University of Alberta

The Use of Demand-wise Shared Protection in Creating Topology Optimized High Availability Networks

by

Brody James Todd

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science
in
Engineering Management

Mechanical Engineering

©Brody Todd
Fall 2009
Edmonton, Alberta

Permission is hereby granted to the University of Alberta Libraries to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only. Where the thesis is converted to, or otherwise made available in digital form, the University of Alberta will advise potential users of the thesis of these terms.

The author reserves all other publication and other rights in association with the copyright in the thesis and, except as herein before provided, neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatsoever without the author's prior written permission.

Examining Committee

John Doucette, Mechanical Engineering

Michael Lipsett, Mechanical Engineering

Peter Flynn, Mechanical Engineering

Armann Ingolfsson, Finance and Management Science

To Michelle, your love and support has meant everything.

To my son, Paxton, may the opportunities that have been awarded to me continue to be available for your generation.

Abstract

In order to meet the availability requirements of modern communication networks, a number of survivability techniques were developed that adapt the demand-wise shared protection network design model to incorporate strategies increasing network availability. The survivability methodologies developed took two approaches. The first approach incorporated availability directly into the network design model. The second approach ensured minimum dual failure restorability was set within the model. These methodologies were developed for predetermined topologies, as well as to have topology optimization incorporated into the model.

All methodologies were implemented and analyzed on a set of samples. The analysis examined the cost, topology and actual availability of the network designs. Availability design was effective but computationally intensive and difficult to design. Minimum dual failure restorability was also effective in increasing availability with a significant caveat. Designing for dual failure restorability increased traffic exposure to possible failures, and without sufficient levels of dual failure restorability could have a negative impact on availability.

Acknowledgements

This work has been made possible through the generosity and support of a number of people and organizations.

First, I would like to acknowledge my supervisor, Dr. John Doucette. The guidance and encouragement you have provided has been appreciated. The freedom you provided allowed me to ample opportunity to explore and discover through this research. Your encouragement and insight has constantly pushed me to improve. Thank you.

To all of you who have worked alongside me, you have enriched my experience greatly. Our interactions have given me context, not only for this work, but for education and the existential meaning of life. Your diversity of backgrounds and the sacrifices you made for your education has been a great inspiration.

TR-Labs and the University of Alberta provided an amazing environment to study and work. To the people involved, thank you for your efforts in establishing and growing such a great environment.

For the financial support in undertaking this work, I would like to thank Dr. John Doucette, TR Labs, NSERC and the University of Alberta. The financial stability provided while undertaking this research has been a key foundation to this final product.

Finally I would like to thank my family. You have made this journey with me, thank you for all your love and support.

Contents

List of Figures

List of Abbreviations

APS	Automatic Protection Switching
DSP	Demand-Wise Shared Protection
DSP-Top	Demand-Wise Shared Protection with Topology Design
DSP-Top-A	Demand-Wise Shared Protection with Topology Design and Availability
DSP-Top-R2	Demand-Wise Shared Protection with Topology Design and Dual Failure Restorability
DSP-TR	Demand-Wise Shared Protection using the Transportation Problem
DSP-TR-A	Demand-Wise Shared Protection using the Transportation Problem with Availability
DSP-TR-R2	Demand-Wise Shared Protection using the Transportation Problem with Dual Failure Restorability
ILP	Integer Linear Program
JCA	Joint Capacity Allocation
LAN	Local Area Network
MPLS	Multi Protocol Label Switching
MTBF	Mean Time Between Failures
MTTF	Mean Time To Failure
MTTR	Mean Time To Repair
QoS	Quality of Service

R2	Dual Failure Restorability
RACA	Restoration Aware Connection Availability
SBPP	Shared Backup Path Protection
SCA	Space Capacity Allocation
SLA	Service Level Agreement
WAN	Wide Area Network

Chapter 1. Introduction

The past 20 years has seen an explosion in communication technologies. The growth in our ability to communicate has largely been enabled by significant increases in the amount of information communication networks can route. From government to banking to personal communication, the ability to communicate efficiently and reliably has enhanced how we live and work to such a degree that it is hard to imagine life without it.

With so much reliance on the communication infrastructure, reliability and availability are significant concerns for all the aspects of the communication infrastructure. Failure in the core communication network can have significant impacts. These failures can have immense economic and social consequences. As the use of technology increases, the reliance on the communication infrastructure to provide highly reliable communication is only growing.

One area that will put significant pressure on the communication networks is the move of healthcare into eHealth [1]. eHealth is a label for the diverse applications that are being developed for the healthcare industry to improve patient care and reduce costs. These initiatives include electronic health records, remote doctor patient consultations, digital imaging, telesurgery, and many others. All these technologies are predicated on having a reliable communication infrastructure, although they vary in the degree of reliability they require. eHealth is one example of the potential of utilizing communication systems to work and live better, and there are many other areas of society that are affected in a similar manner.

It is hard to find an industry or part of society that is not poised to be significantly impacted (or is currently undergoing significant change) by new technologies that rely on reliable high-speed communication channels.

While end users see the communication systems they use as separate distinct networks, the trend in industry is toward converged networks [1]. These converged networks aggregate communication traffic from many disparate sources, such as cell phones, land based telephones, Internet traffic, government and industry communications and others. Since so many systems and people are affected by failures, availability and reliability in these core networks are paramount.

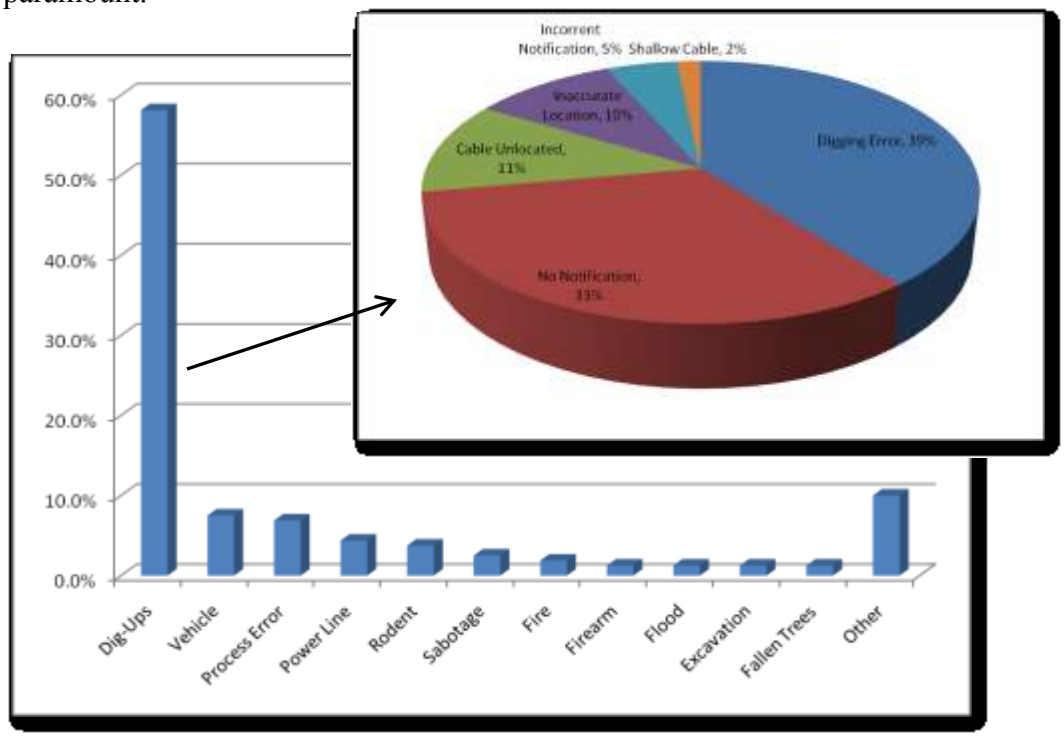


Figure 1 – Common Sources of Network Failure [2][3]

Failure rates in the communication lines (spans) occur at a rate of around 3 failures per 1000 km per year for long haul lines, and 12 failures per 1000 km per year for metropolitan lines. Repair times for a cable break average around 12 hours. If a communication link between Vancouver and Toronto was left unprotected, that would mean that communication between these two centers would be out for 6 days per year (12 outages of 12 hours). A recent failure in

Newfoundland caused the 911 service to be unavailable for 3 hours [5]. Three of the primary fiber cables connecting the Middle East and Asia to Europe were recently cut, eliminating 80% of the communication ability between the two regions. This significantly disrupted many areas, including the ability for outsourcing facilities in India to service their customers in Europe[6]. Overall network failures can have a significant influence on a broad portion of society.

When the impacts of failure of the various aspects of the communication network are evaluated, it is failures in the links between the network hubs that are commonly considered in survivable network design, as they are the largest contributor to network failures[7][8], although there are some *survivability mechanisms* that explicitly protect against node failures[9][10][11]. The hubs are mostly located inside regulated environments, where the availability of the hub can be controlled through redundancy and protection measures within the hub. The spans that connect these hubs, however, are generally in an environment outside of the control of the network operator. Located in utility right of ways, along the ocean floor, and hung on utility poles, these spans are open to a number of sources of failure. [13] documents the most common sources of failure in a communication network.

The spans in the network contribute significantly to the downtime of a network, and because these elements of the network fail, even when design of the individual components attempts to protect from failures, the ability to re-route traffic when a failure occurs is the primary tool to increase uptime.

Having a core communications infrastructure that can provide high availability communication links is fundamental for the next generation of business, social and government applications. While there are many dimensions to provisioning and managing these networks, one of the essentials is for the network to be able to repair itself at a low level without significant or long term propagation of failure through the rest of the network stack. The design and allocation of resources at this level that has failure resiliency is therefore a fundamental component of high availability networks.

