)) and each node the data travels to afterward (shown in Figure 3.1.2 (C)) will only have to switch the signals at the waveband cross connect (BXC), using fewer resources.

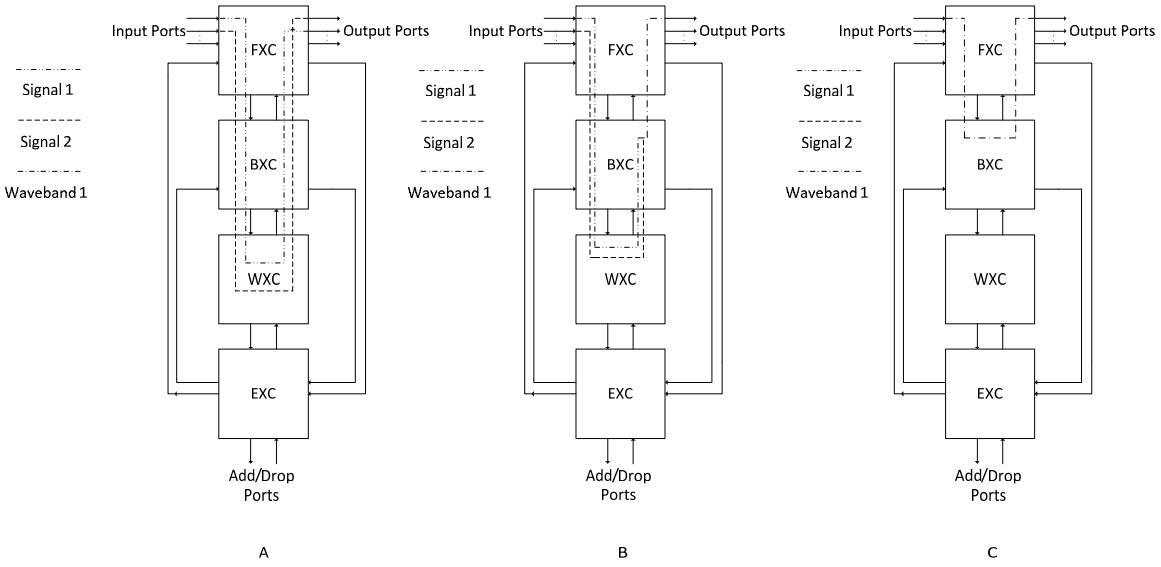

Figure 3.1.2: Example of Grooming: (A) A network that does not utilize grooming. (B) A node in network with grooming where the grooming occurs. (C) A node in a network with grooming where grooming is unnecessary due to grooming occurring at an earlier node.

By grooming, it is possible to lower the number of sub-components necessary to perform switching at each node and, therefore, unneeded components can be switched off. By ensuring that working traffic and backup traffic are kept separate then it can be possible to extend the power savings to include offline mode for sub components that are assigned to handle backup bandwidth only. In [9], the time aware traffic grooming concept

was introduced. Adding time awareness to traffic grooming means that the time a demand is in a network as well as its source and destination are taken into account when making a routing decision. New demands are groomed onto an existing lightpath with a longer remaining holding time whenever possible while also keeping the total length of the path (number of hops) as low as possible. This method does not consider the capacity usage while making routing decisions.

ILP formulations can be complex, require lots of time to solve, and require lots of computing power. Depending on the formulation, the speed problem can be solved by simplifying the ILP problem but at a cost of optimality [10]. Each of the previous works assumed a static routing condition such as the Scheduled Traffic model. The problem with this assumption is, as stated before, it will only give an example of minimizing energy at one given moment and not power which is an average over time.

3.2 Power Efficient P-Cycles

It is possible to extend the static routing model by using network traffic prediction. In this way, the changes to traffic with time can be accounted for by predicting traffic over a large time period. The energy efficient solution for the predicted traffic can then be used to achieve energy efficiency for the real traffic. The predictions can be continuously updated and, therefore, the energy efficient set of paths for the demands can also be updated. In this way, energy efficiency is maintained regardless of changes in the network traffic and power efficiency is achieved. However, this will lead to a large demand set for the static routing problem, and therefore, a very complex ILP. More complex ILP problems require more computer power and time to solve, leading to a higher power usage. Furthermore, since network traffic depends on the number of customers and their bandwidth demand, the traffic on the network will, on average, increase with time as the number of customers and their bandwidth demand increases. The result of the average increase is an introduced error to the prediction that increases with time. This error can be accounted for by monitoring the real traffic data and comparing it to the predicted traffic. The differences can then be used to correct the prediction algorithm (see Section 2.2).

Assuming a static routing algorithm, every time the prediction algorithm is corrected or a new set of predictions for the traffic is generated, the routing algorithm has to be run to plan the energy efficient routes for the new set of traffic. This is a problem if the routing algorithm is very complex and takes a lot of time to complete. Heuristic algorithms are less complex than ILP formulations but do not present the optimal solution. They have the advantage of completing faster then ILP and require less computing power. A heuristic algorithm can be run whenever a new set of traffic is generated without the resource and time problems of ILP.

This work assumes that an adequate prediction algorithm is in place and focuses on the assignment of resources. The set of predicted demands, and the error in prediction, will increase the further into the future the prediction algorithm generates data for. The error is the difference between the predicted outcome and the actual outcome. To keep the error as small as possible, the demands should be predicted for a shorter period of time. The prediction algorithm can then be run whenever a new set of predicted demands is needed. The algorithms presented in this work find an energy efficient routing scheme for a predicted set of demands. When the predicted set of demands changes, the algorithms have to be re-run for the new set of predicted demands. Once the routes for the predicted demands are found, they are stored for later use as the actual demands enter

the network.

3.2.1 Power Efficient Routing Scheme

The path finding algorithms in this work find an energy efficient way to route a given set of demands. If the network traffic never changed over time, the energy efficient routing method would also be power efficient. However, as discussed in section 1.1, network traffic is bursty and changes over time. Once the network traffic changes, the most energy efficient way to route that traffic also changes. Over time, if the routing method does not change as the network traffic changes, the network will not perform in a power efficient way. Therefore, a power efficient routing scheme has to continuously change its routing table as the traffic changes. Operation of the power efficient routing scheme takes place in three stages:

- Prediction Stage
- Path Finding Stage
- Operation Stage

During the prediction stage, a prediction algorithm is run to determine the future demands. The prediction algorithm can be run in the background as the network is in the operation stage to save time and to utilize measured data to determine, and correct for, error that occurs with time. For short term (near future) predictions, care has to be taken to ensure that the prediction algorithm will predict traffic far enough into the future that there will be enough time to predict the next set of data. For long term (far future) predictions, care has to be taken to ensure that the time scale isn't so large that an unacceptable amount of error occurs. The ideal size of the time scale is dependent on the individual network and beyond the scope of this work. Therefore, the traffic for the simulations was generated offline.

During the path finding stage, the routing algorithms are run and the paths for the predicted traffic are found. These paths are then stored for use during the operation stage when the real demands enter the network. These paths could be stored in a look-up table for each source destination pair. When a demand enters the network, the routing decision becomes a matter of looking up which paths to use from the look-up table. Alternatively, just the cycles that will provide the paths for the demands can be stored in the look-up table. However, this

would require a path finding algorithm to find a set of paths for the demands as they enter the network during the operation stage.

The operation stage is when the paths assigned during the path finding stage are utilized for real traffic. When a demand enters the network, the look-up table found during the path finding stage is used to find which paths should be used for the demand in question. The operation stage will continue until the end of the prediction time scale is reached. Once this occurs it is necessary to obtain a new set of demand predictions and a new set of find paths for the new demands. These new demands and their paths can be obtained while the old predictions and paths are being utilized. This enables the network to operate continuously without blocking all demands while a new set of predictions is made.

During operation, the network will experience high and low periods of traffic. These high and low periods are what affects the ideal size of the time scale of the demand prediction. An efficient set of paths for one load may not be efficient for another. The decision to change the routing table is dependent on how long the high or low period lasts and how bursty the traffic is. With very bursty traffic it would be better to just keep one routing table and not update it as the expected traffic changes. For example, for the predicted traffic in Figure 3.2.1, the routing table could be changed between points a and b since the expected traffic isn't very bursty and changes slowly. However, the traffic is very bursty and changes rapidly between points c and d so the routing table would be selected for the highest demand and left that way until the nature of the expected traffic changes. The decision on the minimum length of a high or low period to warrant a change in the routing table is dependent on the network in question and the decision of the length is left up to the designer.

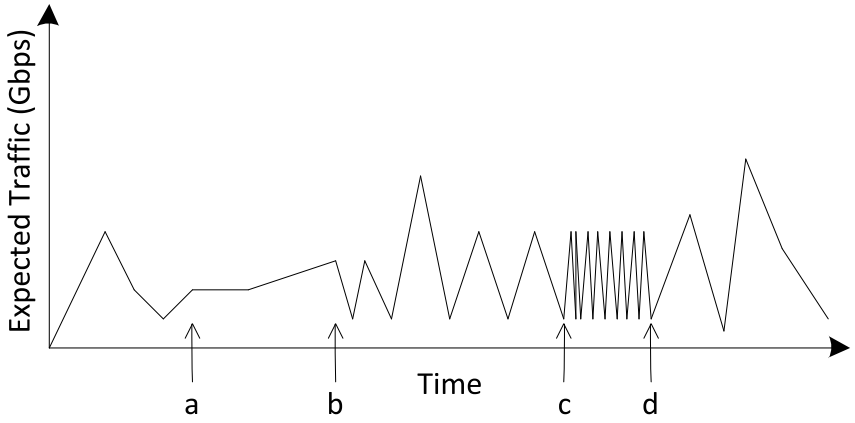

Figure 3.2.1: Decision to change the routing table

3.2.2 Energy Efficiency With P-Cycles

P-cycles are normally used for planning backup paths and sharing backup resources. Previous works have shown that p-cycles provide the restoration speed of rings and the resource efficiency of mesh networks. The resource efficiency comes from the cycles ability to protect both on-cycle and straddling links/paths. However, the p-cycle concept can provide power efficiency as well. By using p-cycles to provide both working and backup paths for traffic, the straddling links/paths are left with no traffic routed over them and can be switched into offline mode. For example, a network with two cycles is shown in Figure 3.2.2 (A). Three demands are shown in Figure 3.2.2 (B). If the working paths are routed around the cycles as shown in Figure 3.2.2 (C), and the backup paths are routed around the cycles as shown in Figure 3.2.2 (D), then 4 links and 1 node can be put into offline mode, and three links can be put into sleep mode. Note that the working paths of demands 1 and 2 are not disjoint so they cannot share backup resources over any backup links they share. However, Demand 3 is disjoint from Demands 1 and 2 so it can share backup resources with either Demand 1 or 2 over link 1-4. If the working paths are routed over the shortest paths as shown in Figure 3.2.2 (E), and the backup paths are routed over the shortest paths as shown in Figure 3.2.2 (F), then four links can be put into offline mode, one node can be put into sleep mode, and four links can be put into sleep mode. Components in offline mode will utilize less energy than components in sleep mode so routing the paths around the cycles like in Figure 3.2.2 (C) and (D) is more energy efficient than routing them over the shortest paths as in Figure 3.2.2 (E) and (F).

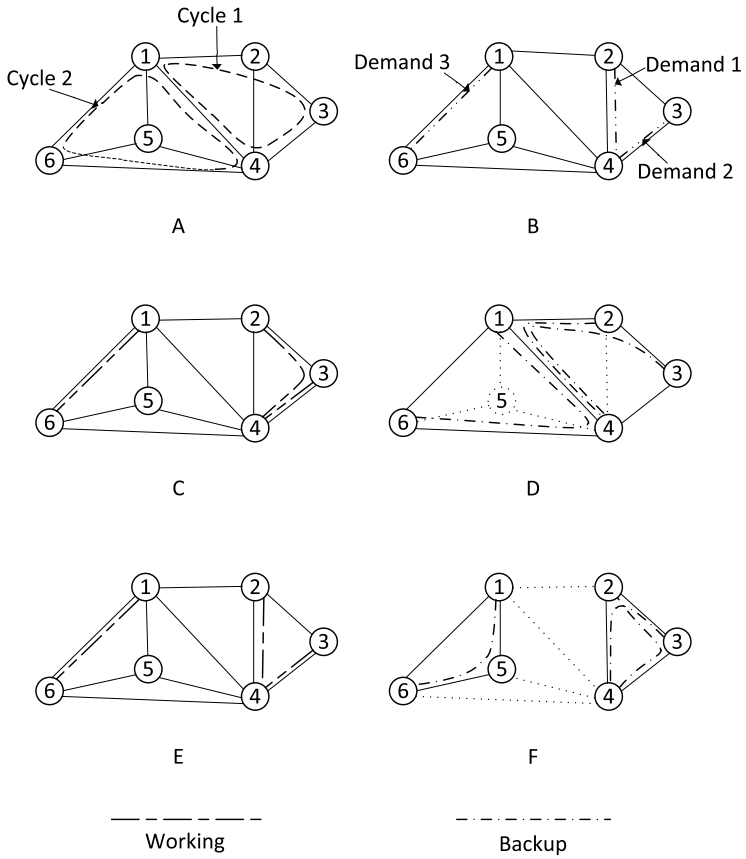

Figure 3.2.2: Use of straddling link and straddling path for power efficiency: (A) An example network with two cycles. (B) Three example demands. (C) Working paths for Demands 1, 2, and 3 using p-cycles. (D) Backup paths for Demands 1, 2, and 3 using p-cycles. (E) Shortest working paths for Demands 1, 2, and 3. (F) Shortest backup paths for Demands 1, 2, and 3.

The backup resource sharing ability of cycles can also be taken advantage of. The backup resources between cycles can be shared over common links provided there are no working paths over those links. If there are working paths over the common links, care has to be taken to ensure that the working paths of the demands are disjoint in order to share bandwidth. If the working paths are not disjoint, the backup paths could be shared but not the bandwidth. When a demand is assigned paths using a cycle, the demand is said to be assigned to that cycle. Figure 3.2.3 (A) shows an example network with two cycles with common links 1-5 and 4-5. A demand between nodes 2 and 4 is assigned to Cycle 1 and a demand between nodes 1 and 6 is assigned to Cycle 2. In Figure 3.2.3 (B), the working path for the demand assigned to Cycle 1 is routed over the common links. Because the working path for the demand assigned to Cycle 1 is over the common links, the demand assigned to Cycle 2 cannot share backup bandwidth with the demand assigned to Cycle 1. In Figure 3.2.3 (C), the backup path for

the demand assigned to Cycle 1 is routed over the common links. The backup path for the demand assigned to Cycle 2 is also routed over the common links. The demand assigned to Cycle 1 can share backup bandwidth with the demand assigned to Cycle 2. In Figure 3.2.3 (D), the working path for the demand assigned to Cycle 2 is over common link 1-5. If a demand assigned to Cycle 1 uses common link 1-5 as its working path, then the demand cannot share its backup bandwidth with the demand assigned to Cycle 2.

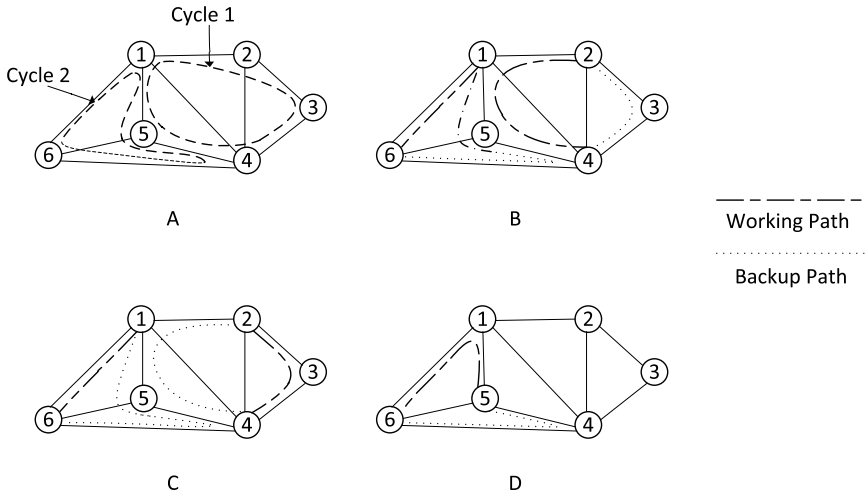

Figure 3.2.3: Resource Sharing Over Common Links: (A) An example network with 2 cycles, with common links 1-5 and 4-5. (B) The working path for a demand between nodes 2 and 4 are routed over the common links. (C) The backup paths for demands between nodes 2 and 4 and nodes 1 and 6 are over the common links 1-5 and 4-5. (D) Common link 1-5 is used for working traffic for a demand between nodes 5 and 6.

Path finding by using p-cycles is relatively simple. The cycles themselves are used to determine the working and backup paths so, unlike the traditional method of p-cycle protection which finds a working path first and then uses the cycles to find a backup path, the working and backup paths are both provided by the cycle. A cycle is said to be able to support a demand if both the source and destination nodes of that demand are on the cycle. Once a cycle that can support a demand is found, the working and backup paths for that demand can be determined by using the links in the cycle itself since only two paths are available on any given cycle between source and destination nodes. The complexity of the routing problem will grow with the number of cycles since it is possible for multiple cycles to be able to support a demand. This problem is solved by first scoring the cycles with an appropriate metric to limit the list of potential cycles that need to be checked when path finding.

3.2.3 Proposed Algorithms

Three algorithms are proposed in this work:

- Hybrid Shared
- Modified Hybrid Shared
- Power Efficient Growing Cycles

The Hybrid Shared and Modified Hybrid Shared algorithms assume that a traffic prediction algorithm is available and requires the demands to be known in advance. The Power Efficient Growing Cycles algorithm is a dynamic routing algorithm that will work in both systems with demand prediction and systems without demand prediction. When traffic prediction is being used, all three algorithms will work during the path finding stage. The algorithms will find an energy efficient way to route the predicted demands. During the operation stage, the assigned routes for the predicted demands will be used to assign routes to the real demands as they enter the network. In order to achieve power efficiency, the demand prediction has to be updated as traffic conditions in the network (i.e. the demands) change. By continuously updating the traffic prediction, and the paths for the predicted traffic, a power efficient routing scheme can be maintained over time and, therefore, power efficiency is achieved.

The power efficient growing cycles algorithm has the advantage of not requiring the traffic to be predicted before finding paths for a set of demands. When demand prediction is not being used, the network does not have to go through a prediction stage or a path finding stage, the algorithm will assign demands a working and backup path dynamically. This means that, when being operated dynamically, the power efficient growing cycles algorithm will continuously maintain an energy efficient routing scheme and therefore, achieves power efficiency. How the power efficient growing cycles algorithm assigns demands their paths and how it handles traffic already in the network when a new demand is being assigned a path is discussed in Section 3.2.3.3.

3.2.3.1 Hybrid Shared Algorithm

This algorithm derives its name from how it allocates backup bandwidth. It is similar to the shared backup scheme with a few changes. It assumes that backup bandwidth can be shared between demands of different cycles, provided the working paths are disjoint. Backup bandwidth between demands that are on the same cycle as can never be shared. So the algorithm will dedicate bandwidth to on cycle demands and it will allow demands to share bandwidth only with demands that are assigned to other cycles.

Stage 1:

In this stage, the algorithm will assign groups of demands from set DM to as many cycles from set CY as possible. CY is a set of all p-cycles for the network and is predetermined prior to operation of the algorithm, and DM is a set of all demands that need to be assigned paths. When finding a group of demands to assign to a cycle, it is necessary for the algorithm to perform the following two steps:

1. The cycles in set CY are scored.

2. The scored cycles are assigned demands.

In step 1, every cycle in set CY is assigned a score based on the size number of links in the cycle, and the number of demands that the cycle can potentially support. A cycle can 'potentially support' a demand if both the source and destination nodes of the demand are on the cycle. The score for a given cycle is calculated as shown in Equation 3.2.1, where NL_c is the number of links in cycle c, ND is the number of demands in set DM, and $D_{i,c}$, given in Equation 3.2.2, indicates if both the source and destination nodes of a demand are on cycle c.

$$S_c = \frac{\sum_{i=1}^{ND} D_{i,c}}{NL_c} \tag{3.2.1}$$

$$D_{i,c} = \begin{cases} 0 & \text{If the source or destination node of demand } i \text{ is not on cycle } c. \\ 1 & \text{If the source and destination nodes of demand } i \text{ are both on cycle } c. \end{cases} \tag{3.2.2}$$

Each demand that has a $D_{i,c} = 1$ is stored in set DS_c. Set DS_c, also called the demand set for cycle c, is a set of all demands that cycle c can potentially support.

Once each cycle is scored, they are assigned demands. The cycles are checked, one by one, from highest scoring cycle to lowest scoring cycle to determine if they can support the demands in their demand sets. The cycle can support the demands in its demand set if a cycle has enough bandwidth on each of its links to support the demands in its demand set. When determining if a cycle can support the demands in its demand set, the algorithm will:

1. Find the total bandwidth required by the demands in the cycles demand set that are not already assigned to another cycle.
2. Ensure there is enough bandwidth on each link that makes up the cycle for the demands in the cycles demand set.
3. If the cycle can support the demands in its demand set, reserve the necessary bandwidth for the demands in the cycles demand set on each link of the cycle.

When assigning demands to a cycle, it is necessary to ensure that there is enough bandwidth on the links in the cycle to support the demands. The total bandwidth required by the demands in a cycles demand set is given in Equation 3.2.4, where b_i is the bandwidth required for demand i, $B_{c,i}$ is the bandwidth required for demand i if it will be assigned to cycle c, BR_c is the total bandwidth required by all demands in the demand set of cycle c (DS_c), N_c is the number of demands in set DS_c, and BR_c, given in Equation 3.2.4, is the bandwidth required for a demand if it will be assigned to cycle c.

$$B_{c,i} = \begin{cases} b_i & \text{If demand } i \text{ is in both set } DS_c \text{ and set } DM. \\ 0 & \text{Otherwise.} \end{cases} \tag{3.2.3}$$

$$BR_c = \sum_{i \epsilon DS_c}^{N_c} B_{c,i} \tag{3.2.4}$$

Once the total bandwidth needed for the demands in a cycles demand set is determined, each link that makes up the cycle is checked to ensure that they have enough bandwidth to support the demands.

If all of the links that make up the cycle do not have enough free bandwidth for the demand, the next cycle is checked. If all of the links that make up the cycle have enough free bandwidth for the demand, the demands are assigned to the cycle. Three things occur when a demand is assigned to a cycle during this stage:

1. The demands that are in both the demand set of the cycle and set DM are stored in set $S1_c$ and the cycle is stored in set UC. Set $S1_c$ is called a Stage 1 demand set for cycle c and is a set of all demands assigned to cycle c during Stage 1. Set UC is a set of all cycles that have been assigned demands during Stage 1 and Stage 2.

2. All demands that are in both the demand set of the cycle and set DM are removed from set DM.

3. The total bandwidth required for the demands in the cycles demand set is subtracted from the free bandwidth on the cycles links. No paths are assigned during this stage. The path finding for the cycles found during this stage is performed during Stage 3. Because the backup bandwidth between demands on the same cycle cannot share bandwidth, the bandwidth necessary for the demands on each cycle can be reserved during this stage even though the paths are not yet known.

Stage 1 ends when all cycles in set CY have been checked or there are no demands left in set DM.