Chapter 2. Background

The design of reliable communication networks requires consideration throughout the network protocol stack. Each protocol layer provides mechanisms with various *quality of service* (QoS) abilities relevant to the scope of the layer [12]. Each layer of the network affects the overall quality of communications in different ways, with each layer providing only as good of connection as the layer below it. This means that the significance of reliable communications increases for lower layers of the network protocol stack. This work focuses on the design paradigms that increase the failure resiliency in the lower levels of the network stack.

Networks are generally broken down into *local area networks*(LAN), and *wide area networks*(WAN), with local networks providing connectivity to individual users, while the *wide area networks* providing aggregated transportation services to the local networks. These WAN networks have been, themselves, aggregated over recent years, using technologies such as *multi-protocol label switching* (MPLS), so that the diversity of LANs and their respective protocols, can utilize a common WAN infrastructure.

As the capacity of the WAN or backbone infrastructure has increased, so has the value, or significance of the traffic that utilize it. This common infrastructure carries a diversity of communications, from residential Internet, telephone, cellular phone, bank transactions, government and corporate communication, and many more. Many of the current and upcoming technologies that utilize this network have significant requirements on the availability of the network and outages have a significant economic and social impact.

Disruptions in the core backbone network can occur due to a range of failures caused by a wide range sources [2]. These failures can be categorized at a high level as failures occurring at network hubs, and failures occurring on the network cabling. The network disruptions that are of relevance to the work contained herein generally occur due to failure in the cables connecting the network hubs. These cables are referred to as *spans*, and the network hubs as *nodes*. Generally the network equipment operating at the nodes is contained in an environment that is largely controlled by the network operator, providing greater assurance that environmental factors will not affect the performance of the equipment, and allowing easier maintenance of the equipment to prevent failures. Network spans travel through diverse environments, often outside the control of the network operator, and co-located with other piping and cabling through utility corridors [14]**.** As such these spans are exposed to a diverse range of possible failures. The most common of these failures is due to the cables being dug up, or cut during construction or maintenance on nearby structures [13]. However, these cables can fail due to rodents, sharks, and ship anchors, to name a few sources that have made headlines. In general, network spans experience a failure rate of 3 failures per 1000 km of routed fiber per year for long haul spans, and 12 failures per year for spans based in metropolitan areas [7].

Discussion on designing networks for survivability started in a significant manner during the 1980s. As fiber optic communications technologies started to become prominent, a level of reliability in these networks that would meet the burgeoning demand would have to be incorporated into the lower levels of these networks. Data rates using fiber optics went from 6 Mbps in 1977 to 1.7 Gbps by 1987. The bandwidth available using fiber optics has continued to grow, with current networks being able to transport information in the Tbps range [15].

The most basic and obvious form of survivability in a network is to have two diverse paths dedicated to each node pair demand. This form of protection is generally known as 1+1 a*utomatic protection switching* (APS). 1+1 APS simultaneously routes data on two disjoint paths at the ingress node, with the

egress node utilizing the better of the two signals (Figure 2). While 1+1 APS is simple, it requires, at a minimum, 100% redundancy of spare capacity in a network (generally more, since the redundant path is usually longer than the shortest path between any two nodes).

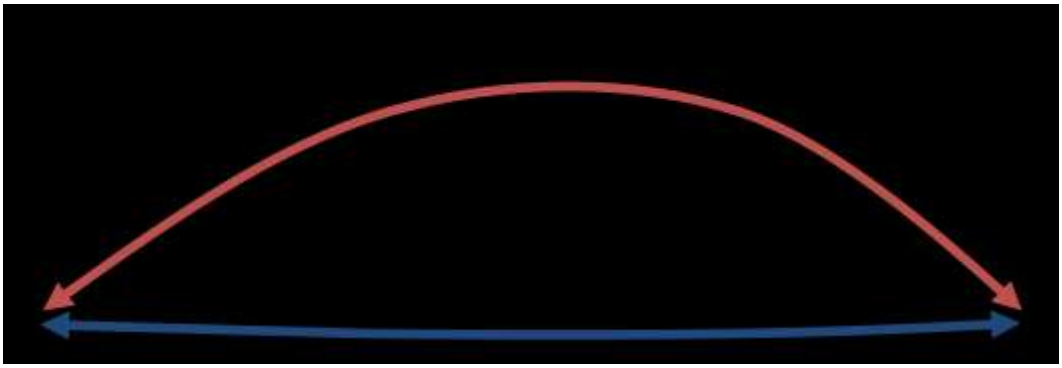


Figure 2 – Example of 1+1 APS routing

Another form of protection employed early on, and still used today, is *ring-based* protection (Figure 3). Ring-based protection is based on routing traffic on a ring or series of rings that connects all the nodes in the network together (or, for larger networks, a set of rings that cover the network with predefined crossover nodes). These rings would have traffic travel in one direction (unidirectional path switched rings) or both directions (bidirectional line-switched rings). The idea behind ring-based protection that allows a ring to be resilient in the case of a failure is that every connection being routed by the ring is bi-connected. Using various mechanisms these rings can re-route traffic that has failed at the lightpath level with little effect on the higher network layers. Ring protection also requires a significant portion of its total capacity dedicated as spare capacity to be utilized when a failure occurs. Redundancy of ring-based networks is at a minimum 100% [13], but less than 1+1 APS due to a limited amount of spare capacity that can be reused depending on the failure scenario.

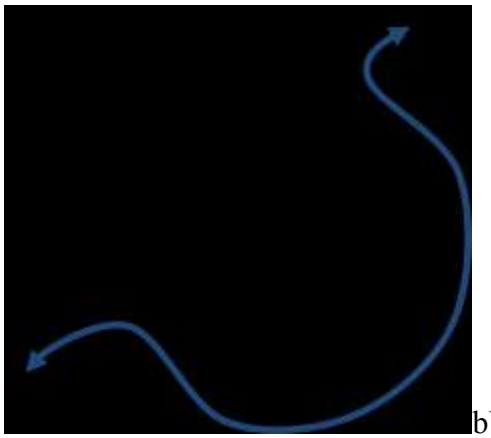
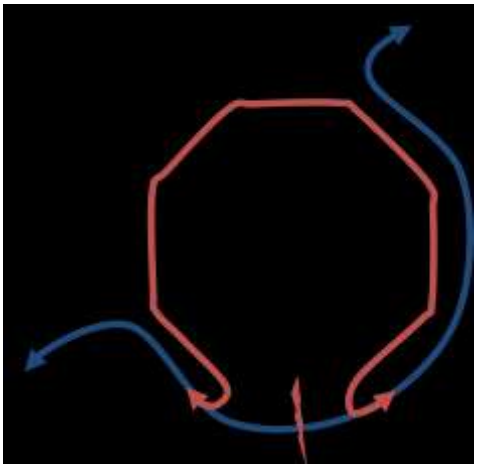
a)  b) 

Figure 3 – Bidirectional Line-Switched Ring Protection in a) a working state, and b) a failed state

In order to reduce the amount of spare capacity in a network, this spare capacity must be shared. This sharing can occur between failure scenarios, demand paths, or paths in general without regard to the source and destination. Routing schemes that allow capacity sharing of this kind fall under the general category of *mesh networks*, where mesh networks are contrasted by the previously mentioned ring networks because the network is viewed as a mesh of spans that could be utilized for any demand path. By allowing spare capacity to protect multiple sources, it cannot be pre-connected, and routing must be established when a failure occurs. This extra overhead increases the restoration time to anywhere from 200 ms to 2 sec [13]. This extra restoration time has little effect on availability and can usually be recovered easily using higher network layer protocols. These shared capacity survivability schemes generally fall into two categories, defined by the vantage point from which the failure is addressed, namely, span- and path-based survivability. Each category has a variety of different mechanisms associated with them, and each has their advantages and disadvantages.

2.1 Restoration and Protection

2.1.1 Span Restoration and Protection

Span based protection attempts to reconnect end nodes of a failed span. While there are a variety of mechanisms that protect the network in such a manner, there are two that are prominent in literature, *span restoration*, and *p-cycles*.

The first, usually called span restoration, uses dynamically allocated paths routed elsewhere in the network to reconnect the two end nodes of the failed span [16] (Figure 4). These paths are generally predefined; however, because spans used in the restoration path are available to multiple failure scenarios, they are not pre-connected. A signaling mechanism is needed to connect each span in the path. This survivability scheme may also utilize multiple restoration paths in order to better utilize spare capacity. Span restoration is more efficient than 1+1 APS and ring-based routing, and also is advantageous in that restoration can be performed without global knowledge of the network. The drawbacks of this scheme are in the complexity of its recovery paths, interactions in multiple failure scenarios, and the time required to set up the restoration paths. The complexity of the recovery paths refers to the amount of signaling required to setup the path, the difficulty in transferring the network back into its working state, and the ability to handle multiple failures.

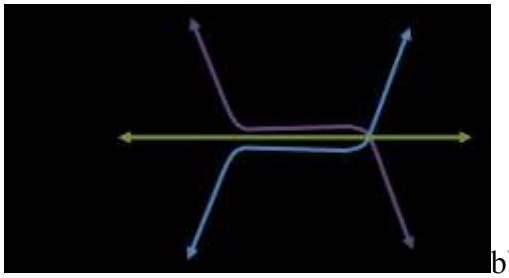
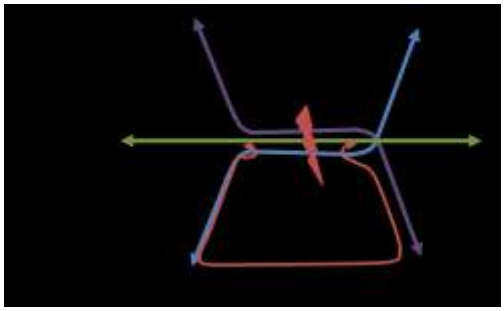
a)  b) 

Figure 4 – Span Restoration in a working (a) and failed (b) state

Another prominent survivability mechanism that restores traffic to the end nodes of a failed span is called p-cycles. p-Cycles use pre-connected rings in the network to restore failed capacity. Working capacity is generally routed on the shortest path in this scheme. These protection cycles gain their capacity efficiency by providing two units of restoration capacity for every one unit of working capacity when the failed span straddles the cycle (i.e. the end nodes are part of the cycle, but the span itself is not). p-Cycles provide lower capacity redundancy compared to ring based survivable networks, and quicker and simpler restoration compared to span restoration [13].