Stage 2:

If there are no demands in set DM when this stage starts, the algorithm moves on to Stage 3. If there are demands left in set DM, then the algorithm will attempt to assign the demands, one at a time, to the cycles in set CY.

For every demand in set DM, it is necessary to perform the following two steps:

1. The cycles in set CY are scored for the demand being considered (also known as the current demand).

2. Each cycle in set CY is checked, from highest score to lowest score, until a cycle that can provide a working and backup path for the current demand that has enough bandwidth on its links to provide a working path for the current demand is found. If there are no cycles in set CY with a score greater than 0, the current demand is rejected, removed from set DM, and the next demand in set DM is considered.

The score assigned to the cycles during this stage is based on the following four criteria:

1. The number of demands that the cycle can provide paths for.
2. The number of links in the cycle.
3. The number of common links the cycle has with cycles that have already been assigned demands. The cycle being scored is not included in this consideration since a cycle cannot have a common link with itself.
4. Whether or not the cycle has already been considered for the current demand.

The number of common links a cycle has with cycles that have already been assigned demands is given as shown in Equation 3.2.5, where TCL_c is the total number of common links that cycle c has with the cycles in set UC, NC is the number of cycles in set UC, NL_c is the number of links in cycle c, and $CL_{c,j,k}$, given in Equation 3.2.6, indicates if link j on cycle c is a common link with one of the links in cycle k from set UC.

$$TCL_c = \sum_{j=1}^{NL_c} \sum_{k=1}^{NC} CL_{c,j,k} \tag{3.2.5}$$

$$CL_{c,j,k} = \begin{cases} 1 & \text{If link } j \text{ from cycle } c \text{ is a common link with one of the} \\ & \text{links in cycle } k \text{ from set } UC \text{ and cycle } k \text{ is not cycle } c. \\ 0 & \text{Otherwise.} \end{cases} \tag{3.2.6}$$

The score for each cycle for a given current demand is calculated as shown in Equation 3.2.7, where $SS_{c,d}$ is the score assigned to cycle c for current demand d, D_i is given above in Equation 3.2.2, ND is the number of demands in set DM, NL_c is the number of links in cycle c, TCL_c is the total number of common links that cycle c has with the cycles in set UC, and F_d, also called the Failed set for cycle d, is a set containing all cycles that were already checked for demand d but did not support it. F_d is initially empty but can have cycles added to it during this stage and Stage 3.

$$SS_{c,d} = \begin{cases} \frac{\sum_{i=1}^{ND} D_i + TCL_c}{NL_c} & \text{If cycle } c \text{ is not in set } F_d \text{ and both the source and destination nodes} \\ & \text{for demand } d \text{ are on the cycle.} \\ 0 & \text{Otherwise.} \end{cases} \tag{3.2.7}$$

Once the cycles have been scored, each cycle is checked, from highest scoring cycle to lowest scoring cycle. All of the links in both of the possible paths that the cycle being considered can provide to the current demand are checked to see if one of them has enough free bandwidth to support the current demand. If neither of the two paths have enough bandwidth on each of its links for the current demand, the cycle being considered is added to the Failed set for the current demand and the next highest scoring cycle is checked. If one, or both, of the paths has enough bandwidth for the current demand, it is assigned to the cycle. Two things occur when a current demand is assigned to a cycle during this stage:

1. The demands stored in set $S2_c$ and the cycle is stored in set UC. Set $S2_c$ is called a Stage 2 Demand set for cycle c and is a set of all demands assigned to cycle c during Stage 2.

2. The current demand is removed from set DM.

The next demand in set DM is then considered. Stage 2 continues until all demands in set DM have been considered.

Stage 3:

During this stage, the paths and resources are allocated to the demands that were assigned to the cycles in set UC during Stage 1 and Stage 2. The demands assigned to cycles during Stage 1 are assigned bandwidth first.

Each cycle in set UC is checked to see if it has any demands stored in its Stage 1 Demand set. If a cycle from set UC that has demands in its Stage 1 Demand set is found, the demands are assigned paths in the network. Since the bandwidth necessary for the demands that were assigned to cycles during Stage 1 was checked during that stage, it is only necessary to assign a working and backup path to demands that were assigned to cycles during Stage 1.

For each demand in the Stage 1 Demand set for the cycle, the two paths that the cycle can provide for the demand are scored based on the number of links in the path and the number of common links the path has with cycles in set UC. The score for each path is given in equation 3.2.8, where $PS_{c,d,k}$ is the path score for path k of demand d that was assigned to cycle c, $L_{c,i}$ indicates if link i is a common link with any cycle in set UC but is not a part of cycle c, and NP_k is the number of links in path k.

$$PS_{c,d,k} = \frac{\sum_{i=i}^{NP_k} L_{c,i}}{NP_k} \tag{3.2.8}$$

$$L_{c,i} = \begin{cases} 0 & \text{If link } i \text{ is not a common link with any cycle in set } UC \text{ that is not assigned to cycle } c. \\ 1 & \text{If link } i \text{ is a common link with any cycle in set } UC \text{ that is not assigned to cycle } c. \end{cases} \tag{3.2.9}$$

The path with the most common links will have a higher Path Score and will be assigned as the backup path. The working path is the path with the lowest Path Score. Once a demand is assigned its paths, it is removed from the cycles Stage 1 Demand set, and the next demand is assigned its paths.

Once every cycle stored in set UC has been checked for demands stored in their Stage 1 Demand sets, the cycles are re-checked to see if they have any demands stored in their Stage 2 Demand sets. If a cycle from set UC that has demands in its Stage 2 Demand set is found, the algorithm attempts to assign paths to its demands.

The two possible paths that the cycle can provide to each demand is scored using Equation 3.2.8. The lowest scoring path becomes the working path candidate and the highest scoring path becomes the backup path candidate. The candidate paths are then checked to ensure that each of their links have enough bandwidth to support the demand.

If each of the links in the working path candidate has enough unused bandwidth for the demand, the working path candidate can support the demand. If one or more of the links in the working path candidate does not have enough bandwidth for the demand, the working path candidate cannot support the demand.

The bandwidth available on links in the backup path candidate is calculated as shown in equation 3.2.10, where $BB_{d,l}$ is the total backup bandwidth available on link l for demand d, FB_l is the amount of free bandwidth on link l, b_i is the amount of bandwidth required for demand i, and $BA_{d,i,l}$, which is given in Equation (3.2.11), is the amount of bandwidth available on link l for sharing with demand d, and M_l is the number of demands that have link l as a part of their backup path.

$$BB_{d,l} = FB_l + \sum_{i=1}^{M_l} BA_{d,i,l} \tag{3.2.10}$$

$$BA_{d,i,l} = \begin{cases} b_i & \text{If demand } i \text{ has link } l \text{ in its backup path, is assigned to a different cycle than demand } d, \text{ and has a working path that is disjoint from the working path candidate of demand } d. \\ 0 & \text{Otherwise.} \end{cases} \tag{3.2.11}$$

If there is enough bandwidth available on all links in the backup path candidate for the demand, the backup path candidate can support the demand. If one of more of the links in the backup path candidate does not have enough bandwidth for the demand, the backup path candidate cannot support the demand.

If both the working path candidate and the backup path candidate can support the demand, the demand is accepted onto the candidate paths. When the demand is accepted, it is removed from the cycles Stage 2 Demand set, assigned its paths and bandwidth on the network, and the next demand in the cycles Stage 2 Demand set is checked.

If either the working path candidate or the backup path candidate cannot support the demand, the working and backup path candidates are swapped and checked again to see if they will support the demand. If both candidate paths can support the demand, the demand is accepted onto the candidate paths. If one of the candidates cannot support the demand, the demand is removed from the cycles Stage 2 Demand set and added set DM.

Once all of the demands in the cycles Stage 3 demand set have been checked, the next cycle in set UC is checked. Stage 3 ends once all the cycles in set UC have been checked. If there are any demands in set DM when Stage 3 completes, set UC is emptied and Stage 2 is run again to find a new set of cycles that could potentially provide paths for the demands. If set DM is empty when Stage 3 completes, all demands have either been rejected during Stage 2 or accepted onto the network during Stage 3 and the algorithm completes.

3.2.3.2 Modified Hybrid Shared Algorithm

This algorithm is a simplified version of the Hybrid Shared algorithm. The Hybrid Shared algorithm assigns demands to cycles prior to path and resource assignment. However, there is no consideration of how much bandwidth each demand uses when it is assigned to links and paths during Stage 2 of the Hybrid Shared algorithm and it is possible for a cycle to be assigned more demands than its links have bandwidth to support. Stage 3 of the Hybrid Shared algorithm deals with the problem by checking the links in the path to ensure they have enough bandwidth to support each demand and, if the links do not have enough bandwidth, return the demand to set DM so it can be assigned another cycle or rejected if all cycles that could provide a working and backup path to the demand have been considered. Therefore, Stage 2 and Stage 3 of the Hybrid Shared algorithm will run multiple times, leading to an algorithm that takes a long time to solve. Stage 1 of the Hybrid Shared algorithm can also lead to bandwidth inefficiencies. Resources are subtracted from all of links in a cycle when a demand is assigned to it. Therefore, demands assigned to cycles in Stage 1 are assigned resources as if dedicated backup path protection is being used. Since sharing isn't considered between demands assigned to cycles during Stage 1, more bandwidth is used for backup paths than is necessary. The Modified Hybrid Shared algorithm eliminates the time and bandwidth problems of the Hybrid Shared algorithm by only scoring the cycles once and then assigning paths to each demand, one at a time. The Modified Hybrid Shared algorithm also assigns the shortest of the two paths provided by a cycle to the working path.

The Modified Hybrid Shared algorithm has two stages. In Stage 1, the cycles in set CY that will provide paths to each demand in set DM are scored. In Stage 2, each of the demands is assigned to a cycle, the working and backup paths are determined, and the bandwidth is assigned. Set CY is a set of all p-cycles for the network and is predetermined prior to operation of the algorithm, and set DM is a set of all demands that need to be assigned paths.

Stage 1:

Each cycle in set CY is assigned a score based on the number of links in the cycle and the number demands in set DM the cycle can provide paths for. The score for a given cycle is calculated as shown in Equation 3.2.12, where NL_c is the number of links in cycle c, ND is the number of demands in set DM, and $D_{i,c}$, given in Equation 3.2.13, indicates if both the source and destination nodes of a demand are on cycle c.

$$S_c = \frac{\sum_{i=1}^{ND} D_{i,c}}{NL_c} \tag{3.2.12}$$

$$D_{i,c} = \begin{cases} 0 & \text{If the source or destination node of demand } i \text{ is not on cycle } c. \\ 1 & \text{If the source and destination nodes of demand } i \text{ are both on cycle } c. \end{cases} \tag{3.2.13}$$

Each demand that has a $D_{i,c} = 1$ is stored in set DS_c. Set DS_c, also called the demand set for cycle c, is a set of all demands that cycle c can potentially support.

Stage 1 ends once all cycles in set CY have been assigned scores.

Stage 2:

During this stage, the demands are assigned paths and resources on the network. For each demand, the cycles are checked from highest scoring cycle to lowest scoring cycle until a cycle is found that has both the source and destination nodes for the demand. This cycle, called a candidate cycle, is then checked to see if it can support the demand.

The cycle can support a demand if it has enough bandwidth on its links to provide both a working path and a backup path for the demand. The two paths the cycle can provide are found and the shortest path is considered the working path candidate and the longest path is considered the backup path candidate. The candidate paths are then checked to ensure that each of their links have enough bandwidth to support the demand. If each of the links in the working path candidate has enough unused bandwidth for the demand, the working path candidate can support the demand. If one or more of the links in the working path candidate does not have enough bandwidth for the demand, the working path candidate cannot support the demand.

Because the backup bandwidth can be shared among demands that are assigned to differing cycles, the bandwidth available on links in the backup path candidate is higher than the working path candidate. The bandwidth available on links in the backup path candidate is calculated as shown in equation 3.2.14, where $BB_{d,l}$ is the total backup bandwidth available on link l for demand d, FB_l is the amount of free bandwidth on link l, b_i is the amount of bandwidth required for demand i, and $BA_{d,i,l}$, which is given in Equation (3.2.15), is

the amount of bandwidth available on link l for sharing with demand d, and M_l is the number of demands that have link l as a part of their backup path.

$$BB_{d,l} = FB_l + \sum_{i=1}^{M_l} BA_{d,i,l} \tag{3.2.14}$$

$$BA_{d,i,l} = \begin{cases} b_i & \text{If demand } i \text{ has link } l \text{ in its backup path, is assigned to a different cycle than demand } d, \text{ and has a working path that is disjoint from the working path candidate of demand } d. \\ 0 & \text{Otherwise.} \end{cases} \tag{3.2.15}$$

If there is enough bandwidth available on all links in the backup path candidate for the demand, the backup path candidate can support the demand. If one of more of the links in the backup path candidate does not have enough bandwidth for the demand, the backup path candidate cannot support the demand.

If both the working path candidate and the backup path candidate can support the demand, the demand is accepted. When the demand is accepted, it is removed from set DM, the bandwidth on its working path candidate and backup path candidate are assigned to the demand, and the next demand is checked.

If either the working path candidate or the backup path candidate cannot support the demand, they are switched with each other. The working path candidate becomes the longer path and the backup path candidate becomes the shorter path. The paths are then checked again to see if they have enough bandwidth on each of their links to support the demand.

If both the working path candidate and the backup path candidate can support the demand, the demand is accepted, removed from set DM, and the bandwidth on its working path candidate and backup path candidate are assigned to the demand. If either the working path candidate or the backup path candidate cannot support the demand, the demand is rejected and removed from set DM, and the next demand is checked.

The algorithm completes set DM is empty. When set DM is empty, all demands have either been assigned paths and bandwidth on the network or rejected.

3.2.3.3 Power Efficient Growing Cycles

The Power Efficient Growing Cycles algorithm operates differently from the Hybrid Shared and Modified Hybrid Shared algorithms. It can be operated in both a static (predicted demands) and dynamic (no prediction so first come, first serve) situation. Each time a demand enters the network, this algorithm is run to assign the demand a set of paths and bandwidth on the network. If multiple demands enter the network at once, or a list of predefined demands exists, the demands are considered one at a time and the algorithm will be run for each. The algorithm goes through three steps when assigning paths and bandwidth to a demand:

1. The current cycles, which are cycles that have demands assigned to them already, are checked to see if any of them can support the demand. A demand can be supported by a cycle if the cycle can provide both a working and backup path for the demand and the links in each path have enough bandwidth for the demand.

2. If no current cycle can support the demand, the algorithm checks the current cycles to see if any of them can be grown to be able to accept the demand.

3. If no current cycle can accept the demand and no current cycle can be grown to accept the demand, the algorithm looks for the smallest new cycle that can accept the demand.

Step 1:

When a new demand enters the network, the algorithm will search through set CR to see if any of the cycles stored in the set can support the new demand. Set CR is a set of all cycles that have been assigned demands. In order to support a new demand, a current cycle has to meet the following two criteria:

1. Both the source and destination nodes of the new demand have to be on the current cycle.

2. The links in the current cycle have to have enough bandwidth for the new demand.

Both the source and destination nodes of the demand have to be on the current cycle. If the source and destination nodes are not on the current cycle, the current cycle cannot provide paths for the new demand and the next cycle in set CR is checked. If both the source and destination nodes are on the current cycle, the current

cycle can provide a working path candidate and a backup path candidate for the new demand. The shortest path the current cycle can provide becomes the working path candidate and the longest path the current cycle can provide becomes the backup path candidate. Both the working and backup path candidates are then checked to see if they have enough bandwidth for the new demand.

The working path candidate can support the new demand if each link on the working path candidate has to have enough unused bandwidth to dedicate to the new demand. The links in the backup path candidate have more bandwidth available for the new demand than the working path candidate. This is because demands can share backup bandwidth among each other. The amount of bandwidth available for the new demand over each of the links in its backup path candidate is shown in Equation 3.2.16, where $BB_{d,l}$ is the total backup bandwidth available on link l for demand d, FB_l is the amount of free bandwidth on link l, b_i is the amount of bandwidth required for demand i, and $BA_{d,i,l}$, which is given in Equation 3.2.17, is the amount of bandwidth available on link l for sharing with demand d, and M_l is the number of demands that have link l as a part of their backup path.

$$BB_{d,l} = FB_l + \sum_{i=1}^{M_l} BA_{d,i,l} \tag{3.2.16}$$

$$BA_{d,i,l} = \begin{cases} b_i & \text{If demand } i \text{ has link } l \text{ in its backup path, is assigned to a different cycle than demand } d, \text{ and has a working path that is disjoint from the working path candidate of demand } d. \\ 0 & \text{Otherwise.} \end{cases} \tag{3.2.17}$$

The backup path candidate can support the new demand if each of its links have enough bandwidth for the new demand. If both the working path candidate and the backup path candidate can support the new demand, the new demand is accepted, the bandwidth on its working path candidate and backup path candidate are assigned to the new demand, and the algorithm completes.

If either the working path candidate or the backup path candidate cannot support the new demand, they are switched with each other. The working path candidate becomes the longer path and the backup path candidate becomes the shorter path. The paths are then checked again to see if they have enough bandwidth on each

of their links to support the new demand. If both the working path candidate and the backup path candidate can support the new demand, the new demand is accepted, the bandwidth on its working path candidate and backup path candidate are assigned to the new demand, and the algorithm completes. If either the working path candidate or the backup path candidate cannot support the new demand, the algorithm continues to look through set CR.

If all cycles in set CR have been checked, and no current cycle that can support the demand has been found, the algorithm moves to Step 2.

Step 2:

During this stage, the current cycles are checked to see if any can be grown to provide paths for the new demand. If either the source or the destination node of a new demand, but not both, is on a current cycle, it is possible to grow the cycle to include the missing node. For example, if the source node of a new demand is on a current cycle but the destination node is not, the current cycle may be grown, by adding links, to include the destination node of the new demand.

The algorithm searches through set CR until a current cycle with either the source or destination node for the new demand, but not both, is found. Set CY, a set of all possible cycles in the network, is then searched for all cycles that meet the following criteria:

- Cycle contains both the source and destination nodes of the new demand.
- Cycle contains at least one of the links in the current cycle.

Each cycle that meets the above two criteria is added to set GC. Set GC is a set of all cycles that can be grown from a current cycle.

Once all cycles in set CY have been checked, the algorithm then goes through the cycles in set GC to check if the current cycle being considered (called the original cycle) can be replaced by one of the cycles in set GC.

Two conditions need to be met when switching growing the original cycle to the grown cycle:

- Only the backup paths for the demands assigned to the original cycle can be changed.

- The grown cycle needs to have enough bandwidth on all of its links for all of the demands that were assigned to the original cycle and the new demand.

Only the backup path can be changed when moving demands onto a new cycle. All of the working path links for the demands assigned to the original cycle have to be on the grown cycle so that moving the demands will not affect service to the customers. If all of the working path links of all of the demands assigned to the original cycle are not in the grown cycle, the next cycle in GC is checked. If the working path links of all of the demands on the original cycle are in the grown cycle, the demands could possibly be moved onto the grown cycle and the first condition is met.

If the grown cycle has enough bandwidth on its links for all of the demands assigned to the current cycle and also the new demand, the demands assigned to the current cycle can be moved onto the grown cycle and the new demand can be accepted onto the cycle. To check if enough bandwidth is available on the grown cycle, the two paths that the grown cycle can provide to the new demand are found first. The shortest path is considered the working path candidate and the longest path is considered the backup path candidate.

The working path candidate can support the new demand if there is enough unused bandwidth on each of its links to dedicate to the new demand. The working path candidate cannot support the new demand if there is not enough bandwidth on each of its links to dedicate to the new demand.

The working paths for the demands assigned to the original cycle are not considered since they are already accepted on their working path links. They are not changed and already assigned resources so it is unnecessary to check them when checking the bandwidth of the grown cycle.

To calculate the backup bandwidth necessary on the grown cycle, the backup path candidates for each of the demands assigned to the current cycle have to be found. The working path candidate of the demands assigned to the original cycle are obviously the working paths that are already assigned to the demands. Therefore, the backup path candidates are the second path the grown cycle can provide to each of the demands.

Once the backup path candidates for the demands assigned to the original cycle are found, the backup bandwidth needed on each of the links of the grown cycle is determined. The backup bandwidth required on a

given link on the grown cycle is then given as shown in Equation 3.2.18, where BG_l is the backup bandwidth needed on link l, BN is the amount of bandwidth needed by the new demand, DC is the number of demands that are assigned to the original cycle, and $BC_{i,l}$ is the amount of bandwidth needed for demand i on link l and is given in Equation 3.2.19, where b_i is the bandwidth needed for demand i.

$$BG_l = BN + \sum_{i=1}^{DC} BC_{i,l} \tag{3.2.18}$$

$$BC_{i,l} = \begin{cases} b_i & \text{If link } l \text{ is in the backup path candidate for demand } i \text{ and not in the original} \\ & \text{backup path for demand } i. \\ 0 & \text{Otherwise.} \end{cases} \tag{3.2.19}$$

Once the bandwidth needed on each link is calculated, the total backup bandwidth available on each link is calculated. The demands assigned to the current cycle and the new demand are stored in set DG. Set DG is a set of all demands that are going to be assigned to the grown cycle. The amount of backup bandwidth on each of the links is shown in Equation 3.2.20, where $BR_{d,l}$ is the total backup bandwidth available on link l for the demands in set DG, FB_l is the amount of free bandwidth on link l, b_i is the amount of bandwidth required for demand i, and $BL_{i,l}$, which is given in Equation 3.2.21, is the amount of bandwidth available on link l for sharing with demand d, and M_l is the number of demands that have link l as a part of their backup path.

$$BR_{l,d} = FB_l + \sum_{i=1}^{M_l} BL_{i,l,d} \tag{3.2.20}$$

$$BL_{d,i,l} = \begin{cases} b_i & \text{If demand } i \text{ has link } l \text{ in its backup path and has a working path that} \\ & \text{is disjoint from the working path candidates of all of the demands in} \\ & \text{set } DG. \\ 0 & \text{Otherwise.} \end{cases} \tag{3.2.21}$$

If the total backup bandwidth available on the grown cycle less than the amount of bandwidth needed on any of the links in the grown cycle, the next cycle in set GC is checked. If the total backup bandwidth available on the grown cycle is greater than the amount of bandwidth needed on every link in the grown cycle, the new demand is accepted, the bandwidth on its working path candidate and backup path candidate are assigned to

the new demand, the demands assigned to the original cycle are moved to the grown cycle, the original cycle is removed from set CR, the grown cycle is added to set CR, and the algorithm completes.

To move the demands from the current cycle to the grown cycle, the demands are removed from the original backup paths that were assigned by the original cycle, and then the demands are assigned bandwidth on the backup path candidates that the grown cycle can provide.

If either the working path candidate or the backup path candidate cannot support the new demand, they are switched with each other. The working path candidate becomes the longer path and the backup path candidate becomes the shorter path. The paths are then checked again to see if they have enough bandwidth on each of their links to support the new demand. If both the working path candidate and the backup path candidate can support new cycle and the demands assigned to the current cycle, the new demand is accepted, the bandwidth on its working path candidate and backup path candidate are assigned to the new demand, the demands assigned to the current cycle are moved to the grown cycle, the current cycle is removed from set CR, and the grown cycle is added to set CR, and the algorithm completes. If either the working path candidate or the backup path candidate cannot support the new demand, the algorithm continues to look through set GC.

If all cycles in set GC have been checked, the next cycle in set CR is considered. If all cycles in set CR have been checked, and no current cycle can be grown to support the new demand, the algorithm moves to Step 3.

Step 3:

If no current cycle can accept the new demand, and no current cycle can be grown to accept the new demand, the algorithm looks for the smallest new cycle that can support the new demand. Each cycle in set CY is checked to see if any of them can support the new demand. The cycles in CY are assigned scores based on if they have been checked already in the previous two steps and the number of links in each cycle. The cycle score for each cycle is given in Equation 3.2.22, where GS_c is the cycle score for cycle c, NL_c is the number of links in cycle c, and CR is a set of all cycles that are already assigned demands.

$$GS_c = \begin{cases} NL_c & \text{If cycle } c \text{ is not in set } CR. \\ 0 & \text{Otherwise.} \end{cases} \tag{3.2.22}$$

The cycles are then checked, from lowest scoring cycle to largest scoring cycle, to find a cycle that can support

the new demand. In order to support a new demand, a cycle has to meet the following two criteria:

1. Both the source and destination nodes of the new demand have to be on the cycle.
2. The links in the cycle have to have enough bandwidth for the new demand.

If both the source and destination nodes of the new demand are on a cycle, that cycle is called a candidate cycle. The candidate cycle is then checked to determine if it has enough bandwidth on each of its links to support the new demand. Out of the two paths the candidate cycle can provide for the new demand, the shortest path is considered the working path candidate and the longest path is considered the backup path candidate.

The working path candidate can support the new demand if it has enough unused bandwidth available on each of its links. If the working path candidate does not have enough unused bandwidth on each of its links, it cannot support the new demand.

The backup path candidate can support the new demand if it has enough backup bandwidth on each of its links for the new demand. If the backup path candidate does not have enough bandwidth on each of its links for the new demand, the backup path candidate cannot support the new demand. The amount of bandwidth available for the new demand over each of the links in its backup path candidate is shown in Equation 3.2.23, where $BB_{d,l}$ is the total backup bandwidth available on link l for demand d, FB_l is the amount of free bandwidth on link l, b_i is the amount of bandwidth required for demand i, and $BA_{d,i,l}$, which is given in Equation 3.2.24, is the amount of bandwidth available on link l for sharing with demand d, and M_l is the number of demands that have link l as a part of their backup path.

$$BB_{d,l} = FB_l + \sum_{i=1}^{M_l} BA_{d,i,l} \tag{3.2.23}$$

$$BA_{d,i,l} = \begin{cases} b_i & \text{If demand } i \text{ has link } l \text{ in its backup path, is assigned to a different cycle than demand } d, \text{ and has a working path that is disjoint from the working path candidate of demand } d. \\ 0 & \text{Otherwise.} \end{cases} \tag{3.2.24}$$

If both the working path candidate and the backup path candidate can support the new demand, the new

demand is accepted, the bandwidth on its working path candidate and backup path candidate are assigned to the new demand, the candidate cycle is added to set CR, and the algorithm completes.

If either the working path candidate or the backup path candidate cannot support the new demand, they are switched with each other. The working path candidate becomes the longer path and the backup path candidate becomes the shorter path. The paths are then checked again to see if they have enough bandwidth on each of their links to support the new demand.

If both the working path candidate and the backup path candidate can support the new demand, the new demand is accepted, the bandwidth on its working path candidate and backup path candidate are assigned to the new demand, the candidate cycle is added to set CR, and the algorithm completes. If either the working path candidate or the backup path candidate cannot support the new demand, the algorithm continues to look through set CY.

If all cycles in set CY have been checked, and no current cycle that can support the demand has been found, the demand is rejected and the algorithm completes.

Chapter 4

Simulation

4.1 Benchmark Algorithms

Two standard algorithms were used for comparison in this work. The Shared Backup Path Protection algorithm utilizes shared backup path protection and the Dedicated Backup Path Protection algorithm utilizes dedicated backup path protection. Path finding for the Shared Backup Path Protection algorithm is performed with the Dijkstra Least Cost Path algorithm and path finding for the Dedicated Backup Path Protection algorithm is performed with the Modified Dijkstra Least Cost Path algorithm.

4.1.1 Shared Backup Path Protection

The Shared Backup Path Protection algorithm uses the Dijkstra Least Cost Path algorithm to find two disjoint paths between the source and destination nodes. This algorithm has two steps.

Step 1:

Each link is checked to ensure that they have enough bandwidth to support the demand. If they do not, they are assigned a cost of infinity. If they do have enough bandwidth to support the demand, they are assigned a cost that depends on the network planners preference. For example, the cost could be dependent on the physical length of the link, or it could be set to 1 if the minimum hop path is desired, or it could be based on something more complex like a cost dependent on how much traffic is desired over each link. In this work, the cost was set

to 1 so that the paths with the shortest paths (minimum number of hops) are found. When the costs are set, the Dijkstra algorithm is run, and the least cost path is found.

Step 2:

The working path is found first. Each link is checked to ensure that they have enough bandwidth to support the demand. If there is enough bandwidth available, the link is assigned a cost of 1, if there isn't enough bandwidth available, the link is assigned a cost of ∞ (Equation 4.1.1).

$$C_j = \begin{cases} 1 & \text{If } b_i < Ba_j \\ \infty & \text{If } b_i > Ba_j \end{cases} \tag{4.1.1}$$

Where C_j is the cost of link j, b_i is the bandwidth demand of demand i, and c.

Once the costs are assigned to all of the links, the Dijkstra Least Cost Path algorithm is run to find the least cost path between source and destination nodes. This path is the working path of the demand. After the working path is found, the links are assigned a cost based on if there is enough bandwidth available for use by backup traffic. The bandwidth available for backup path links is equal to the sum of the total amount of bandwidth available to share and the total free bandwidth available over the link (Equation 4.1.2).

$$BWD_{d,j} = Ba_j + \sum_{i=1}^{D} BAD_{i,j} \tag{4.1.2}$$

Where $BWD_{d,j}$ is the bandwidth on link j for demand d that is assigned to demands that have disjoint working paths from demand d. Ba_j is the free bandwidth on link j, D is the number of demands on link j, and $BAD_{i,j}$ is the bandwidth of demands that use link j as a backup path and have working paths that are disjoint from demand d.

The cost of each link for backup path assignment is shown in Equation 4.1.3

$$C_j = \begin{cases} 1 & \text{If } b_i < BWD_{d,j} \\ \infty & \text{If } b_i > BWD_{d,j} \\ \infty & \text{If the link is one of the links in the working path demand } d. \end{cases} \tag{4.1.3}$$

After the costs are assigned, the Dijkstra algorithm is run again and the second path is found. The second path is the backup path for the demand.

4.1.2 Dedicated Backup Path Protection

The Dedicated Backup Path Protection algorithm used in this work uses the Modified Dijkstra algorithm to find two disjoint paths between the source and destination node in the network. The Dijkstra algorithm is useful for finding a single least cost path between any given pair of nodes. However, if the network contains any links with a negative cost, the algorithm can fail to find the shortest length path. Negative link cost can occur due to either the network design or as a result of simultaneous discovery of multiple disjoint paths.

When discovering multiple disjoint paths it is possible for the first path to block any further paths from being found. For example, if two disjoint paths between nodes A and E are desired in Figure 4.1.1 (A), The Dijkstra algorithm will first find path A-C-D-E as the least cost path. This path is then set to have a cost of infinity (Figure 4.1.1 (B)) and the algorithm will look for another least cost path. However, since there are no more paths available between nodes A and E, the algorithm has effectively blocked any further paths from being found. This is called the Trap Topology.

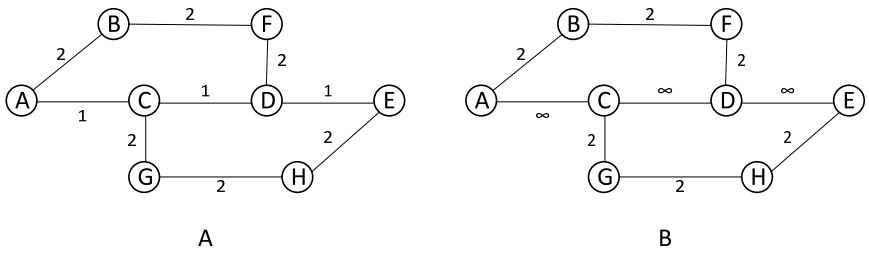

Figure 4.1.1: Example of Trap Topology

A way to avoid the trap topology, in the case of twin disjoint paths, is to find both paths at once rather than one at a time[41]. This is done by first finding two candidate paths (that may not be disjoint) and combining them into two new paths (that will be disjoint). The combination is performed by erasing the interlacing parts of the two candidate paths [41].

The dedicated path protection algorithm is a three step algorithm. Two paths between the source and

destination nodes are found and then combined to find two disjoint paths.

Step 1:

The links in the network are assigned a cost using the same cost function that was used used in the Shared Backup Path algorithm to find working paths, Equation 4.1.1 on page 62.

Step 2:

The first least cost path is found using the Modified Dijkstra Algorithm (see Section 4.1.4) and stored in set CP_1. CP_1 is a set of links in the first least cost path found.

Step 3:

The links in the first least cost path are assigned a negative cost and also converted into unidirectional links that flow toward the source node. A unidirectional link only allows traffic to flow in one direction. A second least cost path is then found using the Modified Dijkstra Algorithm (see Section 4.1.4) on the new graph and stored in set CP_2. CP_2 is a set of links in the second least cost path found.

Step 4:

The two paths stored in CP_1 and CP_2 are combined by removing the common links (this is exactly like the Straddling Link Method for finding p-cycles discussed in Section 2.1.1 on page 18 but applied to two sets of non-circular paths rather than circular paths like cycles are). CP_1 and CP_2 are searched and a new set, CP_3, is created. CP_3 is a set containing all the links in CP_1 and CP_2 that are not common between sets CP_1 and CP_2.

Just like a p-cycle, set CP_3 contains links for two unique paths. These unique paths are found using the Dijkstra algorithm. The two unique paths then become the working and backup paths for the demand. The shortest of the two paths is the working path and the longest is the backup path.

Example:

Consider the network in Figure 4.1.2 on the following page. Two disjoint paths between nodes A and E are desired. In Step 1 the algorithm would find the path: A-C-D-E as the shortest path (Figure 4.1.2 on the next page (A)). The links A-C, C-D, and D-E are made into unidirectional links that flow toward node A and have negative costs. The algorithm then finds a second least cost path: A-B-F-D-C-G-H-E (Figure 4.1.2 on the following page (B)). The two paths are then combined and the links that overlap are removed from the new set.

CP_3 = {A-B, B-F, F-D, D-E, A-C, C-G, G-H, H-E}(Figure 4.1.2 (C)). The two paths can be taken from CP_3 and are: A-B-F-D-E and A-C-G-H-E.

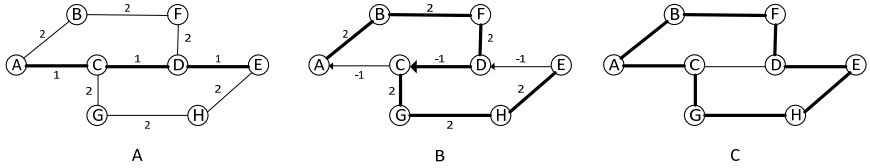

Figure 4.1.2: Finding Disjoint Paths With Modified Dijkstra Algorithm

4.1.3 Dijkstra Least Cost Path Algorithm

The Dijkstra Algorithm is a fast and efficient way of finding the least cost path from one node to any other node in the network. The algorithm will find the shortest route to each node until it reaches the destination node. The network is compiled into a graph that contains each node and each link. Each edge has an associated cost which is used while finding the path. The cost can be based on any cost such as the physical length of the path, the cost of transmission over that link, etc. The Dijkstra algorithm can also be used to find the minimum number of hops between a source and destination by setting every link to have the same cost. The following pseudo code (Algorithm 4.1), reproduced from [41], describes the basic operation of the Dijkstra Algorithm:

Algorithm 4.1 Dijkstra Least Cost Path Algorithm

Step 1: Start with $d(A) = 0$,
$d(i) = \begin{cases} l(Ai) & \text{if } i \epsilon \Gamma_A \\ \infty & \text{otherwise} \end{cases}$
Γ_i = set of neighbour nodes of node i, l_{ij} = length of the path from node i to node j.
Assign $S = V - \{A\}$, where V is the set of nodes in the given graph.
Assign $P(i) = A \forall i \epsilon S$.

Step 2: a) Find $j \epsilon S$ such that $d(j) = \min d(i)$, $i \epsilon S$.
b) Set $S = S - \{j\}$.
c) If $j = Z$ (the destination node), END; otherwise, go to Step 3.

Step 3: $\forall i \epsilon \Gamma_j$ and $i \epsilon S$, if $d(j) + l(ji) < d(i)$, set $d(i) = d(j) + l(ji)$, $P(i) = j$.
Go to Step 2.

$d(i)$is the cost of node $i(i \epsilon V)$ from the source node A. It is the sum of the costs of each link in the path from node A to node i. $P(i)$ is the predecessor of node i on the same path.

Example:

Consider the network shown in Figure 4.1.3 on the following page. If a least cost path between nodes A and G is desired, the Dijkstra algorithm would yield:

Step 1: $i = A$ $V = \{A, B, C, D, E, F, G\}$ $\Gamma_A = \{B, C, D\}$ $S = \{B, C, D, E, F, G\}$

i	$d(i) = l(Ai)$	$P(i)$
A	0	A
B	1	A
C	2	A
D	4	A
E	∞	A
F	∞	A
G	∞	A

Step 2: $j = B$ $S = \{C, D, E, F, G\}$

Step 3: $\Gamma_B = \{C, F\}$

i	$d(i) = d(B) + l(Bi)$	$P(i)$
C	$1 + 1 = 2$	-
F	$1 + 4 = 5$	B

Step 2: $j = C$ $S = \{D, E, F, G\}$

Step 3: $\Gamma_C = \{E\}$

i	$d(i) = d(C) + l(Ci)$	$P(i)$
E	$2 + 7$	C

Step 2: $j = D$ $S = \{E, F, G\}$

Step 3: $\Gamma_D = \{E, G\}$

i	$d(i) = d(D) + l(Di)$	$P(i)$
E	$4 + 1 = 5$	D
G	$4 + 2 = 6$	D

Step 2: $j = E$ $S = \{F, G\}$

Step 3: $\Gamma_E = \{F, G\}$

i	$d(i) = d(E) + l(Ei)$	$P(i)$
F	$5 + 2 = 7$	-
G	$5 + 6 = 11$	-

Step 2: $j = F$ $S = \{G\}$

Step 3: $\Gamma_F = \{G\}$

i	$d(i) = d(F) + l(Fi)$	$P(i)$
G	$5 + 6 = 11$	-

Step 2: $j = G$ $S = \{\}$; END

Least Cost Path: $\{A, D, G\}$

j	A		B		C		D		E		F		G	
i	$d(i)$	$P(i)$	$d(i)$	$P(i)$	$d(i)$	$P(i)$	$d(i)$	$P(i)$	$d(i)$	$P(i)$	$d(i)$	$P(i)$	$d(i)$	$P(i)$
A	0	A	0	A	0	A	0	A	0	A	0	A	0	A
B	1	A	1	A	1	A	1	A	1	A	1	A	1	A
C	2	A	2	A	2	A	2	A	2	A	2	A	2	A
D	4	A	4	A	4	A	4	A	4	A	4	A	4	A
E	∞	∞	∞	∞	9	C	5	D	5	D	5	D	5	D
F	∞	∞	5	B	5	B	5	B	5	B	5	B	5	B
G	∞	∞	∞	∞	∞	∞	6	D	6	D	6	D	6	D

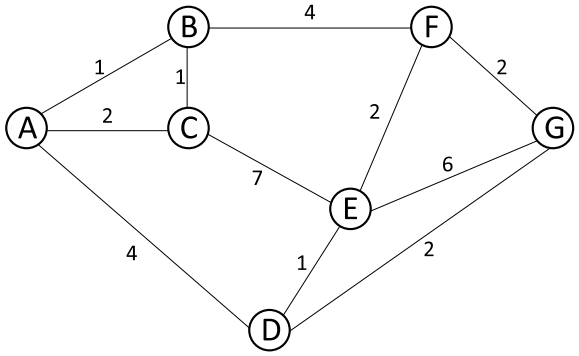

Figure 4.1.3: Dijkstra Example

4.1.4 Modified Dijkstra Least Cost Path Algorithm

The Modified Dijkstra Algorithm is a way of finding least cost paths in networks that have some negative cost links. Like the Dijkstra algorithm, this algorithm will find the shortest route to each node until it reaches the destination node. The network is compiled into a graph that contains each node and each link . Each link has an associated cost which is used while finding the path. The cost can be based on any cost such as the physical length of the path, the cost of transmission over that link, etc. The Modified Dijkstra algorithm can also be used to find the minimum number of hops between a source and destination by setting every link to have the same cost. If all of the links in the network and have non-negative costs, there will be no difference in operation between the Dijkstra and Modified Dijkstra algorithms. The following pseudo code (Algorithm 4.2), reproduced from [41], describes the basic operation of the Modified Dijkstra Algorithm:

Algorithm 4.2 Modified Dijkstra Least Cost Path Algorithm

Step 1: Start with $d(A) = 0$,
$d(i) = \begin{cases} l(Ai) & \text{if } i \epsilon \Gamma_A \\ \infty & \text{otherwise} \end{cases}$
Γ_i = set of neighbour nodes of node i, l_{ij} = length of the path from node i to node j.
Assign $S = V - \{A\}$, where V is the set of nodes in the given graph.
Assign $P(i) = A \forall i \epsilon S$.

Step 2: a) Find $j \epsilon S$ such that $d(j) = \min d(i)$, $i \epsilon S$.
b) Set $S = S - \{j\}$.
c) If $j = Z$ (the destination node), END; otherwise, go to Step 3.

Step 3: $\forall i \epsilon \Gamma_j$ if $d(j) + l(ji) < d(i)$
set $d(i) = d(j) + l(ji)$, $P(i) = j$.
set $S = S \cup \{i\}$.
Go to Step 2.

$d(i)$is the cost of node $i(i \epsilon V)$ from the source node A. It is the sum of the costs of each link in the path from node A to node i. $P(i)$ is the predecessor of node i on the same path.

Example:

Consider the network shown in Figure 4.1.4 on page 70. If a least cost path between nodes A and G is desired, the Modified Dijkstra algorithm would yield:

Step 1: $i = A$ $V = \{A, B, C, D, E, F, G\}$ $\Gamma_A = \{B, C, D\}$ $S = \{B, C, D, E, F, G\}$

i	$d(i) = l(Ai)$	$P(i)$
A	0	A
B	4	A
C	∞	A
D	3	A
E	∞	A
F	∞	A
G	∞	A

Step 2: $j = D$ $S = \{B, D, E, F, G\}$

Step 3: $\Gamma_D = \{E, G\}$ $S = \{B, D, E, F, G\}$

i	$d(i) = d(D) + l(Di)$	$P(i)$
E	$3 + 4 = 7$	D
G	$3 + 6 = 9$	D

Step 2: $j = E$ $S = \{B, C, F, G\}$

Step 3: $\Gamma_E = \{E\}$ $S = \{D, C, F, G\}$

i	$d(i) = d(E) + l(Ei)$	$P(i)$
C	$7 - 9 = -2$	E
D	$7 + 4 = 11$	-
F	$7 + 3 = 10$	E

Step 2: $j = C$ $S = \{B, D, F, G\}$

Step 3: $\Gamma_C = \{B, E\}$ $S = \{B, D, F, G\}$

i	$d(i) = d(C) + l(Ci)$	$P(i)$
B	$-2 + 3 = 1$	C

Step 2: $j = B$ $S = \{D, F, G\}$

Step 3: $\Gamma_B = \{F, G\}$ $S = \{D, F, G\}$

i	$d(i) = d(B) + l(Bi)$	$P(i)$
F	$1 + 3 = 4$	B

Step 2: $j = D$ $S = \{F, G\}$

Step 3: $\Gamma_D = \{G\}$ $S = \{E, F, G\}$

i	$d(i) = d(D) + l(Di)$	$P(i)$
E	$3 + 4 = 7$	-
G	$3 + 6 = 9$	-

Step 2: $j = F$ $S = \{E, G\}$

Step 3: $\Gamma_F = \{G\}$ $S = \{E, G\}$

i	$d(i) = d(F) + l(Fi)$	$P(i)$
E	$4 + 3 = 7$	-
G	$4 + 2 = 6$	F

Step 2: $j = G$ $S = \{\}$; END

Least Cost Path: $\{A, D, E, C, B, F, G\}$

j	A		D		E		C		B		D		F	
i	$d(i)$	$P(i)$	$d(i)$	$P(i)$	$d(i)$	$P(i)$	$d(i)$	$P(i)$	$d(i)$	$P(i)$	$d(i)$	$P(i)$	$d(i)$	$P(i)$
A	0	A	0	A	0	A	0	A	0	A	0	A	0	A
B	4	A	4	A	4	A	1	C	1	C	1	C	1	C
C	∞	A	∞	A	-2	E	-2	E	-2	E	-2	E	-2	E
D	3	A	3	A	3	A	3	A	3	A	3	A	3	A
E	∞	A	7	D	7	D	7	D	7	D	7	D	7	D
F	∞	A	∞	A	10	E	10	E	4	B	4	B	4	B
G	∞	A	9	D	9	D	9	D	9	D	9	D	6	F

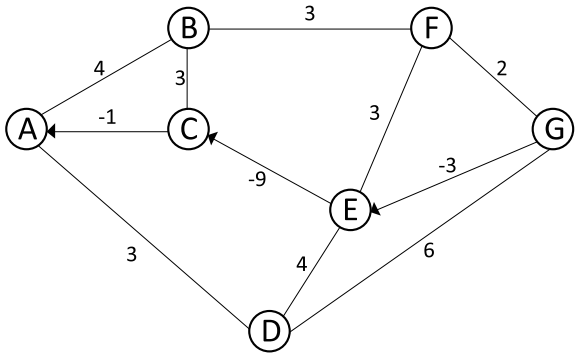

Figure 4.1.4: Modified Dijkstra Example

4.2 Simulation

4.2.1 Performance Metrics

The performance metrics compare the algorithms based on:

- Resource Usage
- Network Performance
- Energy Efficiency

The Resource Usage metrics are designed to illustrate the performance of the algorithms with respect to bandwidth efficiency. Measurements of average link load and path lengths (in number of hops) were taken to compare how the algorithms load the network and utilize the links. Of the two benchmark algorithms, the Shared Backup Path Protection algorithm (Section 4.1.1 on page 61) is the most bandwidth efficient, and the Dedicated Backup Path Protection algorithm (Section 4.1.2) is the least bandwidth efficient. The difference in bandwidth efficiencies of the two benchmark algorithms is due to the sharing of backup bandwidth between demands. It is possible for multiple demands to be protected by the same resources so less than 100% redundancy is necessary to protect the network from any single failure. In the case of dedicated protection, the network has to dedicate backup resources to each demand, and 200% redundancy is required to protect the network from any single failure. The energy efficient and power efficient algorithms will utilize more bandwidth than the Shared Backup Path Protection benchmark algorithm. However, since they also utilize shared backup path protection, they

should use less bandwidth than the Dedicated Backup Path Protection benchmark algorithm. Average link load, the average amount of bandwidth used by all of the demands assigned to the network, is used to compare the bandwidth efficiency of the algorithms and is defined in Equation 4.2.1, where N is the number of links in the network, BW_j is the used bandwidth on link j, and C_j is the total capacity of link j.