2.1.2 End to End Restoration and Protection

There are a number of restoration mechanisms that restore failed traffic at the origin and destination of each of the paths affected by a failure. 1+1 APS, already mentioned, is one of the more common path restoration methods currently in use. Others allow spare capacity to be shared between paths routing traffic from the same demand, or more generally, between any disjoint working path servicing any demand. Examples of the former are M:N APS, and *demand-wise shared protection* (DSP) [17], which is the focus of the studies in subsequent sections. *Path restoration*, and *shared backup path protection* (SBPP) [18] are examples of the latter. In path-based restoration, there is generally a number of paths dedicated to routing traffic under normal operating conditions. These paths are referred to as working paths. The paths which are utilized to reroute traffic when a working path fails are called restoration paths.

1:1 APS is a variation on 1+1 APS, with the difference being that traffic is not simultaneously transmitted on the second path (making it available for low priority traffic). Generalizing the 1:1 APS is the M:N APS scheme. In this scheme, M restoration paths are provided to restore capacity from N working paths. In this scheme, the only requirements on the paths are that the set of M paths are disjoint from the N working paths. All paths are pre-connected, and do not require any action by nodes other than the end nodes of the working path.

SBPP is similar to 1:1 APS, except the spare capacity for a restoration path is also available to any other restoration path from any other demand, so long as the working paths of each restoration path that shares a common span are disjoint (Figure 5). This sharing significantly reduces the spare capacity required in the network compared to any of the previously mentioned survivability mechanisms. Work has also been done to devise an SBPP survivability model that utilizes multiple working and spare paths per demand [19]. While capacity efficient, SBPP requires each node along the preplanned restoration paths to connect the assigned path, creating complex restoration states and possible difficulties in transferring back to a network state utilizing solely working paths.

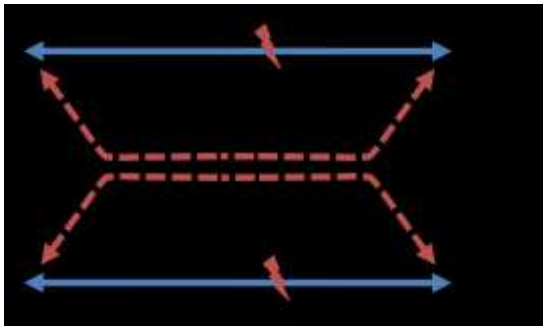
a)  b) 

Figure 5 – Shared Backup Path Protection in a working state (a) and two possible failure states (b) demonstrating capacity sharing

Another prominent path-based survivability mechanism is generically called path restoration. Path restoration operates in a similar manner to SBPP, except that it also utilizes a stub release mechanism. This allows restoration paths to utilize working capacity on the spans of the failed path that is not actually involved in the physical failure. This further reduces the capacity redundancy when compared to SBPP. However, the selection of restoration paths for each failed working path requires knowledge of where the failure occurred. Path restoration, while extremely efficient also carries significant complexity costs.

The last end-to-end restoration and protection model that will be mentioned here is DSP. DSP is discussed in further detail in the following section, as it is the mechanism studied in this work. It is an end-to-end restoration mechanism that attempts to balance complexity and capacity redundancy.

2.1.3 Demand-wise Shared Protection

As mentioned, each survivability scheme comes with tradeoffs in capacity efficiency, complexity, and restoration timing. In an attempt to find an appropriate balance between capacity efficiency and complexity, DSP was developed that attempts to keep the complexity of transitioning between failed and working states similar to that of 1+1 APS, or more specifically M:N APS, while capitalizing on the capacity efficiency of path restoration.

DSP provides a straightforward protection scheme that focuses on survivability from the vantage point of a single demand. By provisioning multiple distinct or disjoint routes for each demand node pair, spare capacity can be more efficiently shared when compared to 1+1 APS. The concept of DSP was first introduced in [17].

On the left hand side of Figure 6 an example of 1+1 APS between two end nodes with a demand of 4 units is given. In this example there are 8 units of capacity assigned to the spans in the network, with 4 for the working and 4 for the backup route.

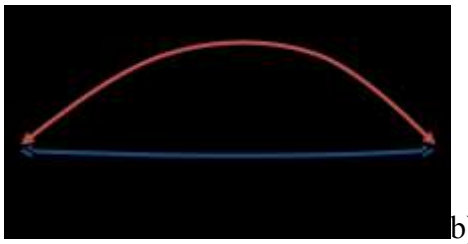
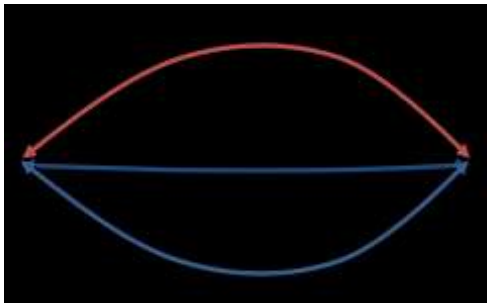
a)  b) 

Figure 6 – 1+1 APS routing (a) and DSP routing (b)

Instead of routing all capacity along a single working route in a network and protecting it with one or more alternate backup routes, DSP uses multiple disjoint working routes and protects them with a single backup route (or more than one backup route if multi-failure restorability and/or higher levels of availability are required). On the right side of Figure 6 the same demand is routed using DSP, and takes 6 units of capacity, reducing the overall cost of routing the given demand.

Two of the routes are used for working, and if either of them fails, the 3rd route is used to continue routing the traffic.

DSP is similar to M:N APS, DSP differentiates itself by enforcing the working paths to be node disjoint, and therefore is able to capitalize on the spreading of the capacity on working paths by reducing the required capacity assigned to the spare paths.

While there are many more survivability mechanisms, most are variations of one of these. All have tradeoffs between spare capacity redundancy, restoration complexity, and the time required to recover the failed traffic.

2.2 Design paradigms

When designing a network using one of these techniques, there are a number of different paradigms from which a survivability scheme may be applied. For the path-based schemes, there is *spare capacity allocation* (SCA) where the working paths are fixed, and only the routing and capacity allocation of spare capacity is required. In contrast to SCA is *joint capacity allocation* (JCA). JCA attempts to concurrently find the best paths for working and spare capacity.

Many papers have been written on applying these survivability techniques on networks that already have fixed maximum span capacities, attempting to route as much of the required demand as possible [21]. Other work attempts to minimize the total capacity in the network while meeting all of the demand requirements. Both of these design paradigms can be utilized when talking about network survivability design.

There is also the challenge of applying these schemes dynamically, being able to add or drop demand, as the circumstances and traffic patterns change. Another paradigm that these survivability schemes are applied is in the allocation of actual spans in a network, usually called topology design. The common

implementations of these survivability mechanisms are in JCA, SCA, fixed capacity, dynamic allocation, and topology design.

2.3 Availability Design and Analysis

2.3.1 Availability Background

As the impact of an outage in a network becomes more significant, the idea of *service level agreements* (SLAs) has become the prominent method of communicating expectations between the user and the network operator [22],[23]. In order to define an effective SLA, both the user and the operator need to be able to quantitatively define the performance requirements of the network connection, and the costs associated with meeting/not meeting these requirements. These requirements include many different measurements associated with network communication. Examples of these can include security, jitter, bandwidth, latency, and outage time. This discussion focuses on outage time, as this is the factor that is primarily affected by network design problems.

If users can quantify the cost of network outages, they can also quantify how much they are willing to spend in order to guarantee that the network will be operating at the required standard. For an example, if a user is looking to set up a connection that will support remote surgery [24], the impact of failure in the network is so significant that there must be virtually no risk of a disruption regardless of cost. If a business has work groups with members at various sites, the cost of an outage of a couple of hours would impact the productivity of the group members. The cost of this loss of productivity would have to be quantified in order to accurately determine the level of service required. In general, as users are able to quantify the impact of network outages in monetary terms, the network operators have more incentive to provide a cost to a differentiated level of service, and as such must incorporate availability considerations in their network planning.

From a network operator's perspective, outage time is represented in the availability of a given connection. In order to predict the expected outage time for a given time period, the operator needs to understand the probabilities of a failure in the network, and the effect which restoration has on the ability of a network to continue to route traffic. Provisioning spare capacity in the network to protect against more than a single span failure can be expensive, with the potential of increasing costs by easily more than 100% of the cost to route traffic without any protection, depending on the survivability scheme chosen. How to define and determine availability for different network designs and configuration is a question that is highly relative to the network user. The design of a network to meet the availability requirements, in as cost effective a manner as possible, is a concern that can be, at least somewhat be met through incorporating survivability into network design and capacity allocation.

2.4 Availability Calculations

Availability is generally understood to be the probability of finding the system in working condition after time *t*. When *t* is sufficiently large availability converges to a given level, generally called *steady-state availability*. This steady-state availability is what is meant when discussing availability in the context of network design [13]. Although the terms "availability" and "reliability" are commonly interchanged when talking about the resiliency to failure of a system, it should be noted that there is a distinction. Reliability is generally a mission orientated term measuring the probability of a system to run without interruption for a specific period of time, or to complete a specific task. Availability on the other hand is more generally understood to be the portion of time a system spends in an operational state.