$$ALL = \frac{\sum_{j=1}^{N} BW_j}{\sum_{j=1}^{N} C_j} \tag{4.2.1}$$

The other Resource Usage metrics are measurements of the average length of the working and backup paths. These metrics illustrate the difference between the average lengths of the working and backup paths assigned to the demands by the algorithms. These measurements are important since, one of the tradeoffs of energy and power efficiency is an increased path length. An algorithm should be power/energy efficient but also not increase the path lengths too much. The average working and backup path lengths are defined in Equations 4.2.2 and 4.2.4 respectively, here N is the number of links in the network, TD is the number of demands in the network, $WPC_{d,i}$ indicates if link i is in the working path of demand d and is given in Equation 4.2.3, TD is the number of demands in the network, and $BPC_{d,i}$ indicates if link i is in the backup path of demand d and is given in Equation 4.2.5.

$$AWP = \frac{\sum_{d=1}^{TD}\sum_{i=1}^{N} WPC_{d,i}}{TD} \tag{4.2.2}$$

$$WPC_{d,i} = \begin{cases} 0 & \text{if link } i \text{ is not in the working path of demand } d. \\ 1 & \text{if link } i \text{ is in the working path of demand } d. \end{cases} \tag{4.2.3}$$

$$ABP = \frac{\sum_{d=1}^{TD}\sum_{i=1}^{N} BPC_{d,i}}{TD} \tag{4.2.4}$$

$$BPC_{d,i} = \begin{cases} 0 & \text{if link } i \text{ is not in the backup path of demand } d. \\ 1 & \text{if link } i \text{ is in the backup path of demand } d. \end{cases} \tag{4.2.5}$$

The Network Performance metric is a measurement of the amount of the demands that are rejected by the routing algorithm. Demand rejection is important since an algorithm that leads to a high number of rejected demands will not support as many customers as an algorithm that leads to a low number of rejected demands. The number of rejected demands will increase with the bandwidth usage of the algorithm. This means that algorithms that tend to use more bandwidth to assign a set of demands to a network will reject more demands than algorithms that use less bandwidth for the same set of demands. Therefore, the Dedicated Backup Path Protection benchmark algorithm will reject more demands than the Shared Backup Path Protection benchmark algorithm. The energy and power efficient algorithms should reject less demands than the Dedicated Backup Path Protection benchmark algorithm but will reject more than the Shared Backup Path Protection benchmark algorithm. The demand rejection metric is given as shown in Equation 4.2.6, where RD is the number of demands that were rejected and TDS is the total number of demands that needed to be assigned paths and bandwidth in the network.

$$DRM = \frac{RD}{TDS} \tag{4.2.6}$$

The Energy Efficiency metrics are used to compare the energy efficiency of each of the algorithms. As discussed above in Section 3.2, the algorithms will find an energy efficient solution for a given set of demands. These demands are provided by a demand prediction algorithm. Power efficiency is obtained by maintaining energy efficiency over time. An exact comparison of energy efficiency is difficult when using a simulation. It is dependent on the equipment used, if that equipment supports sleep mode or not, if the nodes support wavelength conversion or not, etc. However, by focusing on the three states of links and nodes, a relative comparison of the performance of the algorithms can be provided that is independent of the equipment used to build the network. As discussed above in Section 3.1, the three states of links and nodes are: Sleep mode, Online mode, and Offline mode. Online mode links and nodes will obviously use the most energy since the components are online and active. Offline mode links and nodes will use the least energy since the components are completely powered down or switched off. Sleep mode links and nodes will use an amount of energy that is between Online and Offline mode. Comparing the number of links and nodes that are online, offline, and in sleep mode between the algorithms will provide a good comparison of how the algorithms perform energy efficiency wise. The Energy Efficiency metrics used here are:

- Links in Sleep Mode

- Links Offline
- Nodes in Sleep Mode
- Nodes Offline

Each of these metrics is simply the number of links/nodes that are in offline/sleep mode. All of the metrics are plotted against Traffic Demand in graphs. Traffic Demand was chosen because the performance of each algorithm, on the same set of demands, was being compared. The Traffic Demand will be the same for each of the algorithms since the demand for each network load was used for each algorithm in each simulation, thus providing a common variable for use in comparison of the algorithms. The network load, measured in Gb/s, is the total amount of bandwidth that each set of demands required and is shown in Equation 4.2.7, where TDS is the number of demands that needed to be assigned paths and bandwidth in the network and b_i is the bandwidth required for demand i.

$$TRD = \sum_{i=1}^{TDS} b_i \tag{4.2.7}$$

4.2.2 Test Demands and Networks

Three test networks were used to compare the performance of all of the algorithms:

- Global Crossing Network
- Kaleidoscope Network
- Random Layout Network

The Global Crossing Network is based on the North American portion of the Global Crossing Network. The network, shown in Figure 4.2.1, consists of 27 nodes, with an average nodal degree of 2.6, and 38 links. The Global Crossing network was selected to compare the algorithms performance on a real topology. The Shared Backup Path protection algorithm was implemented first. The network was then loaded to a link load of 0.8 in steps of 0.1. These demands were stored and used with each of the other algorithms. Demands were randomly assigned a bandwidth of either 200Gb/s, 100Gb/s, 50Gb/s, 20Gb/s, or 10Gb/s. These demands had a random

source and destination node assigned to them that could be any node in the network except for a small group of "forbidden nodes" which are shown in grey in Figure 4.2.1. If every node could potentially be a source or a destination node, then it would be possible to have every node in Online mode since a node that is a source or a destination for traffic must be online to transmit/receive the traffic. However, at low traffic periods, it is possible that some nodes will not be either a source or destination node for demands. Therefore, it was necessary to insure some nodes would not be a source or destination node and could be turned off.

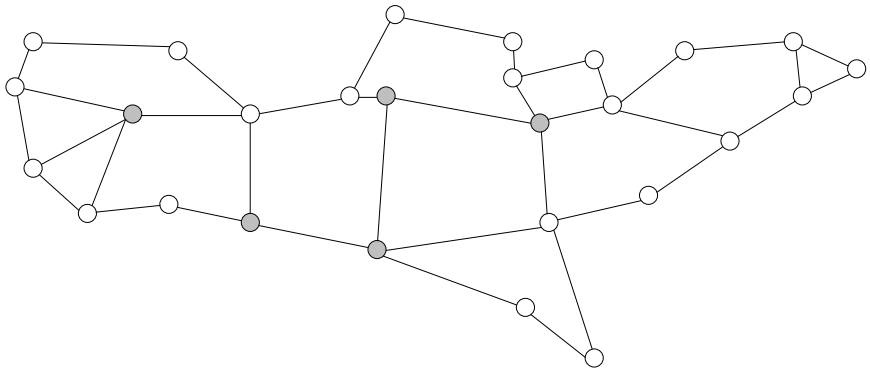

Figure 4.2.1: Global Crossing Network

The Kaleidoscope Network derives its name from its topology and how the network is used in the simulation. Three simulations with identical demands were run and the network topology itself was altered instead of the demands in order to better illustrate how the three energy efficient algorithms, operate relative to each other. Each of the topologies of the network are shown in Figure 4.2.3 on the next page. The way the network changes as the nodes and links are added, is similar to a kaleidoscope image in appearance.

With self similar traffic, at low points in network load, it is possible for traffic to become isolated in small pockets of nodes in a network as shown in Figure 4.2.2 (A). Traffic is isolated to nodes 1, 2, 3 and 4, 5, and 6. Even in a network where traffic between most of the nodes will exist, even in low load conditions, it is possible for the majority of traffic to become isolated in small pockets of nodes in a network as shown in Figure 4.2.2 (B).

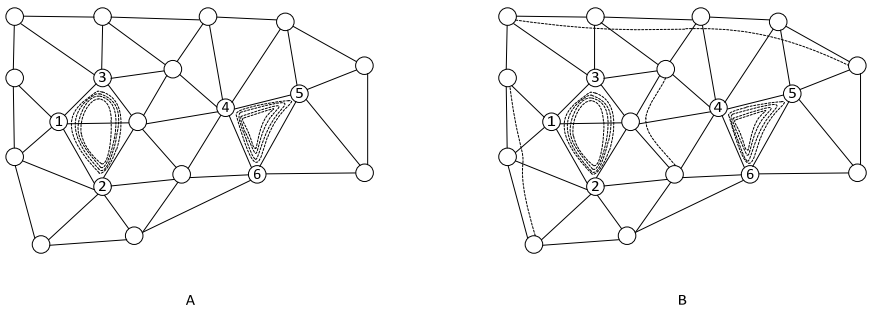

Figure 4.2.2: Example of traffic isolation: (A) Example network with isolated pockets of traffic shown in dashed lines. (B) Example network with majority of traffic in isolated pockets and some other demands shown in dashed lines.

This simulation will create a small pocket of nodes with varying amounts of traffic between them. Each network topology will add more nodes and links to the network, but the traffic will remain unchanged so it will be between the nodes that were in the original topology. The demands, each requiring the same amount of bandwidth, were assigned random sources and destinations. Demands were added into the First Network (Figure 4.2.3 (A)) and assigned paths by the Shared Backup Path Protection algorithm until it was loaded to 0.8 link load. This set of demands was then used on the second (Figure 4.2.3 (B)) and third (Figure 4.2.3 (C)) configurations.

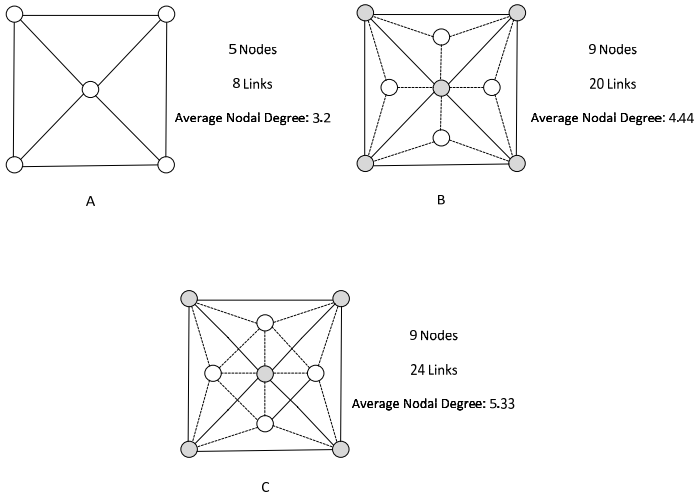

Figure 4.2.3: Kaleidoscope Network: (A) First Configuration. (B) Second Configuration with added nodes shown in white and added links shown as dashed lines. (C) Third Configuration with added nodes shown in white and added links shown as dashed lines.

The Random Layout Network, shown in Figure 4.2.4, is a random topology that consists of 19 nodes, with an average nodal degree of 3.6, and 38 links. This network is a random topology with a higher degree of connectivity

than the Global Crossing Network. This network was selected to greater illustrate the effect of the algorithms on networks with a high degree of connectivity. With very little connectivity, there is a limited number of paths between any given source and destination node. This means that the network will become 'saturated' with working paths (working paths assigned to every link and node in the network) at lower traffic loads than a network with a higher degree of connectivity would. The higher connectivity network will have many more potential paths between any source and destination pair. Thus, more opportunities to route working traffic around nodes will arise. The network test demands were generated using the same method as for the Global Crossing Network discussed above. The forbidden nodes are shown in grey in Figure 4.2.4.

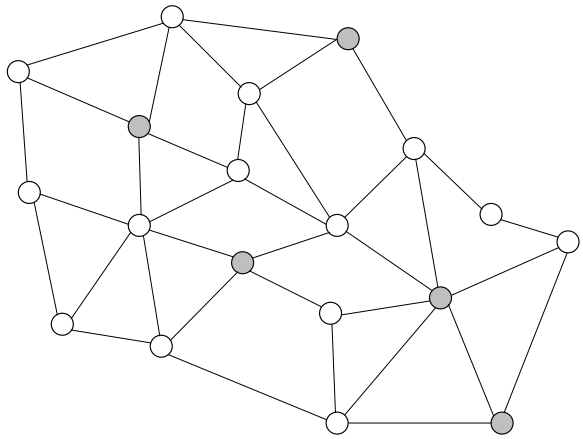

Figure 4.2.4: Random Layout Network

For each of the simulations, the length of each link is assumed to be the same. This will affect the total size of any given cycle (measured in links instead of a unit of distance) but it will have a negligible effect on the performance of the algorithms. Each node has full conversion/regeneration capabilities, but it is assumed in these simulations that the path length is short enough that regeneration is unnecessary.

4.3 Results

The results of the simulations run on each of the test networks is presented and discussed in this section. Four simulations were run on each test network. Each simulation had an individual set of demands and the average of the results is provided in each of the graphs. The experiments were run to compare the energy efficiency of each of the algorithms under different levels of traffic demand.

4.3.1 Global Crossing Network

The average link load for each algorithm is shown in Figure 4.3.1. At low network loads, the Power Efficient Growing Cycles and Modified Hybrid Shared algorithms load the links more than the Dijkstra Shared Backup Paths algorithm but less than the Dijkstra Dedicated Paths algorithm. This behaviour, discussed above in Section 4.2.1, is expected when using energy efficient routing algorithms.

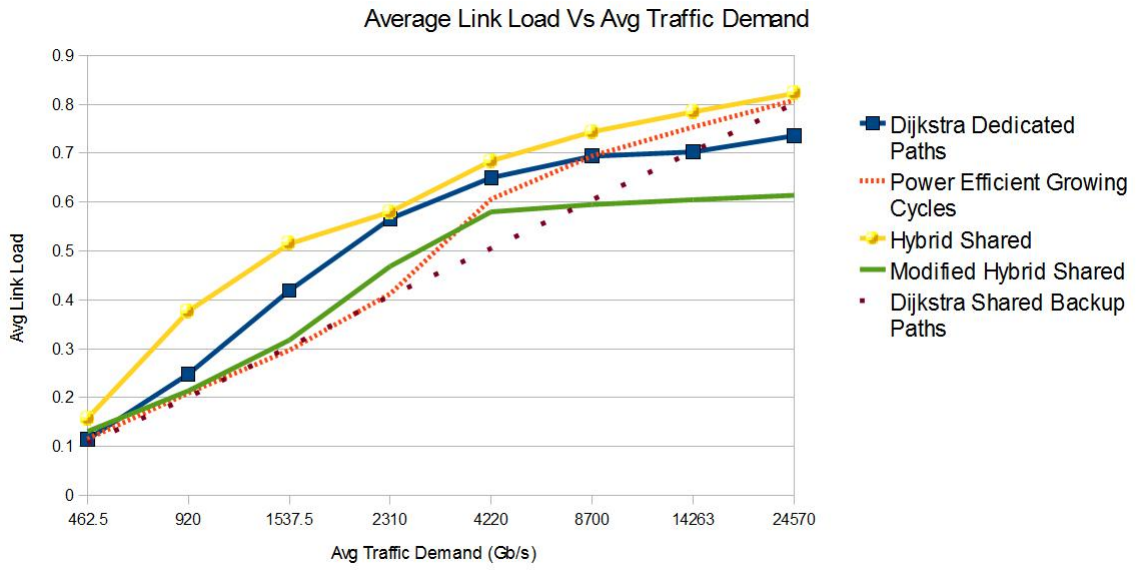

Figure 4.3.1: Average Link Load vs Traffic Demand (Global Crossing Network)

One of the tradeoffs of energy efficiency is a loss of bandwidth efficiency. However, since the energy efficient algorithms allow the sharing of backup bandwidth, they should be more bandwidth efficient than dedicated backup path protection algorithms. At all levels of network load, the Hybrid Shared algorithm utilizes more bandwidth than the Dijkstra Dedicated Paths protection algorithm. This is because of the longer length working

paths that are assigned by the Hybrid Shared algorithm. Bandwidth assigned for working paths cannot be shared among demands so, longer working path lengths will lead to bandwidth inefficiencies. As discussed in Section 3.2.3.2, the Modified Hybrid Shared algorithm fixes the path assignment problems of the Hybrid Shared algorithm and is much more bandwidth efficient.

At lower traffic demand levels, the Power Efficient Growing Cycles algorithm had short working paths and at higher traffic demands it had long working paths while the backup path lengths stayed nearly the same at every traffic demand level (See Figures 4.3.2 and 4.3.3 on the next page). The Modified Hybrid Shared algorithm had working path lengths that were nearly the same at every traffic demand level while the backup paths were longer at low traffic demand levels and shorter at high traffic demand levels (See Figures 4.3.2 and 4.3.3 on the following page). The increase in longer working paths at higher traffic demand levels is why the Power efficient growing cycles algorithm loads the links more than the Dijkstra Dedicated Backup Paths protection algorithm at higher traffic demand levels.

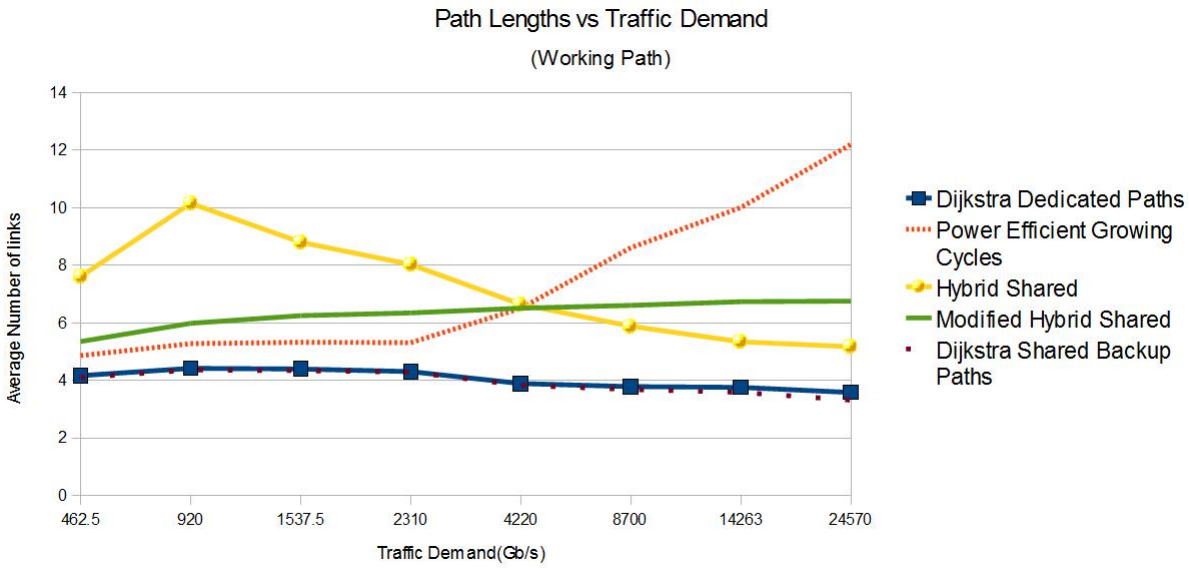

Figure 4.3.2: Working Path Lengths vs Traffic Demand (Global Crossing Network)

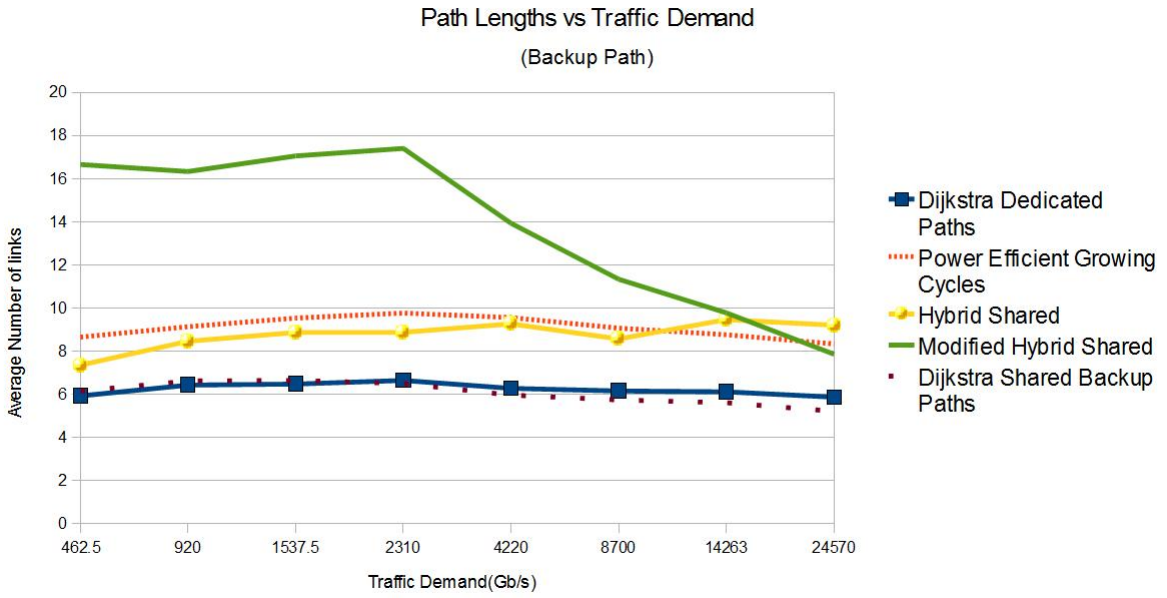

Figure 4.3.3: Backup Path Lengths vs Traffic Demand (Global Crossing Network)

The Demand Rejection Metric vs Traffic Demand is shown below in Figure 4.3.4 on the next page. The Hybrid Shared and Modified Hybrid Shared algorithms operate as expected. As discussed in Section 4.2.1, the energy efficient algorithms should reject more demands than the Dijkstra Shared Backup Paths algorithm and less than the Dijkstra Dedicated Paths algorithm. However, the Power Efficient Growing Cycles algorithm rejected fewer demands than the Dijkstra Shared Backup Paths protection algorithm. This behaviour occurs because the Dijkstra Shared Backup Paths protection algorithm is not designed to deal with the trap topology. In networks with a low degree of connectivity like in the Global Crossing Network, the number of times the trap topology occurs is much higher than in a network with a high degree of connectivity. Algorithms that use P-Cycles to plan both working and backup paths are not affected by the trap topology since there are always two possible paths available between two on cycle nodes. At higher traffic demand levels, the Modified Hybrid Shared algorithm rejects an increasingly larger number of demands than the Dijkstra Shared Backup Paths protection algorithm. This is the reason the Modified Hybrid Shared algorithm loads the network less than the Dijkstra Shared Backup Paths protection algorithm. The Modified Hybrid Shared algorithm is not assigning more demands onto the network, and therefore, it is not loading the network more at higher traffic demand levels.