Availability can be defined in a variety of ways depending on what is included in the system from which availability is calculated. Of concern for the user, and hence the concern of the network operator, is the availability of each communication demand in the network. Availability, in order to be relevant to an

SLA agreement, is measured at the demand level. Availability could also be measured at the network level, and while this may be an overall performance indicator of the network, it leaves a significant amount of ambiguity if used as a requirement when designing and capacitating a network. A third scope from which availability may be measured is by per unit of capacity (i.e. per lightpath). For an optical network, if each lightpath required by a demand was enumerated, each one may have a unique availability. This is caused by having partial restorability of a node pair's traffic demand under multiple concurrent failures. Some lightpaths will be restorable, and hence have a higher availability, than others in the same demand. This leads to differentiated service within a given demand. If lightpath availability requirements were to be directly translated to network design requirements, effectively making each demand in the network a single unit of capacity, network management would become unwieldy, and significantly more complex. The individual availability requirements of each unit of demand can be translated into an overall demand availability requirement without losing a significant amount of control.

When examining network availability, there are a variety of levels of granularity from which the network may be examined. The decision of how much detail in which to break down the subsystems of the network can be based on the accuracy of the component availability estimations. If a component has a relatively large unavailability (when compared to sub components), with a significant margin of error, including other peer subsystems with a small availability would be a merely academic exercise with little impact on the quality of the results. While work has been done that goes into significant details of the error contributions of individual parts of the network[7], it is generally the goal of survivable network design methodologies to protect against span failures (although some do protect against node failures, either explicitly, or by default). As such, when calculating availability for the purposes of network design, only span failures are generally considered. This approach is acceptable, since in practice the failure rate of spans is significantly greater than nodes [7]. Spans are considered a single component in this work, although they could be further broken down into individual fibers,

and repeaters [7]. The failure rate used in the calculations contained in this paper are estimated to be 3 failures per year per 1000 km of fiber cable, and the average time to repair of these failures is 12 hours. 3 failures per year per 1000 km of cable translates into a *mean time to failure* (MTTF) of 2 920 000 hours per km of cable.

Although the method of calculation of availability depends on the survivability mechanism used, calculations may be characterized either looking at span failures, or path failures (as an aggregate of the failures in each span). [8] looks at availability from a span restoration point of view, and [25]-[26] examine availability from a variety of path based restoration mechanisms.

Availability can be calculated for a repairable system using equation (1). Figure 7 outlines the timing used in calculating average availability with the MTTF, the *mean time between failures* (MTBF), and *the mean time to repair* (MTTR). (1) is commonly used to empirically calculate availability using historical data. However, in order estimate availability, where component reliability is generally known, a more in-depth examination of the calculation of availability is needed. Work has been done to simplify availability calculations, such as [27]. [25] presents a formula that calculates availability for a path based on each dual failure scenario. [28] presents a method of calculating availability based on each individual span's apparent unavailability (2). This apparent unavailability represents the physical unavailability of the span, along with the effect of restorability on the traffic crossing span *i*. This provides a generic method of calculating availability for any path, U_p, encapsulating the effect of the restoration mechanism in the calculation of the apparent span unavailability, U_i^*.

$$A = \frac{MTTF}{MTTR+MTTF} \tag{1}$$

$$U_p = \sum_{i \epsilon S} U_i^* \tag{2}$$

Figure 7 – Where MTTR, MTTF, and MTBF fit in the failure timeline

When looking at a path-based survivability mechanism, demand availability can be approached as if the working and spare paths are part of a parallel redundant system. Many papers dealing with availability in network systems base the calculations on dual failure analysis [25][26]. Because a network's MTTR is orders of magnitude smaller than its MTTF, the number of failures one above what the network was designed to protect will dominate the availability calculation. Since most literature up to this point has dealt with single failure restorability, most availability analysis was done by calculating the effect of dual failures on the network, and ignored the effect of 3 or more concurrent failures.

Work has been done on span restoration availability [8]; however, the remaining discussion on availability will assume a path restoration mechanism.

Markov availability analysis provides a basis to accurately calculate the availability of a given demand. In order to determine availability, the steady state probability of all (or at least the dominant) the failure states must be determined. The Markovian analysis is an effective method in determining this.

The first step in using Markovian methods to calculate availability is to create a state diagram, where each state is a unique possible combination of failures of components of a network. This can be either elemental failures, such as the failure of a span in the network, or subsystem failures, such as the failure of a path in the demand. The next step is to determine the steady state probabilities of the network being in a given state. This is determined by setting up a Markov chain (describes the failure states of a system, and the state transitions) of all the states,

S. (3) calculates the number of failure states that would be in a system with k components where P_k^n is the *n* pick *k* operator. For example, (4) calculates the number of states for a demand with three paths (treating each path as congruent entity).

$$|S| = \sum_{k=0}^{n} P_k^n \tag{3}$$

$$P_0^3 + P_1^3 + P_2^3 + P_3^3 = 16 \tag{4}$$

Once the paths are enumerated, the transition rates between each must be identified. The transition rates are the rates of failure and repair corresponding to the appropriate change in the failure state. Figure 8 represents the Markov chain for a demand between O and D in the network displayed in Figure 9. The transitions were separated into failures (a) and repairs (b) for clarity. Three paths between *O* and *D* are also shown in Figure 9. The transition rates when transitioning to a state that represents adding a failure is the inverse of the MTTF for the newly failed component (5). The transition rate when the new state is a reduction in the number of failed component is the inverse of the MTTR (6).

$$\lambda_i = \frac{1}{MTTF_i} \forall i \in N \tag{5}$$

$$\mu_i = \frac{1}{MTTR_i} \forall i \in N \tag{6}$$

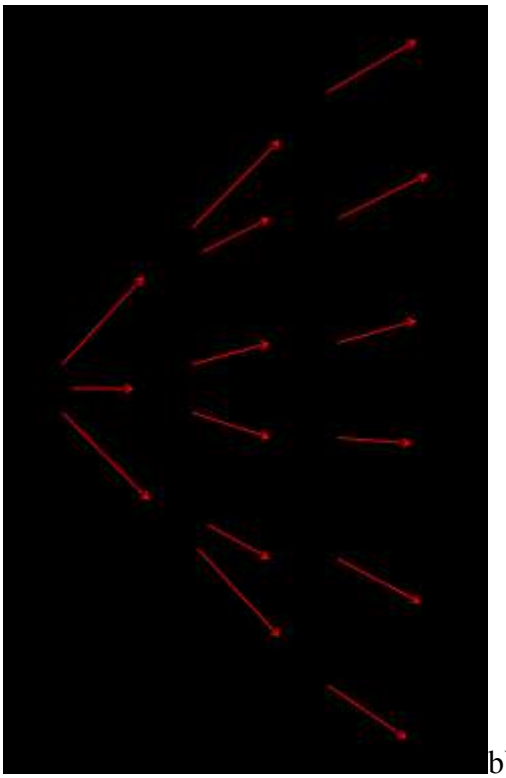
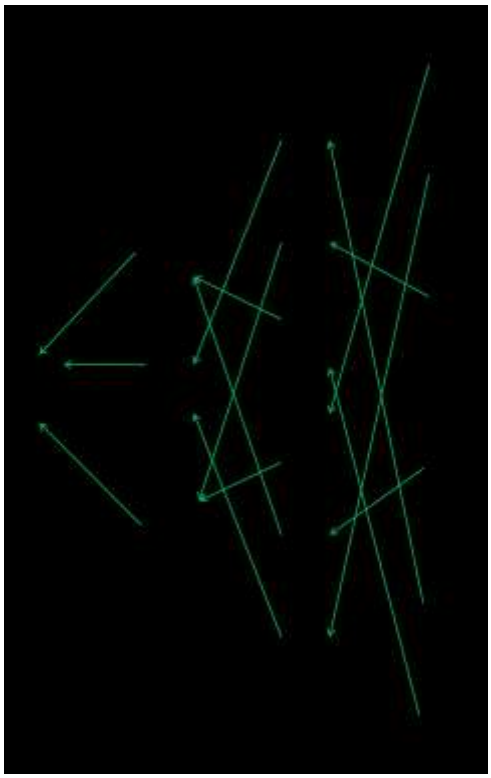
a)  b) 

Figure 8 – Markov chain for three paths with a) transitions due to failures and b) transitions due to repairs

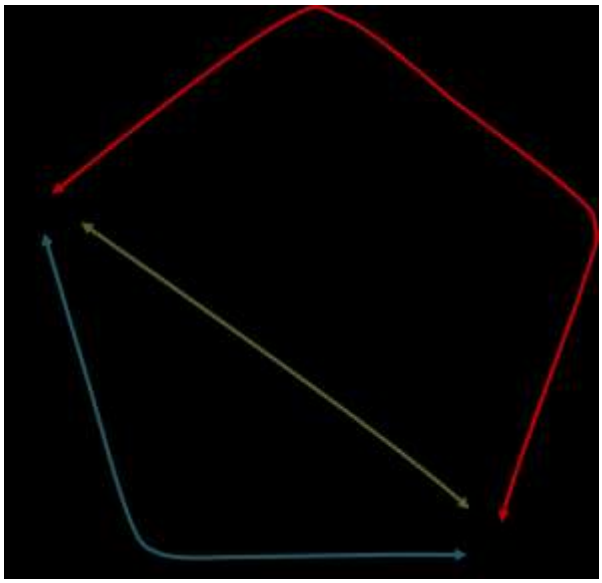


Figure 9 – 3 Paths between origin and destination on the sample graph

The state diagram in Figure 8 enumerates the number of states, as well as describes how the various failure states in a network interact. It is assumed that failures are repaired in the order they occur, and that they do not occur simultaneously. In order to determine the steady-state probabilities of being in each state a couple of equations (7) and (8), derived from the Chapman-Kolmogorov equation, are used [28], assuming a time invariant transition matrix. The Chapman-Kolmogorov equation is used to define the availability in a Markov chain. These equations are a set of linear equations that utilize the transition rates between states, described in an *n*x*n* matrix, **Q**, where *n* is the number of states, which are used to determine the steady-state probabilities of being in each state. Entry $Q_{i,j}$ when $i \neq j$, represents the transition rate from state *i*, to state *j*, and comes from (9), where $\boldsymbol{P}(t)$ is the matrix of probabilities of states transitions over time *t*. In the context of network survivability, these are the rates of failure and repair of the various paths. $Q_{i,i}$ is defined in (10), which is derived from (9) assuming constant (in time) transition rates. $-Q_{i,i}$ represents the total departure rate from state i. π is row matrix that represents the steady-state probabilities of being in a given state. (7) comes from the idea that over a long period of time the probability of going from state *i* to state *j* is going to converge to a single value, so long as the Markov chain is homogeneous and irreducible, which is the case with the network failure models. (8) simply says that $\boldsymbol{\pi}$ is a legitimate probability distribution. π_i is the *i*th indexed value in the $\boldsymbol{\pi}$ vector.