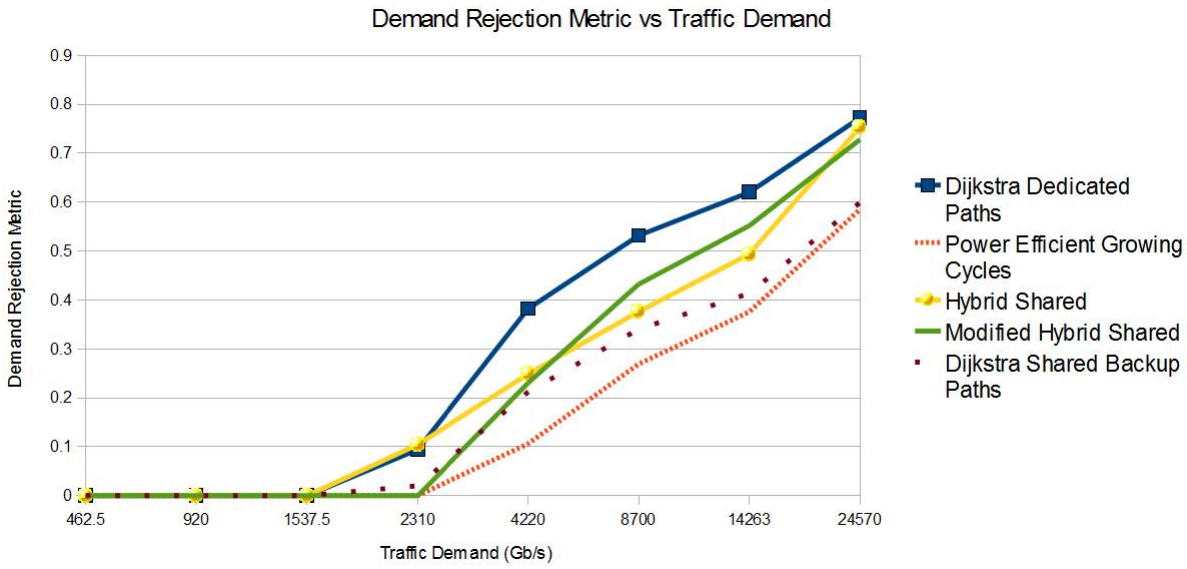

Figure 4.3.4: Demand Rejection Metric vs Traffic Demand (Global Crossing Network)

The number of links in sleep mode is shown in Figure 4.3.5 on the following page. At low traffic demand levels, the Dijkstra Shared Backup Paths and Dijkstra Dedicated Paths algorithms have more links in sleep mode than the energy efficient algorithms. As the traffic demand levels increase, the energy efficient algorithms begin to have more sleep mode links than the Dijkstra Dedicated Paths algorithm. However, with the exception of the Modified Hybrid Shared algorithm, and one level of traffic demand with the Hybrid Shared algorithm, the Dijkstra Shared Backup Paths algorithm has more sleep mode links. This isn't necessarily an indication that the benchmark algorithms are more energy efficient. The number of links in offline mode is shown in Figure 4.3.6 on page 82. Offline links use far less energy than sleep mode links so, even though the benchmark algorithms have more links in sleep mode, the energy efficient algorithms will be more energy efficient since they have more links in offline mode.

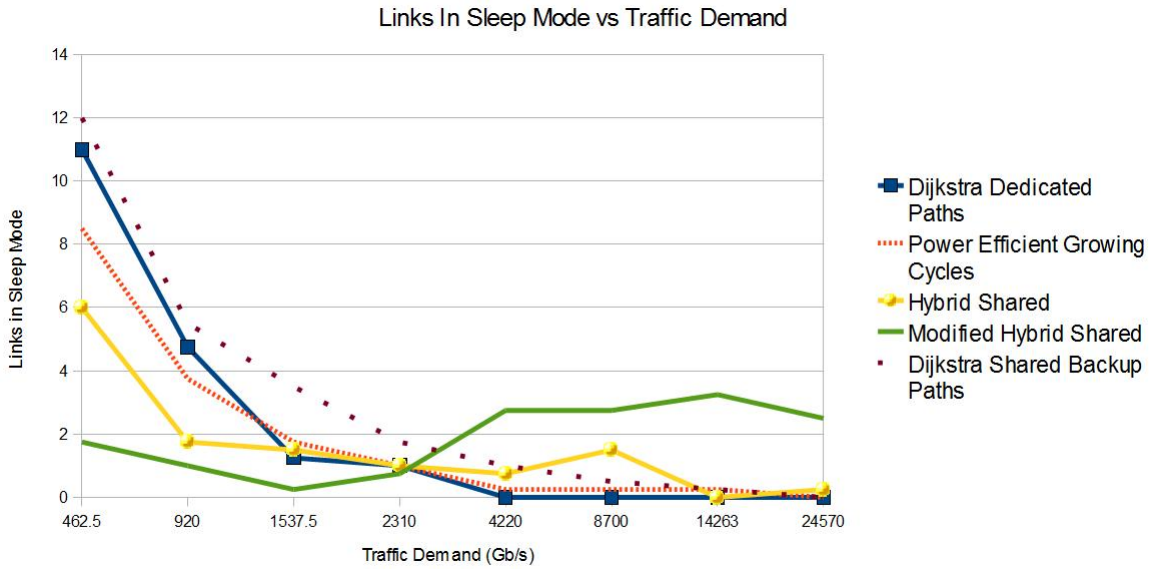

Figure 4.3.5: Links in Sleep Mode vs Traffic Demand (Global Crossing Network)

The energy efficient algorithms have more links offline than the benchmark algorithms. The Hybrid Shared algorithm does not have more links offline for all traffic demand levels and is not as efficient as the other energy efficient algorithms. The Modified Hybrid Shared algorithm is clearly the best as it has by far the most links in offline mode. For the Modified Hybrid Shared algorithm, there is a low number of links in sleep mode but a higher number of links in offline mode at low traffic demands. As the traffic demand increases, more demands enter the network there will be fewer opportunities to put links in offline mode but, since more traffic is in the network, the chances to increase the number of links in sleep mode increases as well. As can be seen in Figure 4.3.5, the number of sleep mode links increases at high traffic demand levels while the number of offline mode links decreases (shown in Figure 4.3.6 on the next page).

In the Power Efficient Growing Cycles algorithm, the number of links in offline and sleep mode lowers as the traffic demand increases. This behaviour is due to the algorithm not selecting cycles prior to routing traffic. Each cycle is chosen and grown based on each individual demand as they enter the network whereas the Hybrid Shared and Modified Hybrid Shared algorithms take all the expected demands into account when selecting cycles. Taking all the expected demands into account before hand will enable the algorithm to select better cycles for the task of routing traffic while maintaining a higher energy efficiency but at the cost of requiring an accurate prediction of network traffic. The prediction of network traffic has to be continuously updated so that, in order

to achieve power efficiency, the energy efficiency can be maintained over time.

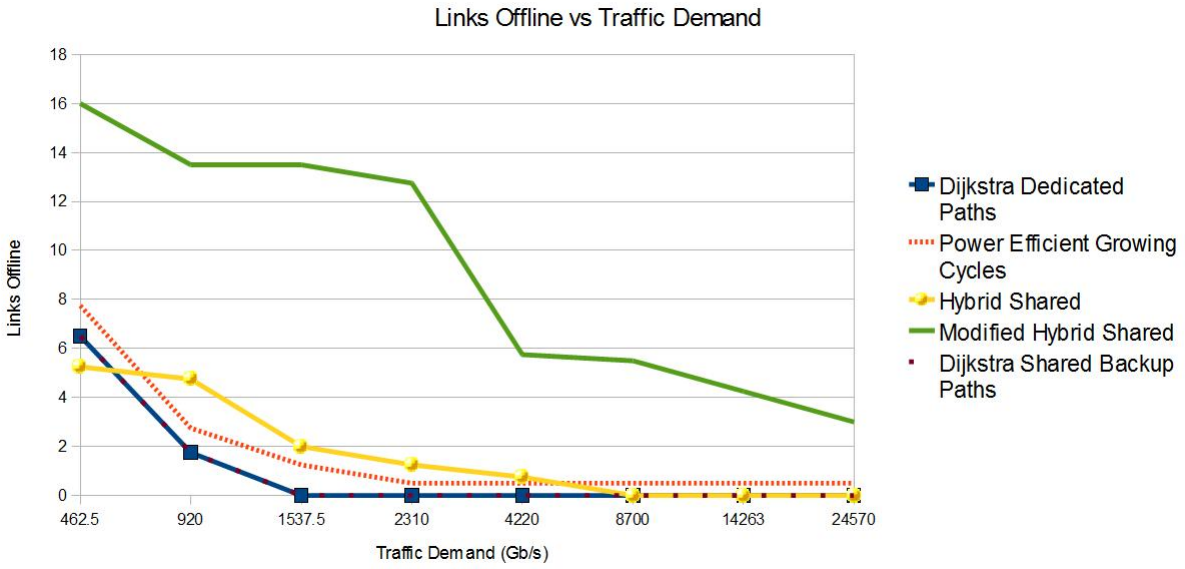

Figure 4.3.6: Links Offline vs Traffic Demand (Global Crossing Network)

The nodes in sleep and offline mode offer the most energy savings. When a link is placed in sleep or offline mode, the resources inside of the nodes required to transmit data over the link are either placed in sleep mode or offline mode. However, when a node is placed into sleep mode or offline mode, all of the resources in that node are placed in sleep mode or offline mode, thus offering much more energy savings over placing links into sleep mode or offline mode. The nodes in sleep mode vs traffic demand for the algorithms is shown in Figure 4.3.7 on the following page and the offline nodes vs traffic demand for the algorithms is shown in Figure 4.3.8 on the next page. With the exception of the Modified Hybrid Shared algorithm, the energy efficient algorithms provided the same number of nodes in sleep mode and nodes in offline mode as the benchmark algorithms at high traffic demand levels while providing less nodes offline and in sleep mode at low traffic demand levels. This behaviour occurs because the benchmark algorithms select the shortest possible paths through the network for the demands. In networks with a low nodal degree, and therefore, a low number of potential paths for demands, the routing scheme that has the lowest number of paths will often lead to the greatest number of nodes that do not have traffic flowing through them or only have backup traffic flowing through them. In the case of the Modified Hybrid Shared algorithm, the paths are longer, but they were selected in order to best service the demands. Multiple demands share the same paths through the network and, the lower the number of unique

paths assigned to demands, the lower the number of nodes that will be needed to route the traffic through the network. However, routing a greater number of demands over the same paths can lead to longer length paths than is necessary for the demands.

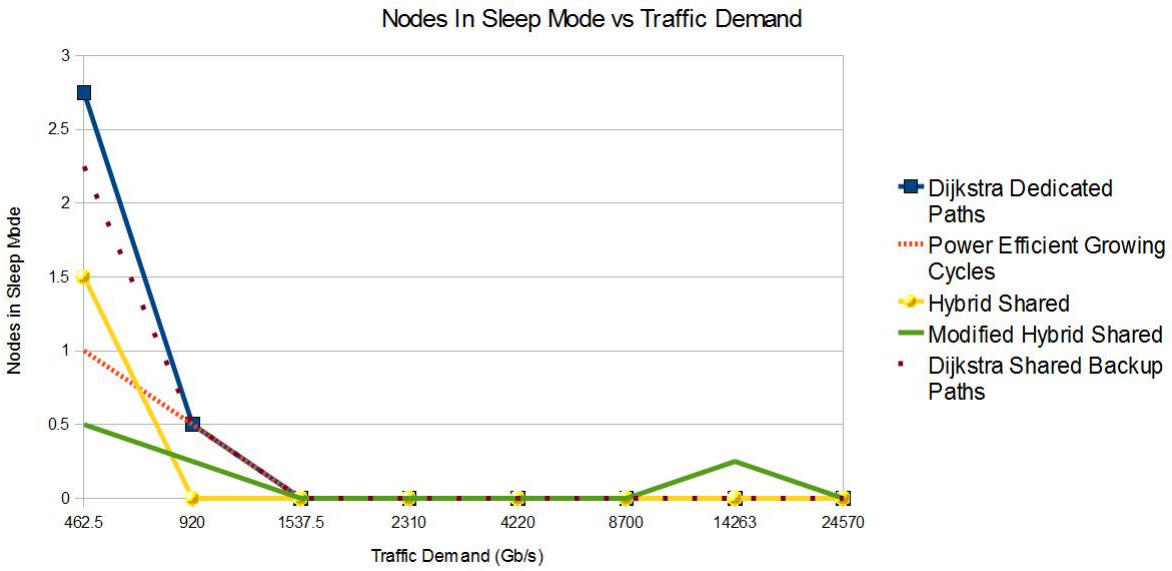

Figure 4.3.7: Nodes in Sleep Mode vs Traffic Demand (Global Crossing Network)

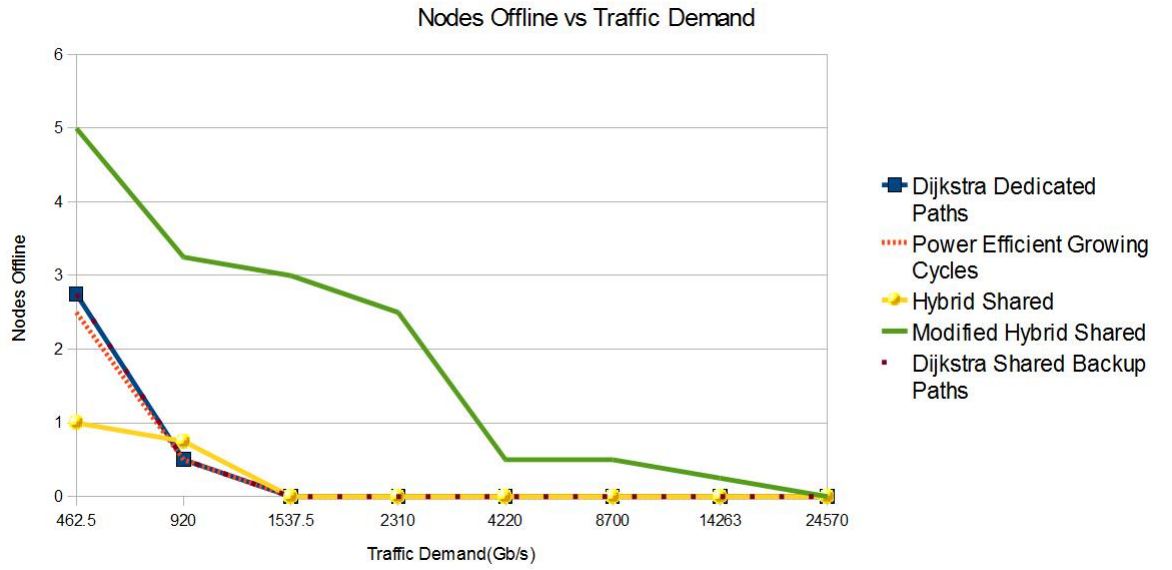

Figure 4.3.8: Nodes Offline vs Traffic Demand (Global Crossing Network)

4.3.2 Kaleidoscope Network

Resource Usage

The results of the Kaleidoscope network simulations for the Power Efficient Growing Cycles, Hybrid Shared, and Modified Hybrid Shared algorithms are shown in this section. The results for the Demand Rejection metric are not presented since none of the demands were rejected for any of the configurations of the network during the simulation. The results for Average Link Load for each configuration of the network are shown in Figure 4.3.9. The Modified Hybrid Shared and Power Efficient Growing Cycles algorithms have similar behaviour when loading the network and the Hybrid Shared algorithm consistently loads the network more than the other two algorithms.

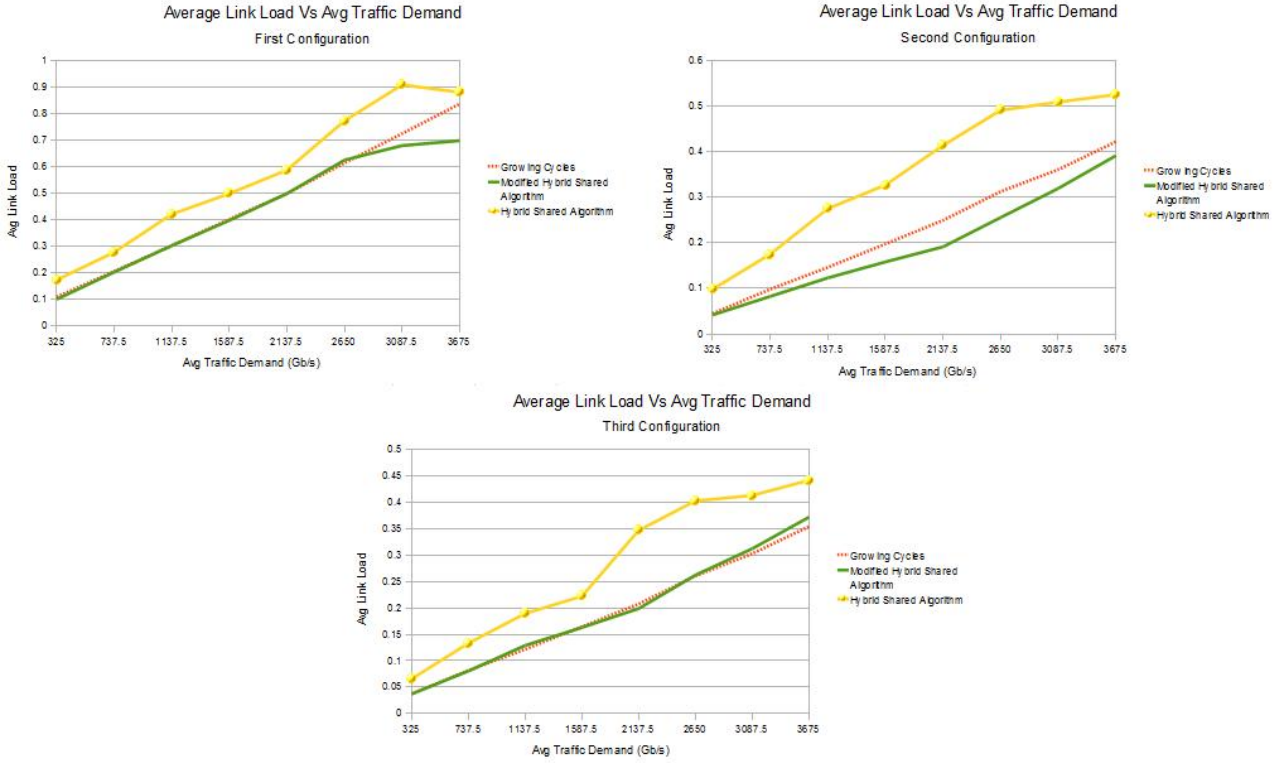

Figure 4.3.9: Average Link Load vs Traffic Demand (Kaleidoscope Network)

However, the working path lengths assigned by the Hybrid Shared algorithm are not significantly longer than the Power Efficient Growing Cycles and Modified Hybrid Shared algorithms (see Figure 4.3.10 on the following page). In the case of the first configuration of the network, the average working path lengths assigned by the Hybrid Shared algorithm are shorter than the other algorithms for higher network loads but the average link

load is still higher. The average backup path lengths assigned by the Hybrid Shared algorithm are shorter than the Power Efficient Growing Cycles and Modified Hybrid Shared algorithms (see Figure 4.3.11 on the next page). Having shorter backup paths for demands leads to fewer opportunities for backup bandwidth sharing and, since working paths require dedicated bandwidth, having working paths that are too long leads to wasteful usage of bandwidth. The working paths assigned by the Modified Hybrid Shared algorithm take the shortest length path. The longer backup paths assigned by the Modified Hybrid Shared algorithm provides more opportunities to share backup bandwidth between demands, and therefore, the average link load for the Modified Hybrid Shared algorithm is lower than the Hybrid Shared algorithm.

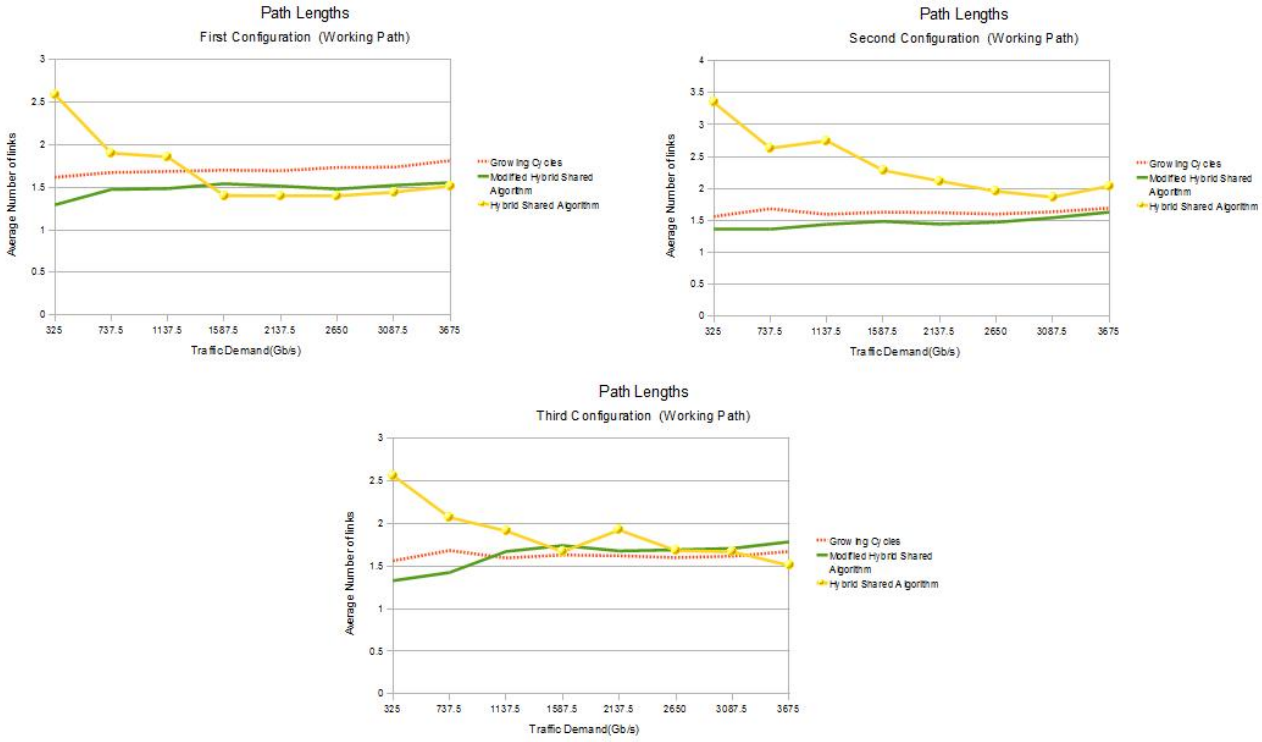

Figure 4.3.10: Average Working Path Length vs Traffic Demand (Kaleidoscope Network)

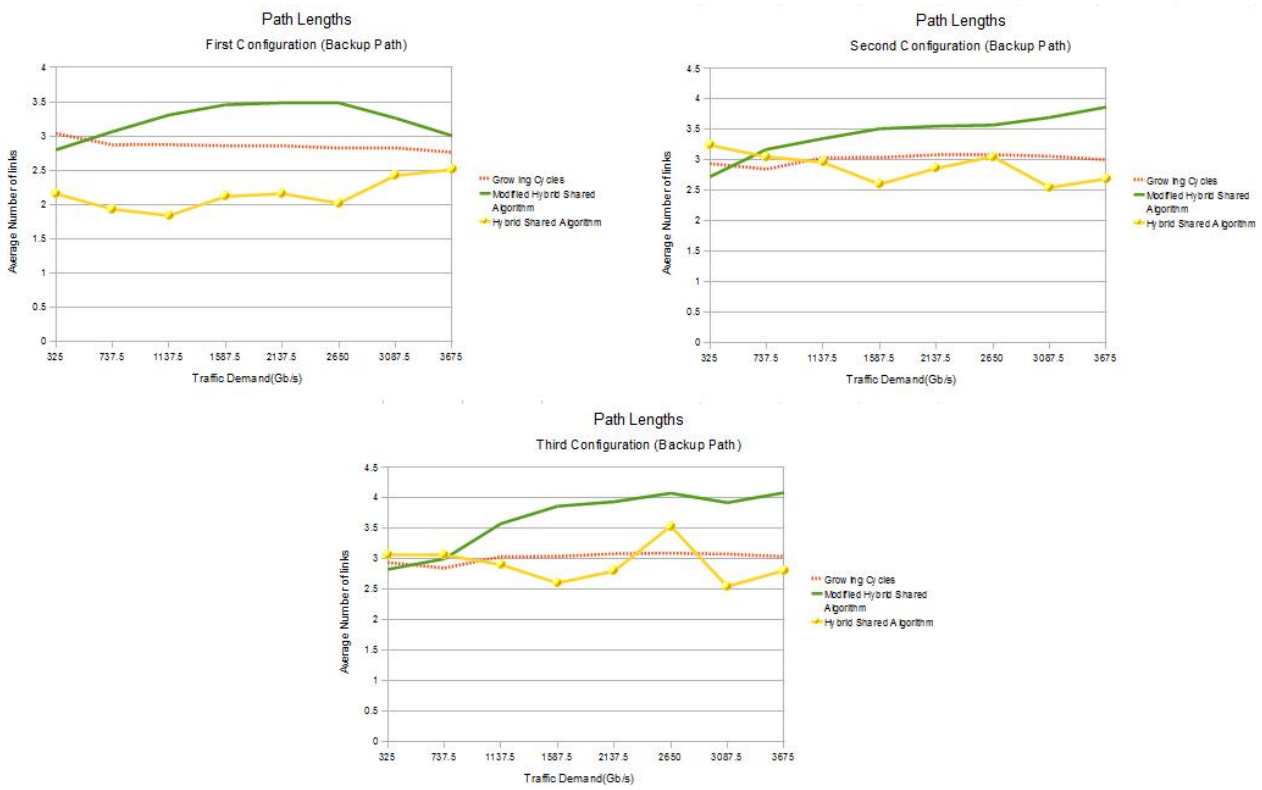

Figure 4.3.11: Average Backup Path Length vs Traffic Demand (Kaleidoscope Network)

The number of links in sleep mode and the links in offline mode for each algorithm on each network configuration are shown in Figure 4.3.12 and Figure 4.3.13 on page 88 respectively. For the first and third configurations, the Hybrid Shared algorithm has more links in sleep mode than the Modified Hybrid Shared and Power Efficient Growing Cycles algorithms. In the second configuration, the Hybrid Shared algorithm has, on average, fewer links in sleep mode than the Power Efficient Growing Cycles algorithm. The Hybrid Shared algorithm has more links in sleep mode than the Modified Hybrid Shared algorithm in each of the network configurations but Modified Hybrid Shared algorithm has more links in offline mode.