$$\boldsymbol{\pi Q} = 0 \tag{7}$$

$$\sum_i \pi_i = 1 \tag{8}$$

$$\frac{\partial \boldsymbol{P}(t)}{\partial t} = \boldsymbol{Q} \cdot \boldsymbol{P}(t) \tag{9}$$

$$q_{i,i} = -\sum_{i \neq j} q_{i,j} \tag{10}$$

Finding $\boldsymbol{\pi}$ allows the calculation of the availability of the demand by using (11), where $\boldsymbol{R_d}$ is the fraction of traffic for a specific demand that is able to be routed in each state.

$$A_d = \boldsymbol{\pi} \times \boldsymbol{R_d} \tag{11}$$

Since solving (7) and (8) can be quite difficult when the number of possible failure states is large, most availability literature simplifies these calculations [8][20][25][26].

Generally it is unavailability that is calculated when evaluating availability in a network. The most common simplification has been to consider only the impact of dual failures on availability when dealing with survivable network design. This is a reasonable assumption, since when designing for single failure restorable networks, it is only when more than one component fails that the network sees a significant degradation in service. Since MTTR << MTTF, three or more failures have a small impact on availability relative to dual failures on single failure restorable networks, and considering the error in estimating failure and repair rates, are not relevant to the results. From [25], (12) provides the basis for calculating availability using the above assumptions. S^d is the set of all dual span failures (spans *a* and *b*) that affects demand *d*. $P_{a,b}$ is the probability of span *a* failing and span *b* failing before span *a* has been repaired, and $R_2^{\mathrm{d}}(a,b)$ is the percentage of the total demand that can be restored for demand *d* when spans *a* and *b* have failed. It should be noted that in general the failure order matters, as $R_2^{\mathrm{d}}(a,b)$ does not always equal $R_2^{\mathrm{d}}(b,a)$, especially for restoration mechanisms that share spare capacity between restoration routes that protect failures of more than one demand.

$$A_d = 1 - \sum_{\forall \mathrm{a,b} \in S^d} P_{a,b} * R_2^d(a,b) \tag{12}$$

$R_2^{\mathrm{d}}(a,b)$ can be calculated easily through failure scenario simulations. The probability of *a* and *b* failing has generally been calculated by multiplying the unavailabilities of *a* and *b* (14). This is taken as an estimation of (13), and is a slight underestimation of unavailability (U_i). The degree of error in this simplification is not significant, and does not affect the probability of dual failure

by more than a percentage point for the scale of MTTFs and MTTRs found in survivable network design.

$$P_{a,b} = p(failure\ time(t_b) < MTTR_a) \times p\big(failure(a)\big)$$
$$= U_a \times \left(1 - e^{\frac{MTTR_a}{MTTF_b}}\right) \quad (13)$$

$$P_{a,b} \approx U_a \times U_b \quad (14)$$

The following is a small example to demonstrate the error introduced by the various assumptions made in calculating availability. Using Figure 10 as a small example, 5 units of capacity were routed on the three paths using DSP. Path 1 and 2 had three units of capacity and path 3 had 2 units. The MTTR was assumed to be 12 hours, and the MTTF was 292000 divided by the path length (this number represents 3 failures per year per 1000km of cable). Availability was calculated by using the Markov chain, by path failure analysis and span failure analysis, assuming only dual failures. Both the path and span analysis were done by multiplying unavailabilities (14) and by using (13). The results are in Table 1, and although the estimation methods under-represent unavailability by 15.5% to 17.5%, this translates to a difference of 9.2 seconds of outage per year. When considering the relatively high availabilities of communication networks, and the errors associated with the estimation of the MTTR and MTTF, it is reasonable to use the simplified methods of calculating availability.

Calculation Type	Unavailability	% Difference
Path analysis (UxU)	1.20022E-05	-17.52%
Path analysis (using probability)	1.20164E-05	-17.42%
Span analysis (UxU)	1.20403E-05	-17.26%
Span analysis (using probability)	1.22960E-05	-15.50%
Markov	1.45512E-05	0.00%

Table 1 – Sample availability calculations using various calculation methods

All availability calculations in this work utilize an expanded version of (15), where $R_2(a, b)$ is defined in (16).

$$A_d = 1 - \sum_{\forall a,b \in S^d} U_a * U_b * R_2(a,b) \quad (15)$$

$$R_2(a,b) = \frac{Units\ of\ restorable\ capacity}{Units\ of\ affected\ capacity} \quad (16)$$

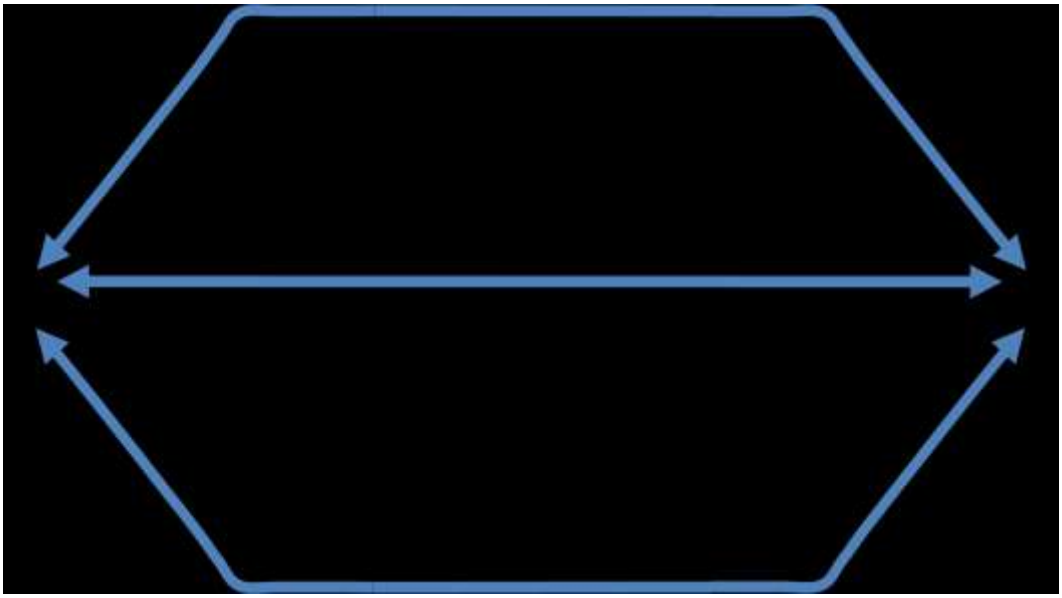


Figure 10 – Network graph with span costs (in brackets) and path routing

There are other methods of calculating availability [20][27]. In [20], a formula for the availability of a connection is presented for an N:M(m) APS (which is extremely similar to DSP). This formula is based on the availabilities on M working, and N failed paths, and iterates for each possible failure scenario. This formula is generally accurate; however there is no accounting for partial restorability. Another method that is presented in [27], which is called restoration aware connection availability (RACA), provides a formulation to estimate the unavailability of each span based on the effects of being able to restore some or all of the traffic on the failed span.

Chapter 3. Algorithms and Integer Linear Programs

The focus of the research done in this work was to explore the use of DSP in high availability networks, as well as the characteristics of networks whose topology has been designed to efficiently implement a DSP-based capacity routing for a given level of restorability or availability. Examining the characteristics of DSP networks with high availability provides a basis to evaluate the potential of utilizing DSP designs to meet the increasing availability demands that are being put on communication networks.

We have created a number of models to explore the use of DSP-type survivability in high availability networks. All the network design results are compared to transshipment-based DSP model (DSP-TR) that ensures single failure restorability. DSP-TR models were created to design networks for given levels of availability (DSP-TR-A), as well as dual failure restorability (DSP-TR-R2). Three models were also created that also optimize topology for no high availability requirements (DSP-Top), per demand availability requirements (DSP-Top-A), and per demand dual failure restorability requirements (DSP-Top-R2). Each of these models are presented along with their *integer linear programming* (ILP) representation.

The work done with designing DSP networks used a number of conventions. First it is assumed that all traffic demands are to be 100% single-failure restorable. The original DSP proposal used differentiated service levels to identify a portion of the demand that is restorable, in light of a single failure, and the rest of the demand not restorable. It is not difficult to route demands on a network that do not require restoration, and can be done so after the network has

been designed for the restorable traffic. Secondly, it is assumed that there are no modularity requirements, as discussed in section 4.2 . Also, implied in DSP is that both working and spare capacity will be jointly routed (using the JCA model). Unless the model explicitly allocates what nodes have connections between them (topology design), it is assumed that the topology of the network is fixed; however, there are no capacity limits on any spans.

3.1 Demand-wise Shared Protection using a Transportation Problem Approach

The DSP-TR model optimizes the routing of traffic in a network using the *transportation flow problem* as its basis. The transportation flow problem [30] is a generic problem that optimally routes the transportation of a commodity throughout a network. By utilizing the transportation flow model, the requirement for predefined paths is eliminated. It is common in other path-based survivability mechanisms to decide on the routing of traffic based on a set of predefined paths.