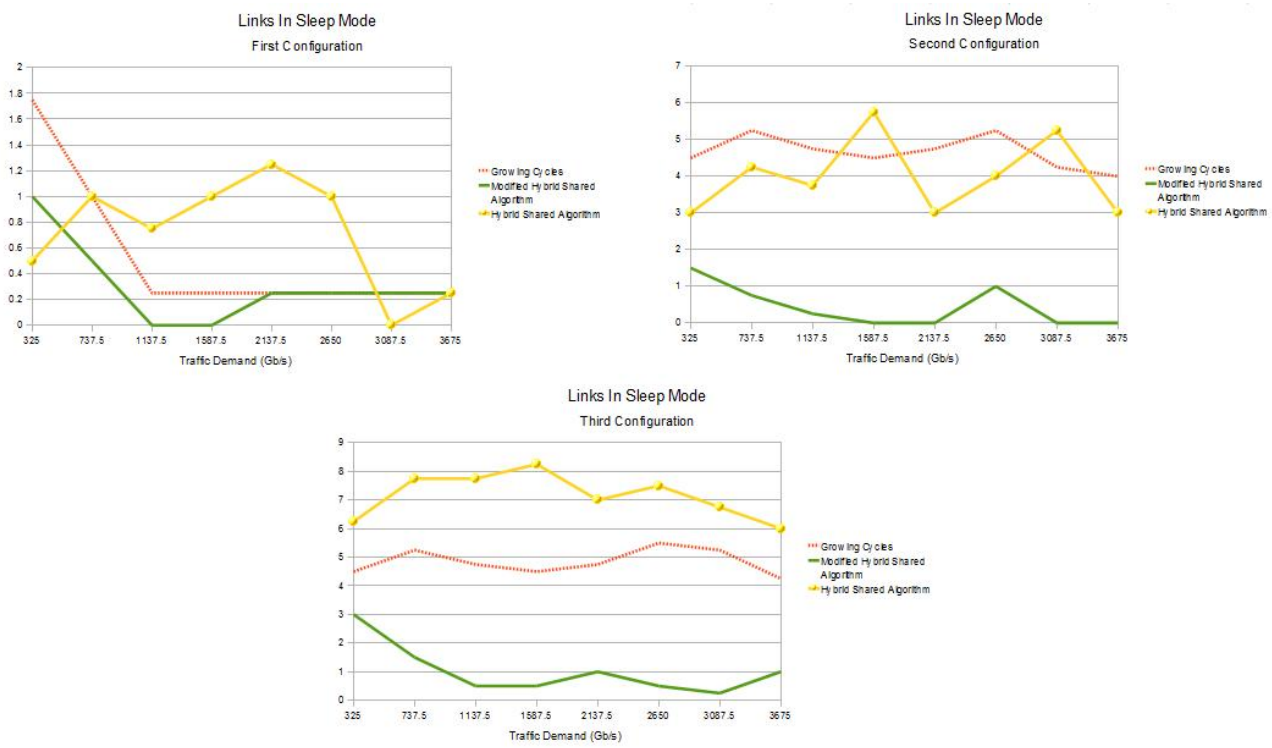

Figure 4.3.12: Links in Sleep Mode vs Traffic Demand (Kaleidoscope Network)

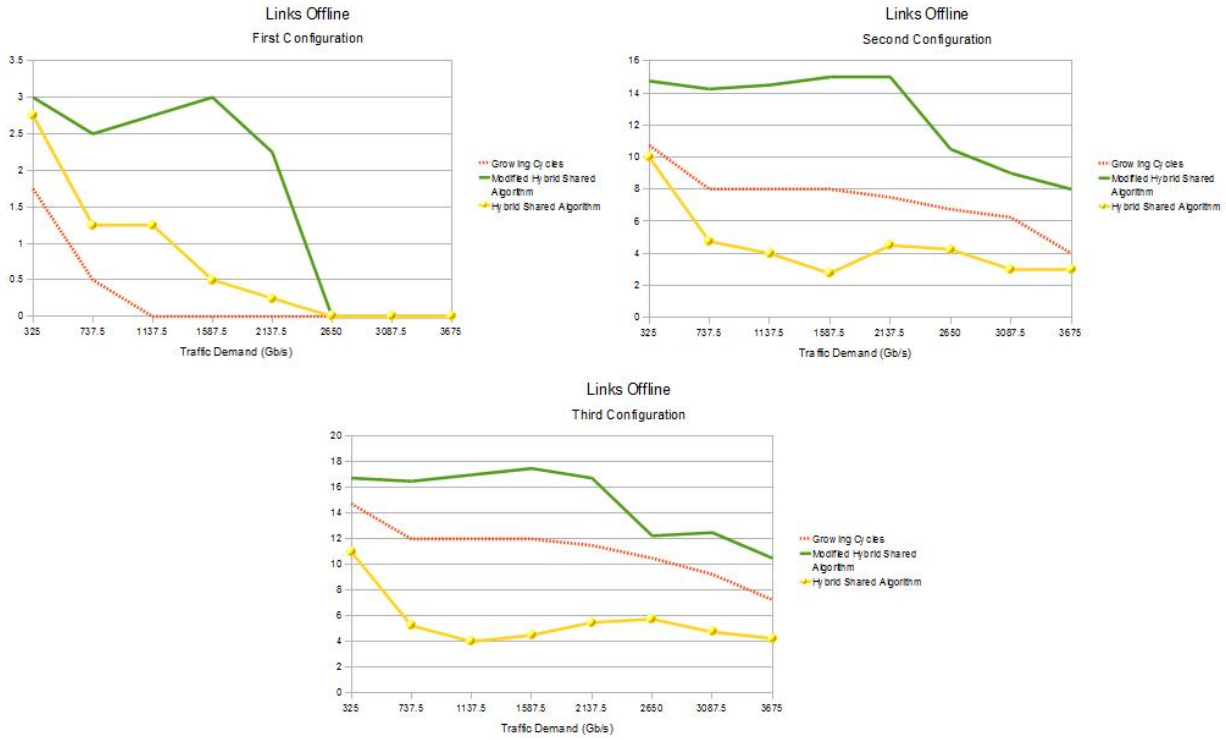

Figure 4.3.13: Links Offline vs Traffic Demand (Kaleidoscope Network)

The number of nodes in sleep mode and offline mode for each algorithm on each network configuration are shown in Figure 4.3.14 on the next page and Figure 4.3.15 on the following page respectively. The Modified Hybrid Shared algorithm has the most nodes offline while the Hybrid Shared algorithm has the least nodes offline. The Power Efficient Growing Cycles algorithm has, on average, more nodes in sleep mode than the Hybrid Shared algorithm in the second configuration but the Hybrid Shared algorithm has more nodes in sleep mode for the third configuration.

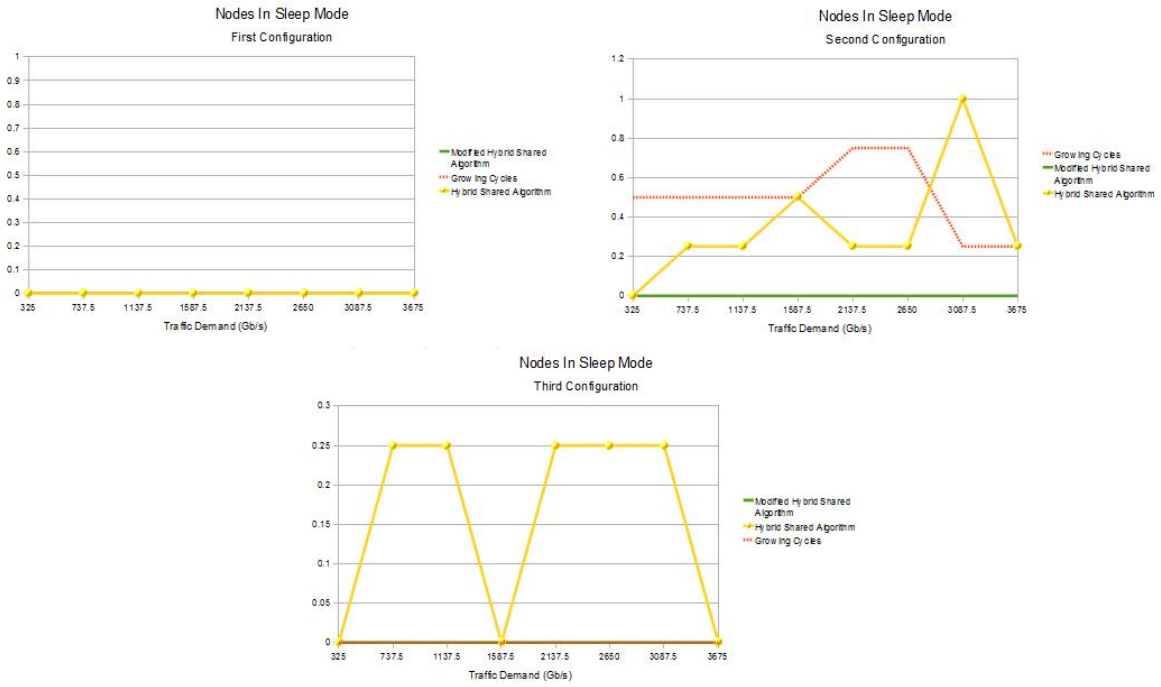

Figure 4.3.14: Nodes in Sleep Mode vs Traffic Demand (Kaleidoscope Network)

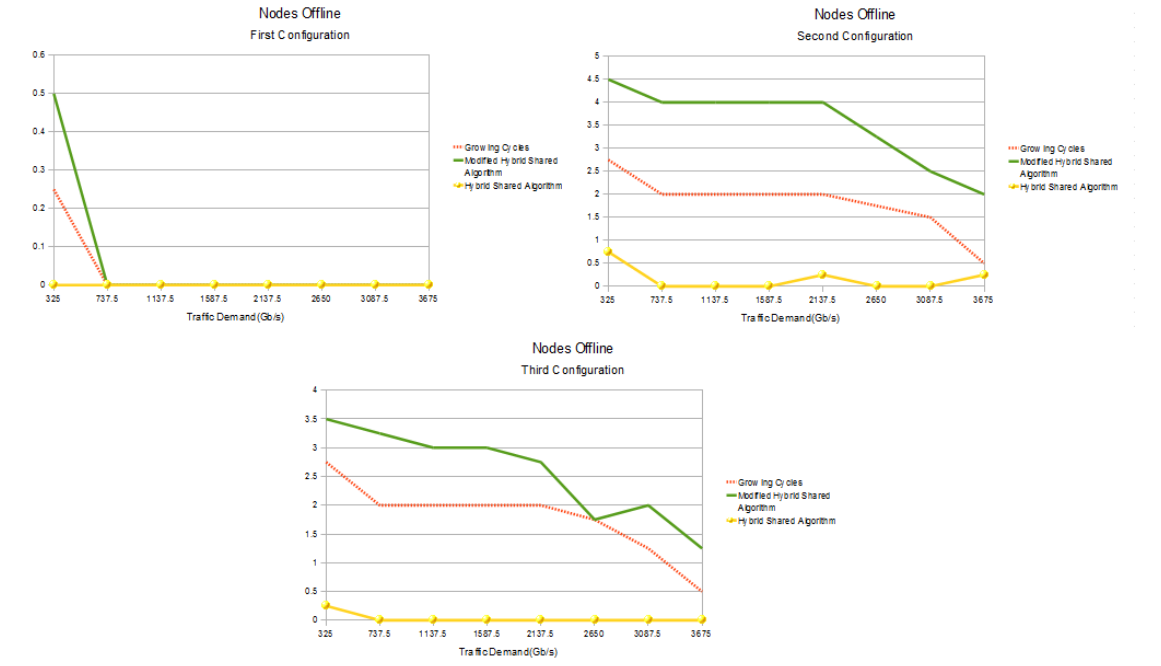

Figure 4.3.15: Nodes Offline vs Traffic Demand (Kaleidoscope Network)

The Modified Hybrid Shared algorithm has the most links and nodes offline, and the least links and nodes in sleep mode for each configuration while the number of links and nodes in offline mode for the Modified Hybrid Shared algorithm on each configuration is far higher than the number of links and nodes in sleep mode for the Hybrid Shared and Power Efficient Growing Cycles algorithms. Since offline links and nodes are more energy efficient than links and nodes in sleep mode, the Modified Hybrid Shared algorithm is more energy efficient than the Hybrid Shared and Power Efficient Growing Cycles algorithms. The Hybrid Shared algorithm is the least energy efficient of the algorithms since it has the least links and nodes in offline mode and the number of links and nodes in sleep mode is much smaller than the number of links and nodes in offline mode for the Modified Hybrid Shared and Power Efficient Growing Cycles algorithms.

4.3.3 Random Layout Network

The Results for the Random Layout Network are presented in this section. The performance of the Modified Hybrid Shared algorithm and the Power Efficient Growing Cycles algorithm is compared to the two benchmark algorithms on a large network with a high degree of connectivity. Since the Hybrid Shared algorithm has thus far shown far less energy performance than the Modified Hybrid Shared and Power Efficient Growing Cycles algorithms, and the Modified Hybrid Shared algorithm was designed to address the bandwidth issues of the Hybrid Shared algorithm (see Section 3.2.3.2), the Hybrid Shared algorithm was not tested on the Random Layout Network.

The average link load for each of the algorithms is given below in Figure 4.3.16. The Power Efficient Growing Cycles algorithm loads the network as expected (see Section 4.2.1). It loads the links more than the Dijkstra Shared Backup Paths algorithm and less than the Dijkstra Dedicated Paths algorithm. The Modified Hybrid Shared algorithm performs as expected for low traffic demand levels but loads the links less than all of the other algorithms at higher link loads.

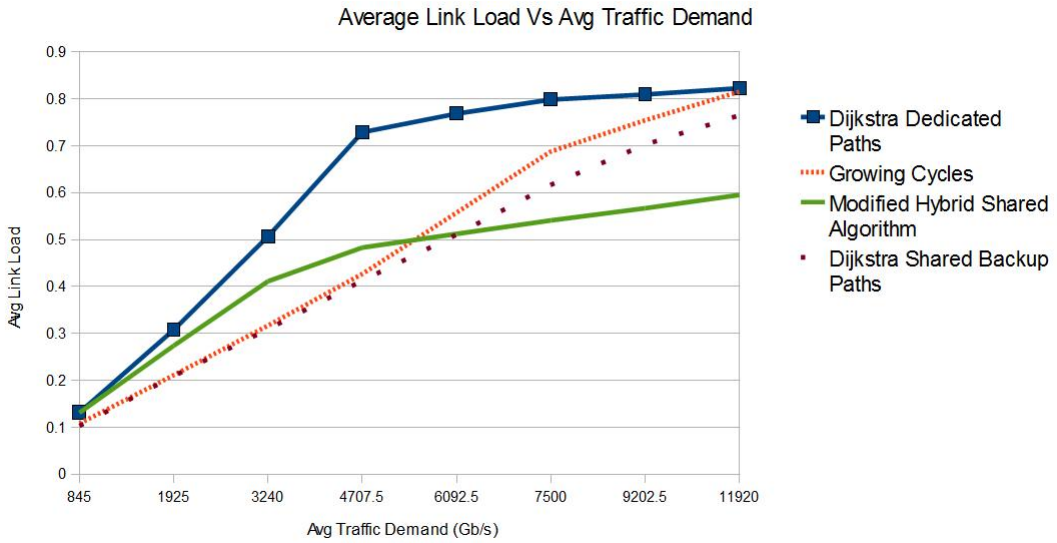

Figure 4.3.16: Link Load vs Traffic Demand (Random Layout Network)

The average working path lengths and backup path lengths are shown in Figures 4.3.17 and 4.3.18 on the following page respectively. The lengths for both the working and backup paths are much higher for the Modified

Hybrid Shared algorithm then the other algorithms. The lengths of the working and backup paths for the Power Efficient Growing Cycles algorithm are higher than the benchmark algorithms and the difference in the working path lengths between the Power Efficient Growing Cycles algorithm and the benchmark algorithms increases at higher link loads. This difference in path lengths is the cause for the increase in the difference between link loads of the Power Efficient Growing Cycles algorithm and the Dijkstra Shared Backup Paths algorithm.

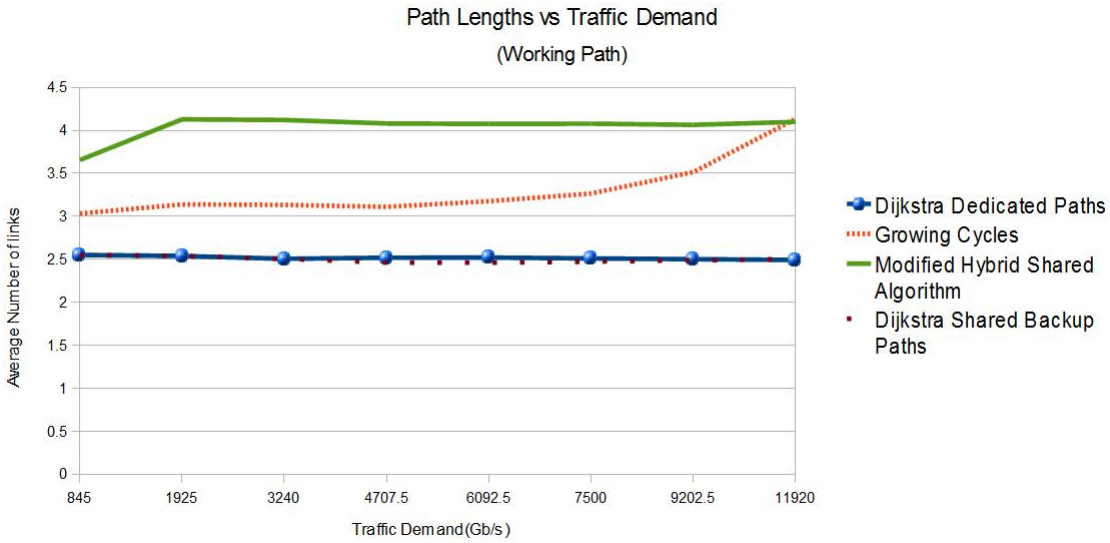

Figure 4.3.17: Working Path Lengths vs Traffic Demand (Random Layout Network)

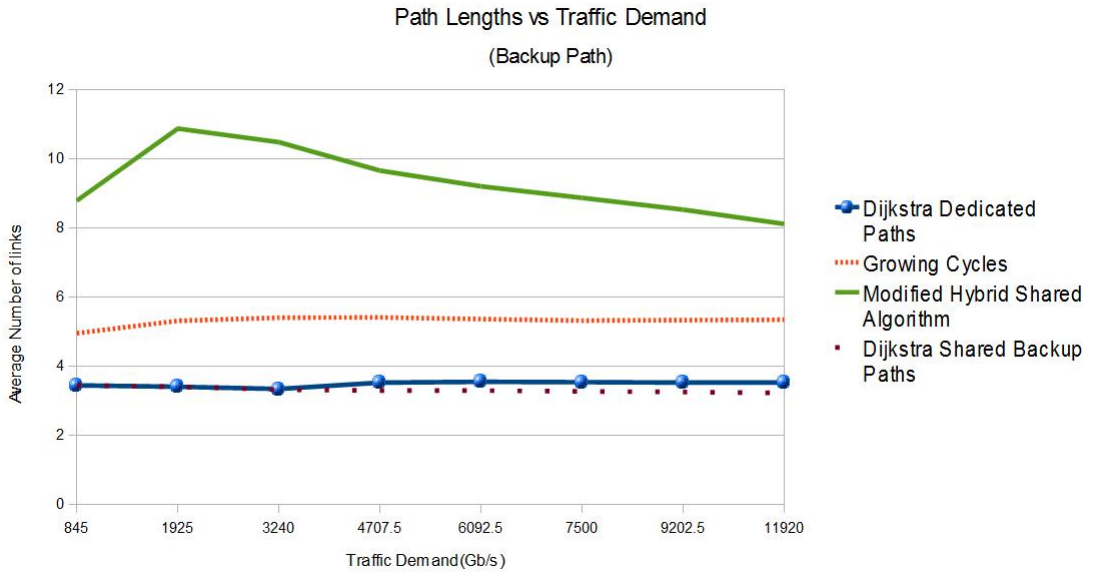

Figure 4.3.18: Backup Path Lengths vs Traffic Demand (Random Layout Network)

As shown in Figure 4.3.19, the Modified Hybrid Shared algorithm rejects far more demands than the other algorithms. The higher number of rejected demands is the cause of the lower link loads at higher traffic demand levels since the algorithm is not accepting any more demands onto the network.