There were three factors that discouraged the use of predefined paths for designing DSP networks. The first is that it is impossible to predefine a path set when the problem also involves topology design as the possible paths available are dynamic as the topology changes. Also, unless the set of paths is exhaustive, there is a possibility that a path that was not included in the path set provided to the ILP could provide a better solution. The third factor that discouraged the use of predefined paths is in the computation time required by an ILP solver. The transportation flow models would solve significantly faster than their path-based counterparts. As such, the DSP survivability models all utilized a transportation flow approach.

Using the transportation flow problem as the basis for the ILP, the model added constraints to ensure that the paths selected were node disjoint, and had adequate capacity to survive a single failure. The topology was defined by the set of nodes

in the network, N, and an indexed set which defined what nodes were connected to one another, A_n. Spans were not represented as a set explicitly, and directionality was required to ensure correct path allocation. D is a set of all the node pairs in the network representing all the possible demands for connectivity in the network. d_r is the volume of traffic that is required to be routed by demand *r*.

This model does not differentiate between working and spare capacity, which is simple to do after the results are analyzed. It does ensure each demand is protected against single failures. When representing spans, the model uses a directed connection from one node to another. This is essential to transportation problems, otherwise, with spans being represented without a direction, it cannot be enforced (directly) that a path originates with the origin and terminates at the destination. With a directionless representation there could be two paths (or more), one that originates and terminates at the origin node, and the other path originating and terminating at the destination node, not actually providing connectivity between the origin and destination nodes. From this directed connection, it is implicitly assumed that demand is symmetric and there is a reciprocal path flowing from the destination to the origin. It should be noted that all spans within the network topology are assumed to be bidirectional.

Sets:

N is the set of all nodes in the network

D is the set of all node pairs with traffic demands between them

$A_n \subseteq N$ is a subset of N, and represents all nodes that are connected to node *n* by a single span

Parameters:

c_i is the cost of span *i*

O_r is the origin node for demand *r*

T_r is the destination node for demand *r*

d_r is the number of units of traffic required by demand *r*

M is a number larger than any possible $\omega_{i,j}^{f}$ value

Variables:

$\omega_{i,j}^{r} \geq 0$ is the traffic flow from node *i* to node *j* where $j \in A\_i$

$f_{i,j}^{r} \in \{0,1\}$ is the 1/0 variable indicating whether capacity is allocated from node *i* to node *j* where $j \in A_i$

Minimize:

$$\sum_{r \in D} \sum_{i \in N} \sum_{j \in A_i} \omega_{i,j}^{r} \times c_{i,j} \tag{17}$$

Subject to:

$$\sum_{k \in A_i : k \neq j} \omega_{i,k}^{r} \geq d_r \qquad \forall r \in D, i \in O_r, j \in A_i \tag{18}$$

$$M \times f_{i,j}^{r} \geq \omega_{i,j}^{f} \qquad \forall r \in D, i \in N, j \in A_i \tag{19}$$

$$f_{i,j}^{r} + f_{j,i}^{r} \leq 1 \qquad \forall r \in D, i \in N, j \in A_i \tag{20}$$

$$\sum_{j \in A_i} \omega_{i,j}^{r} - \sum_{j \in A_i} \omega_{j,i}^{r} = 0 \qquad \forall r \in D, i \in N | i \notin \{O_r, T_r\} \tag{21}$$

$$\sum_{j \in A_i} f_{i,j}^{r} \leq 1 \qquad \forall r \in D, i \in N | i \notin \{O_r, T_r\} \tag{22}$$

$$\sum_{j \in A_i} f_{j,i}^{r} \leq 1 \qquad \forall r \in D, i \in N | i \notin \{O_r, T_r\} \tag{23}$$

The objective of the DSP-TR ILP (17) is to minimize the cost of the allocation of capacity throughout the network. (18) ensures there is enough capacity assigned to route all of the demand traffic if any one path fails. While the model does not explicitly define working routes and failure routes, these are simple to define

using the results, as is common in transportation problems. (19) through (23) are used to create the required paths using a modified transportation problem approach. Because traffic cannot split at transiting nodes, (19) is used to relate the flow assigned to each span to the 1/0 variable that represents whether or not any capacity has been assigned to a given span in the direction from *i* to *j*. While adding a 1/0 variable to the problem increases run time, it is necessary to ensure the paths chosen are disjoint. (20) ensures that traffic on any span will only flow in one direction. This unidirectional property ensures all paths are simple in nature. The network designs already implicitly assume symmetric traffic patterns, and do not explicitly route traffic back from the destination to the origin nodes. If traffic is in fact asymmetrical this can be accounted for by increasing the set of demands to include entries for both the origin to destination, and destination to origin nodes. **(**21**)** ensures flow conservation, allowing traffic to only enter or exit a network at the origin and destination, while (22) and (23) ensure paths are node disjoint.

3.2 Demand-wise Shared Protection with Topology Allocation

The DSP-TR model was presented in the transportation format to be able to add topology design to the problem without significantly altering the formulation of the problem. Adding the ability for the ILP model to select the best set of paths to route the traffic greatly changes the results of the model; however, the mechanism to select paths using the transportation problem remains the same between the DSP-TR and the DSP with topology design (DSP-Top).

The purpose behind allowing the ILP to optimize the topology provides insight into what kind of network design structures are ideal for adopting DSP, as well as providing the ability to efficiently design green-field networks. There has been other work done that utilized an iterative approach to add and remove spans from a network, therefore doing some topology design. By incorporating the topology design into the ILP directly, there is more confidence in the optimality of the design than a growing and pruning approach [28].

DSP is most effective when there are multiple disjoint paths that are relatively similar in cost to the shortest path between the origin and destination nodes. In order to best capitalize on the capacity sharing capabilities of DSP, the topology designs would be able to arrange these multiple path characteristics around the demands that have the highest capacity, thereby reducing the overall cost effectively.

In order to add topology design to the DSP-TR model, the 1/0 variable, $f_{i,j}$, was added which represents whether or not the span from node i to j or j to i has capacity assigned to it. This directionless property is needed so that the cost to implement the span is accounted if a span is utilized in either direction. To do this, the parameter which encodes the cost of instantiating a span, $b_{i,j}$, must encode a cost of zero for one direction, and the implementation cost for the other.

Sets:

N is the set of all nodes in the network

D is the set of all node pairs with traffic demands between them

$A_n \subseteq N$ is a subset of N, and represents all nodes that are connected to node *n* by a single hop

Parameters:

$c_{i,j}$ is the cost of the span connecting nodes *i* and j

$b_{i,j}$ is the cost of instantiating a span connected by nodes *i* and *j*

O_r is the origin node for demand *r*

T_r is the destination node for demand *r*

d_r is the number of units of traffic required by demand *r*

Variables:

$\omega_{i,j}^{r} \geq 0$ is the traffic flow from node *i* to node *j* where $j \in A\_i$

$f_{i,j}^{r} \in \{0,1\}$ is the 1/0 variable indicating whether capacity is allocated from node *i* to node *j* where $j \in A_i$ for demand *r*

$f_{i,j} \in \{0,1\}$ is the variable indicating whether any capacity is allocated from node *i* to node *j* or from node *j* to *i*, where $j \in A_i$

Minimize:

$$\sum_{r} \sum_{i \in N} \sum_{j \in A_i} \omega_{i,j}^{r} \times c_{i,j} + f_{i,j} \times b_{i,j} \tag{24}$$

Subject to:

$$\sum_{k \in A_i : k \neq j} \omega_{i,k}^{r} \geq d_r \qquad \forall r \in D, i \in O_d, j \in A_i \tag{25}$$

$$M \times f_{i,j} \geq \sum_{r \in D} [f_{i,j}^{r} + f_{j,i}^{r}] \qquad \forall i \in N, \forall j \in A_i \tag{26}$$

$$M \times f_{i,j}^{r} \geq \omega_{i,j}^{f} \qquad \forall r \in D, i \in N, j \in A_i \tag{27}$$

$$f_{i,j}^{r} + f_{j,i}^{r} \leq 1 \qquad \forall r \in D, i \in N, j \in A_i \tag{28}$$

$$\sum_{j \in A_i} \omega_{i,j}^{r} - \sum_{j \in A_i} \omega_{j,i}^{r} = 0 \qquad \forall r \in D, i \in N \tag{29}$$

$$\sum_{j \in A_i} f_{i,j}^{r} \leq 1 \qquad \forall r \in D, i \in N \,|\, i \neq \{O_r, T_r\} \tag{30}$$

$$\sum_{j \in A_i} f_{j,i}^{r} \leq 1 \qquad \forall r \in D, i \in N | i \neq \{O_r, T_r\} \tag{31}$$

In actual changes to the model, there are very few. The objective function, **(**24**)**, takes into account the cost of implementing each span, as well as the capacity costs. The constraint **(**26**)** sets $f_{i,j}$ to 1 if capacity is assigned to the span connecting nodes *i* and *j* for any demand. The remaining constraints remain the same as the DSP-TR model.

The changes to the ILP, although minimal, greatly affect the results. When the topology of the network is flexible, a greater degree of freedom is granted that

encourages the ILP to better allocate resources. The resulting network designs can be significantly different than those that do not include topology design.

3.3 Demand-wise Shared Protection with Availability Requirements Using a Transportation Problem Approach

The calculation of availability is essentially a non-linear calculation when working with a transportation problem. In a parallel system, availability is estimated as the product of the unavailabilities of each component of the system. In order to estimate the availability of a network design, each dual failure scenario was added, and the routable traffic for each demand calculated in order to implement (15). Using an approach that pre-calculates possible paths, the unavailability of each path may be pre-calculated, and the dual failure restorability (R2) of each demand is the only variable in the availability calculation. This approach however is not feasible with variable topologies, and was not utilized. When incorporating topology design into the model, paths cannot be effectively pre-calculated, and therefore their availability is unknown when preparing the data for the ILP to solve. In transportation problems, which are a necessary part of topology design, the availability of each path must be dynamically calculated, and the availability calculations become non-linear.