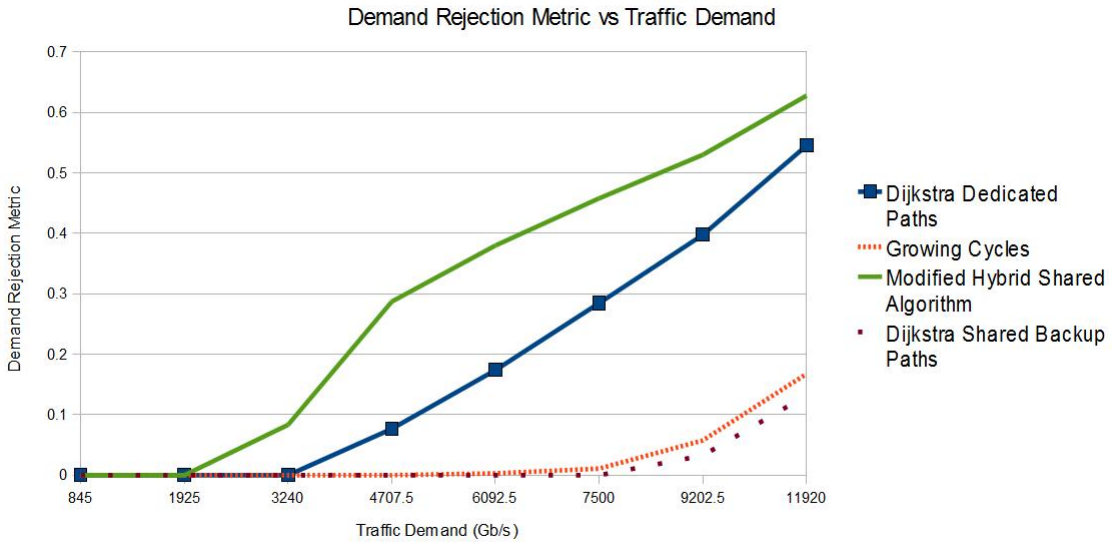

Figure 4.3.19: Demand Rejection Metric vs Traffic Demand (Random Layout Network)

The average number of links in sleep mode and offline mode are shown in Figures 4.3.20 and 4.3.21 on the following page respectively. The Dijkstra Shared Backup Paths and the Dijkstra Dedicated Paths algorithms have more links in sleep mode than the Power Efficient Growing Cycles algorithm. However, this does not mean they are more energy efficient. The difference in the average number of links in sleep mode is, at maximum, approximately 3.2 for the Dijkstra Shared Backup Paths algorithm and approximately 1.5 for the Dijkstra Dedicated Paths algorithm. On the other hand, the Power Efficient Growing Cycles algorithm has more links in offline mode than the Dijkstra Dedicated Paths and Dijkstra Shared Backup Paths algorithms. The difference in the average number of links in offline mode is, at minimum, approximately one node and, at maximum, approximately three nodes. Since links in offline mode use less energy than links in sleep mode, and the difference in the number of links offline and links in sleep mode between the Power Efficient Growing Cycles algorithm and the benchmark algorithms is small, the Power Efficient Growing Cycles algorithm is more energy efficient than the benchmark algorithms when considering only links. The Modified Hybrid Shared Algorithm has, on average, fewer links in sleep mode than the Dijkstra Dedicated Paths algorithm and only has more links in sleep mode than the Dijkstra Shared Backup Paths algorithm at high traffic demand levels. The number of links offline for

the Modified Hybrid Shared algorithm is far higher than the number of links offline for the other algorithms at low traffic demand levels. At higher traffic demand levels, the number of links offline is still higher but the difference is much smaller than at low traffic demand levels. When considering only links, the Modified Hybrid Shared algorithm is the most energy efficient of the algorithms.

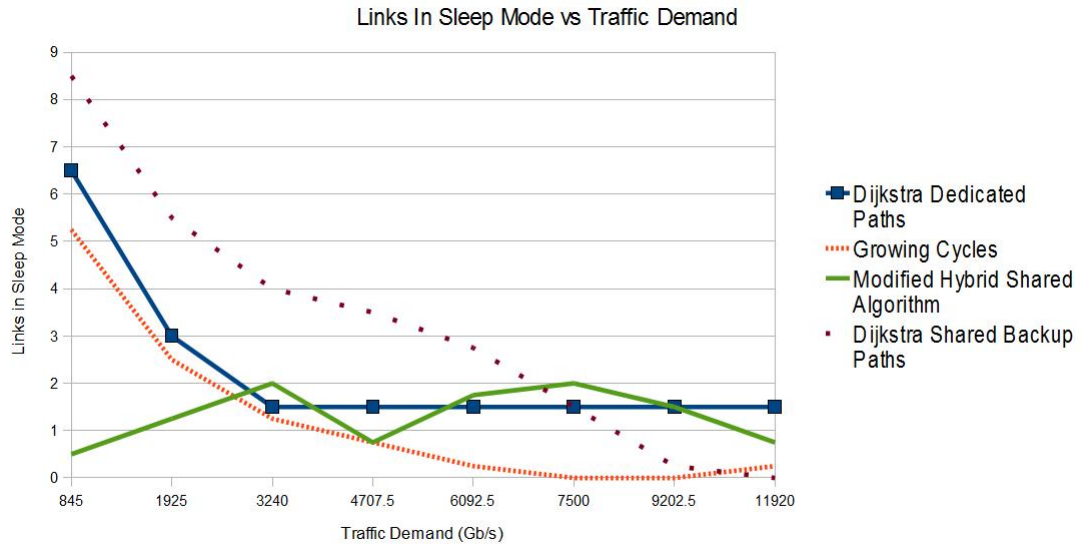

Figure 4.3.20: Links in Sleep Mode vs Traffic Demand (Random Layout Network)

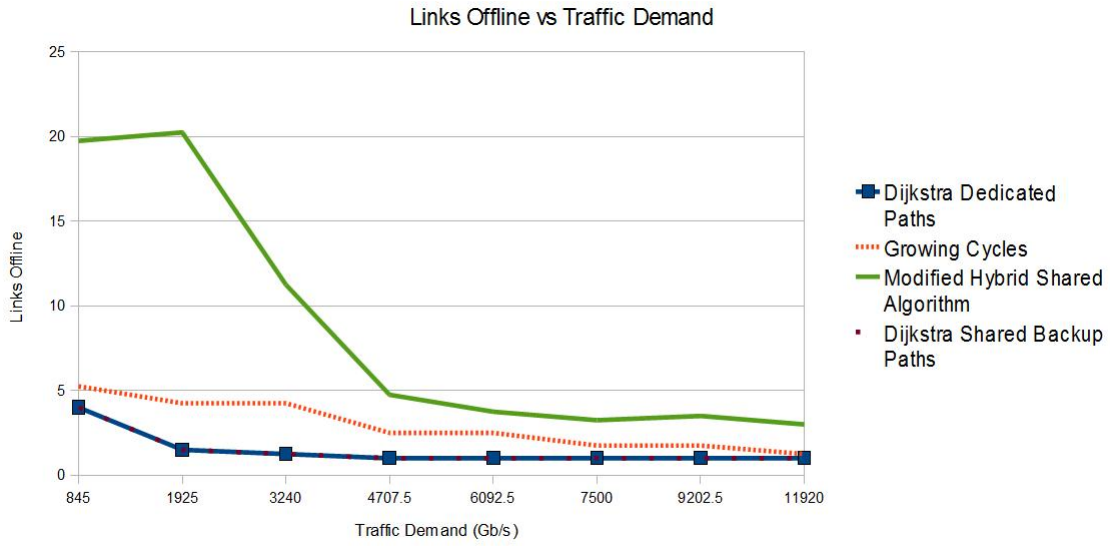

Figure 4.3.21: Links Offline vs Traffic Demand (Random Layout Network)

The number of nodes in sleep mode and offline mode are shown in Figure 4.3.22 and Figure 4.3.23 on the next page respectively. Except for one traffic demand level, the Dijkstra Shared Backup Paths algorithm has more nodes in sleep mode. At low traffic demand levels, the Modified Hybrid Shared algorithm has more nodes in offline mode than the other algorithms. The high number of nodes in sleep mode for the Modified Hybrid Shared algorithm that happens at 3240 Gb/s traffic demand is due to how the Modified Hybrid Shared algorithm routs traffic. Since the cycles used in determining the paths for the demands depends on the traffic being assigned paths and bandwidth in the network, the paths the demands are assigned will change with the traffic. Therefore, it is possible for the number of nodes in sleep mode to be very different from one traffic demand level to another. The backup paths are very long in at 3240Gb/s and get shorter at higher traffic demand levels. This means that very large cycles are being used (approximately 14.2 links long when adding working and backup path lengths). Cycles that large are going to include a lot of the links and nodes from the edge of the network graph. Therefore, most of the traffic is being routed over the edge of the network graph and not much is being routed through the inside of the network. Many of the nodes on the outside of the network graph have three links connected to them and one only has two. Many of the backup paths are routed through the nodes on the outside of the network graph but do not pass back into the inside of the network and only use two of the three links available to them. This leaves the third link unused. Thus the nodes wind up in sleep mode. As the traffic demand level increases, the path lengths decrease and smaller cycles start being used that overlay the larger cycles. This causes more traffic to be routed through the inside links and nodes of the network and so the number of nodes in sleep mode decreases back to zero. The Power Efficient Growing Cycles algorithm has more nodes in offline mode than the benchmark algorithms at low traffic demand levels. At high traffic demand levels, the number of nodes in offline mode falls to the same level as the benchmark algorithms. Offline nodes utilize far less energy than sleep mode nodes so, even though the Dijkstra Shared Backup Paths protection algorithm has more nodes in sleep mode than the number of nodes in offline mode for the Modified Hybrid Shared algorithm; the Modified Hybrid Shared algorithm is more energy efficient than the Dijkstra Shared Backup paths algorithm when only considering nodes.

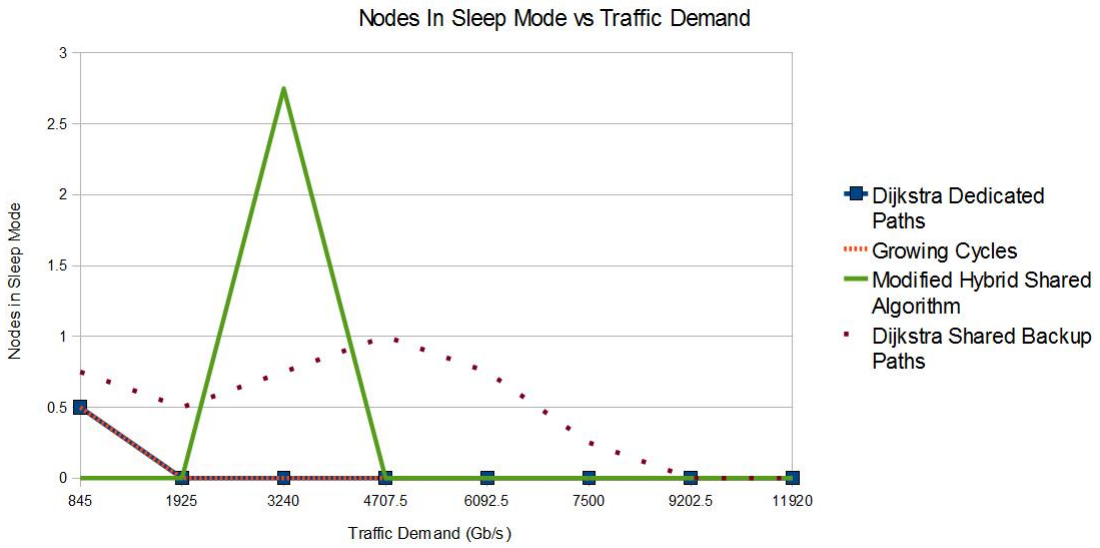

Figure 4.3.22: Nodes in Sleep Mode vs Traffic Demand (Random Layout Network)

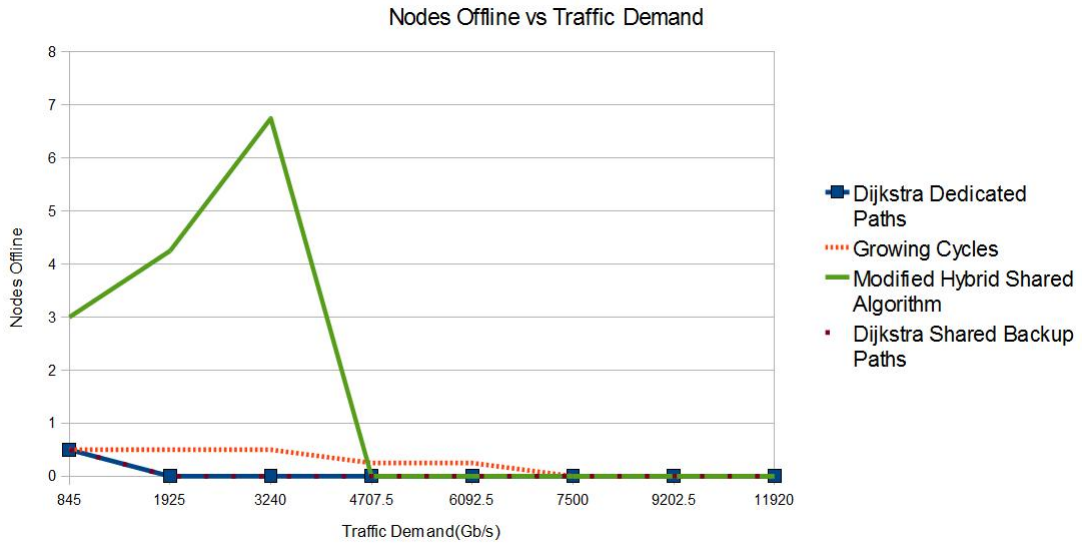

Figure 4.3.23: Nodes Offline vs Traffic Demand (Random Layout Network)

In the Random Layout Network, the Modified Hybrid Shared algorithm was the most energy efficient while the Power Efficient Growing Cycles algorithm was less energy efficient than the Modified Hybrid Shared algorithm but still more energy efficient than the benchmark algorithms. Even though the Modified Hybrid Shared algorithm also had the longest path lengths and rejected the most demands. It was the most bandwidth inefficient. The Power Efficient Growing Cycles algorithm had longer paths than the benchmark algorithms and the amount of

rejected demands was close to the Dijkstra Shared Backup Paths algorithm. However, the energy efficiency of the Power Efficient Growing Cycles algorithm was close to, but still higher than, the Dijkstra Shared Backup Paths algorithm. Thus, the tradeoff between energy efficiency and bandwidth efficiency is apparent. The higher the energy efficiency the lower the bandwidth efficiency and vice-versa.

Chapter 5

Future Work

This thesis focused on the Path Finding Stage of the power efficient routing scheme. The proposed algorithms were used to find an energy efficient routing solution for a given set of demands and their performance was compared with two common benchmark algorithms. The feasibility of the power efficient routing scheme is proven since p-cycles can be used to find the energy efficient set of paths for each of the different levels of network load. However, due to the focus on the Path Finding Stage, this thesis did not have the opportunity to explore the other stages of the power efficient routing scheme. These stages are:

- Prediction Stage
- Operation Stage

A study of the four algorithms for prediction of self similar processes, discussed in Section 2.2, has to be performed in order to determine how well they predict the behaviour of real network traffic, how difficult it is to update the prediction algorithm to meet the changes to the traffic pattern that occur as the network grows in size and number of customers, the speed the algorithms can predict a set of traffic, and the ideal size of the time period the algorithm will predict traffic for. It is important for a prediction algorithm to be accurate but also execute quickly. However, shorter execution times come with a loss in prediction accuracy so it is important to ensure that an algorithm will execute quickly but not have an unacceptable amount of error. With time, a network prediction algorithm will begin to have errors that are introduced by changes in the network itself. As the network grows and more customers are added, the behaviour of the traffic will also change, and it is

necessary to choose a prediction algorithm that will be easy to update so that it will continue to accurately predict the behaviour of traffic as it changes. The ideal size of the time period the algorithm predicts traffic for is also important. The predictions will become more inaccurate the farther into the future the algorithm predicts. However, since the length of time the Operation Stage runs for is the same as the size of the time period the traffic is predicted for, it is also important to make sure that the time period is sufficiently large to allow for the next set of predictions to be made and the set of paths for the demands of the next set of predictions to be found.

Another important area for further study is in how to handle resident demands when the paths are found for a set of predicted demands. An important issue when switching routing tables is how to handle the resident traffic. Traffic that is still in the network when the table is switched is known as resident traffic. There are three ways to handle the resident traffic:

- Ignore Resident Traffic
- Include Resident Traffic
- Adjust Resident Traffic Paths

When ignoring resident traffic, the path finding stage runs without considering the resident traffic (i.e. it acts as if it isn't there). Changing the routing table in this stage will have no effect on the service of customers since the resident traffic is left alone. If the resident traffic has a low holding time, it will leave the network quickly enough that it wouldn't raise any significant power efficiency issues. However, if the resident traffic has a long holding time, it can cause the routing algorithm to find a less energy efficient solution than it would if the resident traffic were considered when routing. If this occurs every time the routing table is generated during the path finding stage, the power consumption will also increase. Consider the network shown in Figure 5.0.1. Two cycles used to protect the network are shown in Figure 5.0.1 (A), and the demands that will be assigned to the network, including a resident demand, are shown in Figure 5.0.1 (B). Demands 1 and 3 are assigned to Cycle 1 while Demand 2 is assigned to Cycle 2. The working paths for these demands is shown in Figure 5.0.1 (C). The resident demand is left alone and not assigned to any cycles. The backup paths are routed around the cycles as shown in Figure 5.0.1 (D). There is one link with no traffic on it that can be put into offline mode and there are four links that have only backup traffic routed over them that can be put into sleep mode. Link 6-2 is in online mode (it has working paths over it) and link 5-6 is in sleep mode (only backup paths are assigned

to it). However, if the resident demand was not in the network, both links would have been in offline mode. In fact, if the resident traffic is considered during the path finding stage as it is in the cases discussed below, the energy efficiency can be improved.

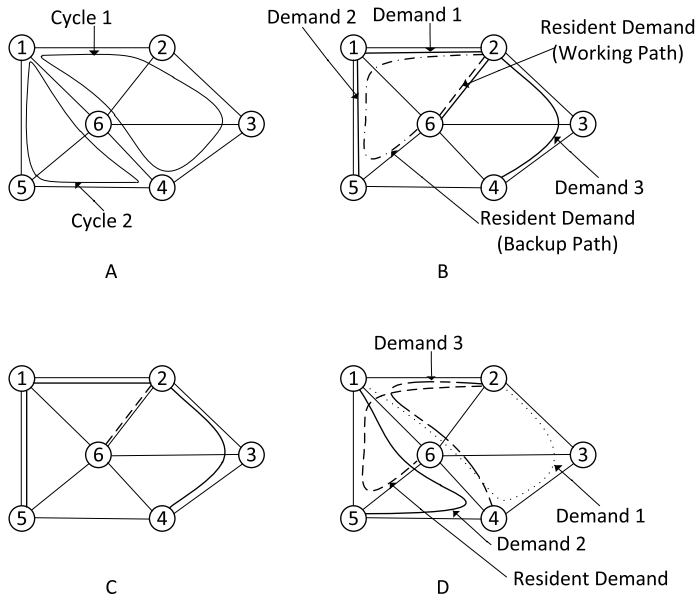

Figure 5.0.1: Ignoring Resident Traffic: (A) Two cycles protecting an example network. (B) A resident demand and three new demands. (C) Working paths for the demands. (D) Backup paths for the demands. Note that the working and backup paths for the resident demand are not assigned by Cycle 1 or Cycle 2.

Including resident traffic will fix the energy issue if the resident traffic has a long holding time and will be in the network long after the routing table changes. In this case, the resident traffic is added to the projected traffic when altering the routing table, but it is considered fixed. This means that the paths for the resident traffic cannot be changed. The cycles used by resident traffic are known as resident cycles. The resident cycles would then be given a higher weight when assigning scores to the cycles so that they will most likely be considered first over the non resident cycles. In this way, the new traffic will be grouped with the resident traffic and there will not be any issues associated with switching resident traffic onto a new path. Consider again the example and network discussed above in Figure 5.0.1. The network, reproduced in Figure 5.0.2 (A), is protected by Cycles 1 and 2. The resident cycle is the cycle that was assigned to the resident demand in a previously run path finding stage (Figure 5.0.2 (B)). Also shown are three demands that need to be assigned paths in the current path finding stage. The paths for the resident traffic do not change but, unlike in the case where resident traffic is ignored, the demands can now be assigned to the resident cycle as well as Cycles 1 and 2. The working paths for the

demands are shown in Figure 5.0.2 (C) and the backup paths for the demands are shown in Figure 5.0.2 (D). Demands 1 and 2 are assigned to the resident cycle and Demand 3 is assigned to Cycle 1. There are two links with no traffic on them that can be put into offline mode and there are three links that have only backup traffic routed over them that can be put into sleep mode. This is an improvement over the example case discussed above, where resident traffic is ignored.

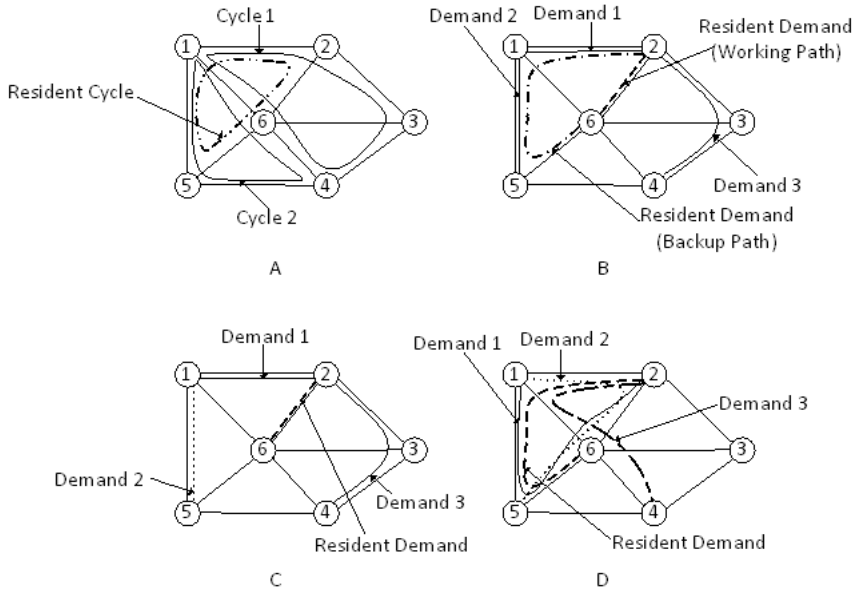

Figure 5.0.2: Including Resident Traffic: (A) Two cycles protecting an example network. One cycle that was used for the Resident Demand is also shown. (B) A resident demand and three new demands. (C) Working paths for the demands. (D) Backup paths for the demands. Note that the working and backup paths for non resident demands can be assigned to the resident cycle.