In order to get around the non-linearity of the availability calculations, the calculations were done through evaluating the routing for each failure scenario from within the ILP. From this evaluation, the impact of each failure scenario could then be measured on the individual demands. When the percentage of routable demand is combined with the probability of each failure scenario, the contributions of the specific failure scenarios to the overall unavailability can be a summed up to provide an estimated availability.

Sets:

N is the set of all nodes in the network

D is the set of all node pairs with traffic demands between them

$A_n \subseteq N$ is a subset of N, and represents all nodes that are connected to node n by a single hop

Parameters:

$c_{i,j}$ is the cost of the span connecting nodes i and j

O_r is the origin node for demand r

T_r is the destination node for demand r

d_r is the number of units of traffic required by demand r

v_r is the required availability for demand r

$MTTF_i$ is the mean time to failure for span i

MTTR is the mean time to repair, assumed to be constant across all spans

Variables:

$\omega_{i,j}^r \geq 0$ is the traffic flow from node i to node j where $j \in A\_i$

$f_{i,j}^r \in \{0,1\}$ is the 1/0 variable indicating whether capacity is allocated from node i to node j where $j \in A_i$

$\rho_{i,j,k,l}^r$ is the amount of traffic that cannot be routed for demand r if the connection between nodes i,j and k,l fail, where $j \in A_i$ and $l \in A_k$

$\psi_{i,j,k,l,m,n}^r$ represents the traffic assigned to the span connecting nodes m and n if if the connection between nodes i,j and k,l fail, where $j \in A_i$, $l \in A_k$ and $n \in A_m$.

Optimize

$$\sum_r \sum_{i \in N} \sum_{j \in A_i} \omega_{i,j}^r \times c_{i,j} \tag{32}$$

Subject to

$$\sum_{k\in A_i:k\neq j} \omega_{i,k}^r \geq d_r \qquad \forall r \in D, i \in O_d, j \in A_i \qquad (33)$$

$$M \times f_{i,j}^r \geq \omega_{i,j}^r \qquad \forall r \in D, i \in N, j \in A_i \qquad (34)$$

$$f_{i,j}^r + f_{j,i}^r \leq 1 \qquad \forall r \in D, i \in N, j \in A_i \qquad (35)$$

$$\sum_{j\in A_i} \omega_{i,j}^r - \sum_{j\in A_i} \omega_{j,i}^r = 0 \qquad \forall r \in D, i \in N \qquad (36)$$

$$\sum_{j\in A_i} f_{i,j}^r \leq 1 \qquad \forall r \in D, i \in N \qquad (37)$$

$$\sum_{j\in A_i} f_{j,i}^r \leq 1 \qquad \forall r \in D, i \in N \qquad (38)$$

$$\sum_{i,k\in N, j\in A_i, l\in A_k} \frac{MTTR}{MTTF_{i,j} + MTTR} \times \frac{MTTR}{MTTF_{k,l} + MTTR} \times \frac{\rho_{i,j,k,l}^r}{d_r} \leq v_r \qquad \forall r \in D \qquad (39)$$

$$\rho_{i,j,k,l}^r \geq d_r - \sum_{n\in A_m} \psi_{i,j,k,l,m,n}^r \qquad \forall r \in D, (i,k) \in N, j \in A_i, l \in A_k, m \in O_r \qquad (40)$$

$$\psi_{i,j,k,l,j,j}^r = 0 \qquad \forall r \in D, (i,k) \in N, j \in A_i, l \in A_k \qquad (41)$$

$$\psi_{i,j,k,l,k,l}^r = 0 \qquad \forall r \in D, (i,k) \in N, j \in A_i, l \in A_k \qquad (42)$$

$$\psi_{i,j,k,l,m,n}^r \leq \omega_{m,n}^r \qquad \forall r \in D, (i,k,m) \in N, j \in A_i, l \in A_k, n \in A_m \qquad (43)$$

$$\psi_{i,j,k,l,m,n}^r - \sum_{o\in A_m: o\neq n} \psi_{i,j,k,l,m,o}^r \leq 0 \qquad \forall r \in D, (i,k,m) \in N, j \in A_i, l \in A_k, n \in A_m \qquad (44)$$

In order to develop a model that would calculate availability dynamically, the restorability for each failure combination must be calculated. The ILP builds on the DSP-TR model, adding constraints that calculate dual failure restorability and subsequent availability for each possible dual failure. This could be thought of as a number of sub-problems that indicate if the current solution provides adequate availability.

This was done by adding the variable $\psi^{r}_{i,j,k,l,m,n}$ which represents the volume of flow on the span connecting *m,n* when the spans connecting *i,j* and *k,l* have failed. This works with the directional span representation as when traffic flow for a given demand is from *i* to *j*, the dual failure restorability when *j* to *i* fails is 1, and hence the failure of the specific span is counted only once. Because the total potential flow of traffic between the origin and destination could be greater than the demand, **(**40**)** prevents the number of failed spans from going below zero. **(**41**)** and **(**42**)** state that traffic cannot be assigned to failed spans. **(**43**)** enforces the surviving traffic to only be assigned to paths that have capacity allocated in the non-failure scenario. **(**44**)** allows traffic conservation on non-end nodes. This formula works on its own because path disjointedness has been enforced in **(**36**)**.

The model presented assumes that three or more concurrent failures will occur with a low enough probability as to not affect availability, which will not be the case for networks that require extremely high availability. However, the concept used to calculate the dual failure contribution to the actual unavailability could also be used to calculate the contribution of three or more concurrent failures. This would, however, have a significantly detrimental impact on the computational complexity of the model.

The purpose of the DSP-TR-A survivability model is to design networks with a straight forward correlation between the designs and the required availabilities of each demand. This however requires a complex ILP that evaluates each failure scenario. The ILP presented calculated availability based on two or fewer concurrent failures. The model can be extended for failure scenarios of more than

two concurrent failures; however, this comes at a significant computing complexity costs.

3.4 Demand-wise Shared Protection with Availability Requirements and Topology Design

The DSPT-TR-A survivability model was based on the transportation type problem for routing traffic and, as with the DSP-TR model, can be modified to add green-field topology design. The topology design of availability constrained DSP networks takes into consideration the tradeoff between shorter routing and multiple paths, balancing exposure to failure and failure protection.

In order to optimize the instantiation of spans in a network, along with capacity allocation, the DSP-TR-A ILP was modified in a similar manner to the way the DSP-TR model was modified to implement topology design, with **(**45**)** replacing the cost function (32) and adding constraint **(**46**)**.

$$\sum_{r}\sum_{i\in N}\sum_{j\in A_i} \omega_{i,j}^{r} \times c_{i,j} + f_{i,j} \times x_{i,j} \tag{45}$$

$$M \times f_{i,j} \geq \sum_{r\in D} f_{i,j}^{r} + f_{j,i}^{r} \qquad \forall i \in N, \forall j \in A_i \tag{46}$$

3.5 Demand-wise Shared Protection with Guaranteed Minimum Dual Failure Restorability using the Transportation Problem Approach

One approach to increasing the availability and overall uptime of a given demand or network is to make the routing scheme explicitly more resilient to failures. The DSP with guaranteed minimum dual failure restorability using the transportation problem approach (DSP-TR-R2) survivability model allows a network to be designed by specifying a certain minimum level of dual failure restorability for

each demand. By doing so, a predetermined fraction of traffic for each demand is guaranteed to be routable for any single or dual failure that may occur in a network.

The advantages of using this paradigm versus directly solving for availability as in the DSP-TR-A are a more direct correlation between failure scenarios and restoration procedures and an ILP structure that solves with less computing resources. The DSP-TR-R2 is more scalable in that the number of constraints grows in the O($|N|$) scale, while the DSP-TR-A model has constraints in the scale of O($|N^3|$). Calculating the number of units of capacity restorable under a given failure scenario is linear, and hence the linearization strategies employed in the DSP-TR-A model are not needed.

Sets:

N is the set of all nodes in the network

D is the set of all node pairs with traffic demands between them

$A_n \subseteq N$ is a subset of N, and represents all nodes that are connected to node *n* by a single hop

Parameters:

$c_{i,j}$ is the cost of span *i*

O_r is the origin node for demand *r*

T_r is the destination node for demand *r*

d_r is the number of units of traffic required by demand *r*

R_2^r is the required fraction of traffic to survive any dual failure

Variables:

$\omega_{i,j}^r \geq 0$ is the traffic flow from node *i* to node *j* where $j \in A\_i$

$f_{i,j}^r \in \{0,1\}$ is the 1/0 variable indicating whether capacity is allocated from node i to node j where $j \in A_i$

Minimize:

$$\sum_r \sum_{i \in N} \sum_{j \in A_i} \omega_{i,j}^r \times c_{i,j} \quad \textbf{(47)}$$

Subject to

$$\sum_{k \in A_i : k \neq j} \omega_{i,k}^r \geq d_r \qquad \forall r \in D, i \in O_d, j \in A_i \quad \textbf{(48)}$$

$$M \times f_{i,j}^r \geq \omega_{i,j}^f \qquad \forall r \in D, i \in N, j \in A_i \quad \textbf{(49)}$$

$$f_{i,j}^r + f_{i,j}^r \leq 1 \qquad \forall r \in D, i \in N, j \in A_i \quad \textbf{(50)}$$

$$\sum_{j \in A_i} \omega_{i,j}^r - \sum_{j \in A_i} \omega_{j,i}^r = 0 \qquad \forall r \in D, i \in N \quad \textbf{(51)}$$

$$\sum_{j \in A_i} f_{i,j}^r \leq 1 \qquad \forall r \in D, i \in N \quad \textbf{(52)}$$

$$\sum_{j \in A_i} f_{j,i}^r \leq 1 \qquad \forall r \in D, i \in N \quad \textbf{(53)}$$

$$\sum_{(l \in A_i | l \neq j, l \neq k)} \frac{\omega_{i,l}}{d_r} \geq R_2^r \qquad \forall r \in D, i \in O_d, j \in A_i \quad \textbf{(54)}$$

Enforcing a minimum level of dual failure restorability was done in a similar manner to enforcing single failure restorability. (54) is added to the DSP-TR model and says that if there is a failure on any two paths, there must be enough routable capacity to route the specified fraction of the total traffic for a demand on the remaining paths.