The resident traffic paths could also be adjusted. This means that the resident traffic is added to the projected traffic when altering the routing table and the traffic is not considered fixed. This means that the paths for the resident traffic can be changed and it is considered the same as non resident traffic. Consider again the example and network discussed above in Figure 5.0.1. The network, reproduced in Figure 5.0.3 (A), is protected by two cycles. The demands are also reproduced in Figure 5.0.3 (B). Demands 1 and 3 are assigned to Cycle 1 and Demand 2 is assigned to Cycle 2. Unlike in the cases discussed above, the paths for the resident traffic are treated like the regular demands (Demands 1, 2, and 3), and are assigned to one of the cycles protecting the network (Cycle 1 in this example). The working paths are shown in Figure 5.0.3 (C) and the backup paths are shown in Figure 5.0.3 (D). There are three links with no traffic on them that can be put into offline mode and there are

two links that have only backup traffic routed over them that can be put into sleep mode. Adjusting the paths of resident traffic can lead to the most energy efficient solution. However, changing the paths of resident traffic raises the issue of how to switch the paths without interrupting service to the users who are the source of that traffic. A method to handle this is to purposely interrupt the working path. Then, while the traffic is restored on its backup path, change the working path to the new one. Once the new working path is completed, the backup path is restored onto the new working path and the backup path is updated. This method comes at the cost of a short interruption to the resident traffic while the paths are updated. However, since p-cycles have a very fast restoration time, this interruption is extremely small.

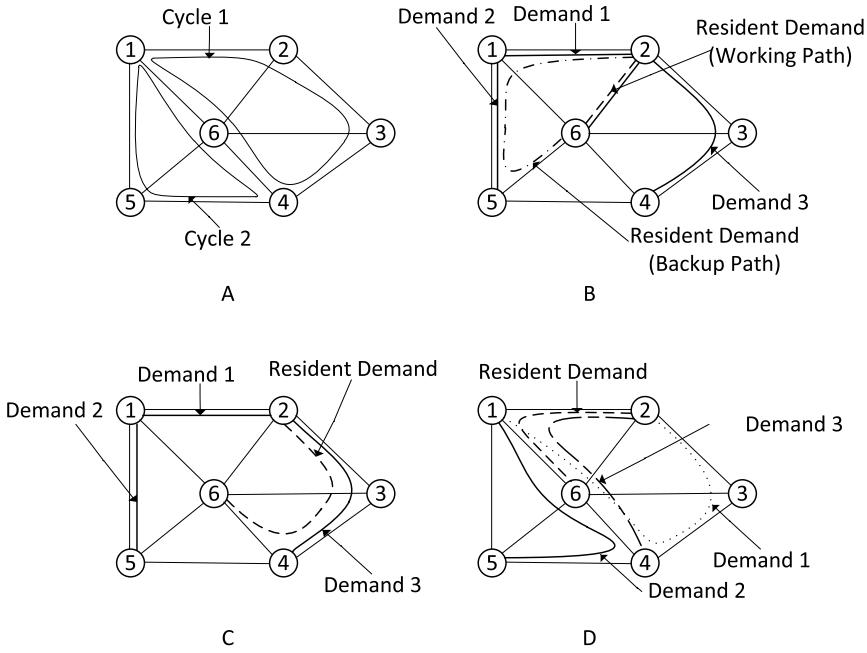

Figure 5.0.3: Adjusting Resident Traffic Paths: (A) Two cycles protecting an example network. (B) A resident demand and three new demands. (C) Working paths for the demands. (D) Backup paths for the demands. Note that the working and backup paths for the resident demand can be assigned to Cycle 1 and Cycle 2 (Cycle 1 in this example).

The effect on the energy efficiency of each method of handling resident traffic has to be explored. In the case of Adjusting Resident Traffic Paths, the effect on service to customers needs to be checked as well. When the resident traffic paths are changed, the old working path is interrupted, and the traffic is restored onto the old backup path. The demand is assigned the new working path and the traffic is restored onto the new working path. The old backup path is then replaced by the new backup path. When the old working path is interrupted,

there will be a very brief interruption in the service for the resident traffic that is having its working and backup paths changed. This interruption in service needs to be studied to determine if it will be long enough to be problematic to customers.

Another area for further study was found in the results of this thesis. The Modified Hybrid Shared algorithm provided an energy efficient set of paths for each network load in the simulations. However, the large amount of energy efficiency gained came at a cost of a very large loss in bandwidth efficiency. This was due to the size of the cycles used for assigning paths to the demands. If the size of the cycles is lowered, the bandwidth efficiency of the Modified Hybrid Shared algorithm can be increased at the cost of some of the energy efficiency.

The Power Efficient Growing Cycles algorithm provided a small increase in energy efficiency at the cost of a small loss in bandwidth efficiency. If the size the cycles could be grown to is changed, the energy efficiency of the Power Efficient Growing Cycles algorithm can be increased at the cost of some of the bandwidth efficiency. The ideal number of links the cycles can be grown to is dependent on the network in question and the nature of the traffic.

The Power Efficient Growing Cycles provided better energy/power efficiency in a large network with a high degree of connectivity and high traffic demand levels. The Modified Hybrid Shared algorithm provided better energy/power efficiency in a large network with a high degree of connectivity and low traffic demand levels and in networks with demands that are isolated between small pockets of nodes. Using the Modified Hybrid Shared algorithm for low traffic periods or where demands are isolated between small pockets of nodes, and the Power Efficient Growing Cycles algorithm for high traffic periods, may provide a more power efficient solution than using only the Modified Hybrid Shared algorithm or the Power Efficient Growing Cycles algorithm would. The Power Efficient Growing Cycles algorithm was also limited to one size for grown cycles. In this work, a cycle could not be grown to have more than five links. If a cycle that was larger than five links was needed it was assigned when the Power Efficient Growing Cycles algorithm looked for the smallest cycle that could support the demand. A study on the effect of the size a cycle could be grown to will provide more insight into the performance of the Power Efficient Growing Cycles algorithm and at what traffic demand levels would be best to switch between the Power Efficient Growing Cycles algorithm and Modified Hybrid Shared algorithm to achieve the best energy/power efficiency.

Lastly, a set of self similar test demands has to be calculated for a large time period, such as a few weeks, so that the power efficient routing scheme can be tested as a whole.

Chapter 6

Conclusions

6.1 Summary of Results

For the Global Crossing Network, which is a small network with a low degree of connectivity and a small number of cycles, the Modified Hybrid Shared algorithm performed the best and loaded the links as expected. The working paths were longer than the backup paths in order to increase the chances of sharing over links. This is much different from the Hybrid Shared algorithm which attempts to route backup paths for demands assigned to a cycle over the common links. By forcing as much backup traffic to flow over as many common links as possible, the number of links in sleep mode should increase. In large networks with traffic isolated between small pockets of nodes, like in the Kaleidoscope network simulation, the Hybrid Shared algorithm lead to the most links in sleep mode. In networks with traffic between all nodes, the Hybrid Shared algorithm has fewer links in sleep mode than the Dijkstra Shared Backup Paths protection benchmark algorithm. Therefore, routing backup paths over common links does not lead to more links in sleep mode when there are large levels of traffic between every node in the network. However, routing backup paths over common links does guarantee that there will be bandwidth for sharing with demands on other cycles, but can result in longer working paths than routing the working paths over the shortest possible paths would. Longer working paths leads to more bandwidth usage since, working bandwidth cannot be shared with other demands, and requires more links and nodes in online mode to handle the longer working paths. The Modified Hybrid Shared algorithm was created to address the bandwidth and longer working path issues of the Hybrid Shared algorithm. In networks with high levels of traffic between all nodes, allowing the working paths to flow over the shortest paths possible leads to shorter working

paths and also lower bandwidth usage than the Hybrid Shared algorithm. In networks with high levels of traffic between all nodes, the Modified Hybrid Shared algorithm leads to far fewer links in sleep mode than the Hybrid Shared algorithm. However, the Modified Hybrid Shared algorithm has far more links in offline mode than the Hybrid Shared algorithm and, therefore, is more energy/power efficient than the Hybrid Shared algorithm. The Modified Hybrid Shared algorithm successfully addresses the issues of the Hybrid Shared algorithm.

The Power Efficient Growing Cycles algorithm showed similar link loading characteristics to the Dijkstra Shared Backup Paths algorithm at low loads, loaded the links more than the Dijkstra Shared Backup Paths algorithm at higher loads and, as expected, the Power Efficient Growing Cycles algorithm had longer path lengths than the benchmark algorithms. In the Global Crossing Network, the Power Efficient Growing Cycles algorithm rejected fewer demands than the Dijkstra Shared Backup Paths algorithm. In the Random Topology Network, the Power Efficient Growing Cycles algorithm and the Dijkstra Shared Backup Paths algorithm had similar rejection characteristics. At low loads, the Power Efficient Growing Cycles and the Dijkstra Shared Backup Paths algorithms rejected the same number of demands. At higher loads the Power Efficient Growing Cycles algorithm rejected more demands than the Dijkstra Shared Backup Paths algorithm. The Power Efficient Growing Cycles algorithm has more links in offline mode than the benchmark algorithms. This is because the Power Efficient Growing Cycles algorithm routes demands around the straddling links of the cycles that are used to determine paths for traffic. The straddling links are then left free of traffic and placed into offline mode. Therefore, more links can be put into offline mode but at a cost of longer paths and lower bandwidth efficiency.

Another important characteristic of the three proposed algorithms is their ability to route traffic around nodes. In order to illustrate this ability, the Global Crossing and Random Topology networks were outfitted with "forbidden nodes" which could not be the source or destination for traffic. In both the Global Crossing and the Random Topology network simulations, the Modified Hybrid Shared algorithm had the most nodes in offline mode. The Power Efficient Growing Cycles algorithm had more nodes in offline mode than the benchmark algorithms in the Random Topology Network and the same number of nodes in offline mode as the benchmark algorithms in the Global Crossing Network. The number of nodes in offline mode was either the same or less than the benchmark algorithms in the Global Crossing Network. With the exception of one traffic demand level where the Modified Hybrid Shared algorithm had the most nodes in offline mode, the nodes in offline mode for the proposed algorithms were the same as the Dijkstra Dedicated Backup Paths algorithm and less than the Dijkstra Shared Backup Paths algorithm. The Modified Hybrid Shared algorithm had more nodes in offline

mode then the other algorithms because it takes all expected demands into account prior to selecting cycles and assigning the paths to the demands. The Power Efficient Growing Cycles and the benchmark algorithms assign paths based on a first come, first serve basis and do not benefit from knowing all demands in advance. Knowing the demands in advance enabled the Modified Hybrid Shared algorithm to select the cycles that were absolutely necessary to service all of the demands and minimized the overall number of links and nodes used for assigning paths and bandwidth to the demands. The Hybrid Shared algorithm did not have a significant difference in nodes in sleep mode and nodes in offline mode than the benchmark algorithms. In the Kaleidoscope Network simulation, the Modified Hybrid Shared algorithm had the most nodes in offline mode and the Hybrid Shared algorithm had the least nodes in offline mode. The Hybrid Shared algorithm had the most nodes in sleep mode with the third configuration but less than the Power Efficient Growing Cycles algorithm with the second configuration and the Modified Hybrid Shared algorithm had no links in sleep mode for any of the configurations of the Kaleidoscope Network.

The Hybrid Shared algorithm was the least efficient of the proposed algorithms and, due to the bandwidth and working path issues, sacrifices too much bandwidth efficiency for the small increase in energy/power efficiency it can provide over the benchmark algorithms. The Modified Hybrid Shared algorithm solves the bandwidth and working path issues of the Hybrid Shared algorithm and effectively replaces it. In small networks with low connectivity such as the Global Crossing Network, networks with traffic that is isolated to small pockets of nodes in the network, and large networks with a high degree of connectivity but low traffic demand levels, the Modified Hybrid Shared algorithm is the most energy/power efficient. The number of links and nodes in offline mode was much higher for the Modified Hybrid Shared algorithm than the number of links and nodes in sleep mode for the other algorithms in the Global Crossing Network and Kaleidoscope Network simulations at all traffic demand levels, and the Random Configuration Network simulation at low traffic demand levels. Since components in offline mode utilize much less power than components in sleep mode, the energy/power efficiency of the Modified Hybrid Shared algorithm is much higher. The Power Efficient Growing Cycles algorithm is the most energy/power efficient for large networks with a high degree of connectivity at high traffic demand levels and also rejects fewer demands than the Modified Hybrid Shared algorithm. Therefore, both the Modified Hybrid Shared algorithm and the Power Efficient Growing Cycles algorithm are useful for achieving energy/power efficiency but choice of which algorithm to use depends on the network and the expected traffic.

In small networks with a low degree of connectivity, the Modified Hybrid Shared algorithm would perform

best. In large networks with a high degree of connectivity and a large number of demands, the Power Efficient Growing Cycles algorithm is preferred. In large networks with a low number of demands, the Modified Hybrid Shared algorithm has the highest energy/power efficiency over the Power Efficient Growing Cycles algorithm. Network Traffic, as discussed in Section 2.2, varies over time and has high and low periods. With the use of network traffic prediction, the high and low periods of traffic can be known in advance and the appropriate algorithm to best handle the traffic can be selected.

Bibliography

[1] Ajmal Muhammad, Paolo Monti, Isabella Cerutti, Lena Wosinska, Piero Castoldi, and Anna Tzanakaki, "Energy-efficient WDM network planning with dedicated protection resources in sleep mode", *IEEE Globecom*, pp.1-5, Miami, Florida, USA, December 2010.

[2] M. Masud Hasan, Farid Farahmand, Ankitkumar N. Patel, and Jason P. Jue, "Traffic grooming in green optical networks", *Proc. IEEE International Conference on Communications (ICC)*, pp. 1-6, Cape Town, South Africa, May 2010.

[3] Pulak Chowdhury, Massimo Tornatore, Suman Sarkar, and Biswanath Mukherjee, "Towards green broadband access networks", *Proc. IEEE Globecom*, pp. 1-6, Honolulu, Hawaii, USA, November 2009.

[4] Gangxiang Shen and Rodney S. Tucker, "Energy-minimized design for IP over WDM networks", *Journal of Optical Communications Networks*, Volume 1, Number 1, pp. 176-186, June 2009.

[5] M. M. Hasan, Farid Farahmand, and Jason P. Jue, "Energy-awareness in dynamic traffic grooming", *Proc. Optical Fiber Communication (OFC), collocated National Fiber Optic Engineers Conference (NFOEC)*, pp. 1-3, San Diego, CA, USA, Mar 2010.

[6] I. Cerutti, L. Valcarenghi, P. Castoldi, "Power saving architectures for unidirectional WDM rings", *Optical Fiber Communication Conference, OSA Technical Digest (CD)*, Optical Society of America, San Diego, CA, USA, 2009.

[7] Yong Wu, Luca Chiaraviglio, Marco Mellia, and Fabio Neri, "Power-aware routing and wavelength assignment in optical networks", *Proc. ECOC* 20-24, pp. 1-2, Vienna, Austria, September 2009.

[8] Luca Chiaraviglo, Marco Mellia, and Fabio Neri, "Energy-aware backbone networks: a case study", *GreenComm'09 in conjunction with ICC*, pp. 1-5, Dresden, Germany, June 2009.

[9] Shuqiang Zhang, Dong, Shen, and Chun-Kit Chan, "Energy efficient time-aware traffic grooming in wavelength routing networks", *Proc. IEEE Globecom* pp.1-5, Miami, Florida, USA, December 2010.

[10] Cicek Cavdar, Feza Buzluca, and Lena Wosinska, "Energy-efficient design of survivable WDM networks with shared backup", *Proc. IEEE Globecom* pp.1-5, Miami, Florida, USA, December 2010.

[11] Maruti Gupta and Suresh Singh, "Greening of the internet", *Proc. ACM SIGCOMM*, pp. 19–26, Karlsruhe, Germany, August 2003.

[12] Joseph Chabarek, Joel Sommers, Paul Barford, Cristian Estan, Davit Tsiang, and Stephen Wright, "Power awareness in network design and routing", *Proc. IEE INFOCOM*, pp. 457-465, Phoenix, USA, April 2008.

[13] Emre Yetginer, and George N. Rouskas, "Power efficient traffic grooming in optical WDM networks", *Proc. IEEE GLOBECOM*, pp. 1838-1843, Honolulu, Hawaii, USA, November 2009.

[14] Shu Huang, Deepa Seshadri, and Rudra Dutta, "Traffic grooming: a changing role in green optical networks", *Proc. IEEE GLOBECOM*, pp. 5655–5660, Honolulu, Hawaii, USA, November 2009.

[15] David A. Patterson and John L. Hennessy, *Computer Organization and Design: The Hardware/Software Interface.* Fourth Ed. Elsevier inc, 2009.

[16] John L. Hennessy and David A. Patterson, *Computer Architecture: A Quantitative Approach.* Fourth Ed. Elsevier inc, 2007.

[17] Christensen, K. Reviriego, P. Nordman, B. Bennett, M. Mostowfi, and M. Maestro, J.A., "IEEE 802.3az: the road to energy efficient ethernet", *IEEE Communications Magazine*, vol. 48, no. 11, pp. 50–56, November 2010.

[18] Hanxi Zhang and Oliver Yang, "Finding protection cycles in DWDM networks", *IEEE* International Conference on Communications (ICC), vol.5, pp. 2756-2760, New York, NY, USA, May 2002.

[19] Lu Ruan and Fangcheng Tang, "Dynamic establishment of restorable connections using p-cycle protection in WDM networks", *IEEE 2nd International Conference on Broadband Networks (BroadNets)*, pp. 147-154, Boston, MA, USA, October 2005.

[20] Dominic A. Schupke, Matthais C. Scheffel, and Wayne D. Grover, "Configuration of p-cycles in WDM networks with partial wavelength conversion", *Photonic Network Communications*, Volume 6, Number 3, pp. 239-252, June 2003.

[21] Dominic A. Schupke, "On hamiltonian cycles as optimal p-cycles", *IEEE Communications Letters*, Volume 9, Number 4, pp. 360–362, April 2005.

[22] Wayne D. Grover, "P-cycles, ring-mesh hybrids, and "ring mining:" options for new evolving optical transport networks", *Proc. Optical Fibre Communications (OFC'03)*, Volume 1, pp. 201–203, Atlanta, GA, USA, March 2003.

[23] Guoliang Xue and Ravi Gottapu, "Efficient construction of virtual p-cycles protecting all cycle-protectable working links", *IEEE Workshop on High Performance Switching and Routing*, Pages 305-309, Torino, Italy, June 2003.

[24] Dominic A. Schupke, Matthais C. Scheffel, and Wayne D. Grover, "An efficient strategy for wavelength conversion in WDM p-cycle networks", *Design of Reliable Communication Networks*, Banff, Alberta, Canada, October 19-22, 2003.

[25] Wayne D. Grover and John Douchette, "Advances in optical network design with p-cycles: joint optimization and pre-selection of candidate p-cycles", *Proceedings of IEEE/LEOS Summer Topicalss*, Pages 49-50 (paper WA2), Quebec, Canada, July 2002.

[26] Diane Prisca Onguetou and Wayne D. Grover, "Solution of a 200-node-p-cycle network design problem with GA-based pre-selection of candidate structures", *IEEE International Conference on Communications (ICC)*, pp 1-5, Dresden, Germany, June 2009.

[27] Francois J. Blouin, Anthon Sack, and Wayne D. Grover, "Benefits of p-cycles in a mixed protection and restoration approach", *Design of Reliable Communication Networks*, pp. 203-210, Banff, Alberta, Canada, October 2003.

[28] Tianjian Li and Bin Wang, "Optimal configuration of p-cycles in WDM optical networks with sparse wavelength conversion", *IEEE Globecom*, Volume 3, pp. 2024-2028, Dallas, Texas, USA, Dec 2004.

[29] Gangxiang Shen and Wayne D. Grover, "Extending the p-cycle concept to path segment protection for span and node failure recovery", *IEEE Journal On Selected Areas In Communication*, Volume 21, Number 8, October 2003.

[30] Wayne D. Grover, *Mesh-Based Survivable Networks: Options and Strategies for Optical, MPLS, SONET, and ATM Networking.* Prentice Hall, August 2003.

[31] Eric Bouillet, Georgios Ellinas, Jean-Francois Labourdette, and Ramu Ramamurthy, *Path Routing In Mesh Optical Networks.* John Wiley & Sons Ltd. 2007.

[32] Huifang Feng and Yantai Shu, "Study on network traffic prediction techniques", *Proc. IEEE International Conference on Wireless Communications, Networking, and Mobile Computing,* Pages 991-994, Honolulu, Hawaii, USA, September 2005.

[33] Geyong Min, Xiaolong Jin, and Speros/Ross Velentzas, "Performance analysis of the guard channel scheme with self-similar call arrivals in wireless mobile networks", *IEEE Globecom*, pp.5032-5036, New Orleans, LA, USA, December 2008.

[34] Jakob Carlstrom and Ernst Nordstrom, "Reinforcement learning for control of self-similar call traffic in broadband networks", *Proceedings of the 16th International Teletraffic Congress (ITC-16)*, pp. 571-580, Edinburgh, Scotland, June 1999.

[35] Ashok Erramilli, Matthew Roughan, Darryl Veitch, and Walter Willinger, "Self-similar traffic and network dynamics", *Proceedings of the IEEE,* Volume 90, Number 5, pp. 800-819, May 2002.

[36] Will E. Leland, Murad S. Taqqu, Walter Willinger, and Daniel V. Wilson, "On the self-similar nature of ethernet traffic (extended version)", *IEEE/ACM Transactions on Networking*, Volume 2, Number 1, pp. 1-15, February 1994.

[37] Michael Jiang, Milan Nikolic, Stephen Hardy, and Ljiljana Trajkovic, "Impact of self-similarity on wireless data network performance", *IEEE International Conference on Communications (ICC)*, Vol. 2, No. 11–14, pp. 477–481, Beijing, China, May 2001.

[38] Bo Zhu, Dan He, Zhili Sun, and Wee Hock Ng, "Network traffic modeling and prediction with ARIMA/GARCH", *Third International Working Conference: Performance Modeling and Evaluation of Heterogeneous Networks*, Ilkley, West Yorkshire, U.K., July 18 – 20, July 2005.

[39] N. Swaminathan, J. Srinivasan, and S.V. Raghavan, "Bandwidth-demand prediction in virtual path in ATM networks using genetic algorithms", *Computer Communications*, pages 1127-1135, Elsevier, 1999.

[40] Fang-Mei Tseng, Hsiao-Cheng Yu, and Gwo-Hsiung Tzeng, "Combining neural network model with seasonal time series ARIMA model", *Technological Forecasting & Social Change*, Volume 69, pages 71-87, North-Holland, 2002.

[41] Ramesh Bhandari, *Survivable Networks: Algorithms for Diverse Routing*, Kluwer Academic Publishers, 1999.

[42] Xiaojun Cao, Vishal Anand, and Chunming Qiao, "Multi-layer versus single-layer optical cross-connect architectures for waveband switching", *IEEE Infocom*, pp.1830 2004, Hong Kong, China, March 2004.

[43] Xiaojun Cao, and Chunming Qiao, "Waveband switching for dynamic traffic demands in multi-granular optical networks", *IEEE/ACM Transactions on Networking*, Volume 15, Number 5, Pages 957-968, October 2007.

[44] Nabil A. Naas, "Towards the realistic planning of the GMPLS-based optical transport networks", Ph.d dissertation, Electrical Engineering, University of Ottawa, Ottawa, Canada, 2008.