3.6 Demand-wise Share Protection with Guaranteed Minimum Dual Failure Restorability and Topology Allocation

Efficient dual failure restorability is predicated by having a network topology that allows for three or more paths for each demand. Incorporating topology design

into dual failure restorable DSP provides a basis for these efficient topologies, customized to provide capacity savings where it is most beneficial.

The approach taken to design the topology of the network was to allow the ILP to choose from a pool of possible spans, each with a given cost to implement it. This was similar to what was utilized in DSP-Top. In order to have a flexible topology, a transportation problem approach was taken that incorporates path discovery into the ILP.

By incorporating path discovery into the ILP, the flexibility to change the topology within the ILP is added. Inherent in the transportation problem approach is the lack of explicitly defined paths within the solution. The routing of each path can be later found by observing the capacity allocation for each demand.

With the paths not explicitly defined an alternate method for ensuring restorability the same method that was utilized in DSP-TR-R2. So long as each path was disjoint, the number and capacity of paths assigned to each demand is represented by the spans adjacent to the origin or destination node. Using this property, dual failure restorability could be set while also optimizing topology.

The dual failure survivability and topology design (DSP-Top-R2) model contains a structure that is similar to the other ILPs that have been presented, incorporating features of dual failure and topology design.

Sets:

N is the set of all nodes in the network

D is the set of all node pairs with traffic demands between them

$A_n \subseteq N$ is a subset of N, and represents all nodes that are connected to node *n* by a single hop

Parameters:

$c_{i,j}$ is the cost of span *i*

$x_{i,j}$ is the cost of implementing span connected by nodes *i,j*

O_r is the origin node for demand *r*

T_r is the destination node for demand *r*

d_r is the number of units of traffic required by demand *r*

R_2^r is the required fraction of traffic to survive any dual failure

Variables:

$\omega_{i,j}^r \geq 0$ is the traffic flow from node *i* to node *j* where $j \in A\_i$

$f_{i,j}^r \in \{0,1\}$ is the 1/0 variable indicating whether capacity is allocated from node *i* to node *j* where $j \in A_i$

$f_{i,j} \in \{0,1\}$ is the 1/0 variable indicating whether any capacity is allocated from node *i* to node *j* or from node *j* to *i*, where $j \in A_i$

Minimize

$$\sum_r \sum_{i \in N} \sum_{j \in A_i} \omega_{i,j}^r \times c_{i,j} + f_{i,j} \times x_{i,j} \quad \textbf{(55)}$$

Subject to

$$\sum_{k \in A_i : k \neq j} \omega_{i,k}^r \geq d_r \qquad \forall r \in D, i \in O_d, j \in A_i \quad \textbf{(56)}$$

$$M \times f_{i,j}^r \geq \omega_{i,j}^f \qquad \forall r \in D, i \in N, j \in A_i \quad \textbf{(57)}$$

$$f_{i,j}^r + f_{i,j}^r \leq 1 \qquad \forall r \in D, i \in N, j \in A_i \quad \textbf{(58)}$$

$$\sum_{j \in A_i} \omega_{i,j}^r - \sum_{j \in A_i} \omega_{j,i}^r = 0 \qquad \forall r \in D, i \in N \quad \textbf{(59)}$$

$$\sum_{j \in A_i} f_{i,j}^r \leq 1 \qquad \forall r \in D, i \in N \quad \textbf{(60)}$$

$$\sum_{j \in A_i} f_{j,i}^r \leq 1 \qquad \forall r \in D, i \in N \qquad \textbf{(61)}$$

$$\sum_{(l \in A_i | l \neq j, l \neq k)} \frac{\omega_{i,l}}{d_r} \geq R_2^r \qquad \forall r \in D, i \in O_r, j \in A_i \qquad \textbf{(62)}$$

$$M \times f_{i,j} \geq \sum_{r \in D} f_{i,j}^r + f_{j,i}^r \qquad \forall i \in N, j \in A_i \qquad \textbf{(63)}$$

The objective function **(**55**)** concurrently optimizes topology implementation costs, as well as capacity placement. The constraints are combined from the DSP-Top and DSP-TR-R2 ILP's. (56) through (61) provide the basis for DSP network design. (62) enforces the dual failure restorability for each network. (63) sets $f_{i,j}$ to 1 if there is any traffic routed in either direction on the span connected by nodes *i* and *j* in order to enable topology optimization.

Chapter 4. Computational Setup

Some implementations of survivability schemes attempt to route traffic in a pre-capacitated network. These can have goals, such as to route as much capacity as possible for a given failure survivability level, or to route traffic such that the ratio of each spans utilized to total capacity is minimized [27]. It is common in literature to have the goal of the survivability model to route traffic in a manner that provides a given level of failure survivability in the most capacity efficient way possible, and that is the paradigm used in testing and simulating the DSP models in this work. The primary metric that was used to evaluate a network survivability model was the redundancy (or capacity cost) of the network. Redundancy is usually measured as the ratio of the increase of capacity required by the final design over the capacity required route all demands using a single shortest path. Since the cost to route all demands using the shortest paths is a constant, regardless of the survivability scheme, redundancy is a measure of capacity cost normalized by the size and amount of traffic in a network.

There are a variety of network characteristics that affect redundancy. Five of these characteristics are: the demand matrix (i.e., the number of lightpath demands between each pair of nodes), the number of nodes in the network, the configuration and distribution of nodes in the network, the number of spans in the network, and the configuration of spans in the network. Some of these characteristic's effects are obvious, such as the number of spans in the network. Some however can have very subtle, yet significant effects on the network's total cost.

The demand matrix is one of those factors that have both obvious and subtle impacts on network performance. The total volume of traffic the network must

route will clearly affect the capacity of a network; however, there are other characteristics of the demand matrix that can have a significant impact on the capacity allocated in a network. Especially important for survivability models that share capacity between different demands and failure scenarios is the relative distribution of the volume of traffic between demands. If there is a large variation in the volume of the demands, then there is the potential that the forcer gap (the greatest capacity required by a failure scenario minus the second greatest failure scenario) is quite high [19], and could require significantly more spare capacity than the same network with a flat distribution of demand traffic. Another feature of the demand matrix that can significantly impact network redundancy is the locality of traffic on the network. When voice traffic dominated the network, most traffic generated by a node would terminate at another node that was in the same geographical region. As data traffic has grown in prevalence on communication networks, this locality of traffic is no longer a significant characteristic of traffic patterns. It is therefore important to understand the effect of the demand matrix on network redundancy when comparing the redundancy performance of a survivability model.

The topology of a network can also affect a network's redundancy. To list a few topology features that can affect redundancy, the ratio of spans to nodes, the amount of clustering in node location and the distribution of nodal connectivity all are significant. The ratio of spans to nodes is commonly presented as the average nodal degree of a network, where nodal degree is the number of spans connected to a node. With relatively more spans in a network, there are more options to route traffic over more physically direct paths, as well as more diversity in paths to be able to more effectively share spare capacity. Besides the ratio of spans to nodes, the distribution of these spans is also significant. If a couple of centralized nodes that are highly connected and the remaining nodes are connected to the network with two, maybe three spans, route diversity would be significantly reduced, when compared to a more even distribution in the nodal degree. In general the topology of a network affects path diversity and distance, and consequently affects the capacity required in the network.

Each network is an abstraction of a possible real network. Generally the networks are thought to represent networks of a national or continental scale. The nodes represent cities, or central routing hubs, with the spans representing the cabling connecting these hubs. The networks used in this study were on the smaller side of the scale when looking at number of nodes in the network, as compared to actual networks. This limitation was due to the computing resources required by the ILP solvers.

4.1 Selection of test networks

In order to study new network design models, a set of test networks are required. Some common approaches in selecting relevant test networks are to use abstracts of actual networks or to create a number of networks whose topology is randomly generated. As discussed in [31], these methods may prove to be ineffective or computationally expensive in providing a generic evaluation of the survivability model's performance over a variety of topologies. Another method of selecting test networks is to use network families. A network family is a set of networks that contain a coherent underlying topology, and differ by the network's nodal degree.

The advantage of using network families is in how it balances the benefits and impracticalities of the other two methods of selecting test networks. By using actual networks, results are relevant to working network designs; however, it is difficult to evaluate the performance of the survivability model across a spectrum of topologies. An actual network may have a certain topology that greatly enhances or reduces a model's performance; however, this is difficult to ascertain. The use of random networks across a variety of nodal and span configurations can provide a more generic result; this, however, often proves impractical as solving a single network for many survivability models can take days of computing time. By using network families, survivability models are evaluated across a spectrum of nodal degrees, while keeping the topologies and demand matrices closely related and minimizing the amount of computing resources required. [31]

demonstrates how closely network families were to the average of many random networks, and therefore had little bias in their results.

A network family is created from an initial network with the maximum nodal degree which is in the realm of possibility in actual networks. From this initial network, spans are removed one at a time in a random manner, with each removal creating a new network. This is repeated ensuring the network remains bi-connected, until removing any span will break the bi-connected property of the network. The resulting sets of networks have identical nodal topologies, with span layouts that are sequentially related to each other. The communication demand between each node also remains constant for the set of networks.

The networks used for the implementation of the DSP models presented earlier fall into two classes. For the topology constrained models, network families of a nodal size of 8, 10, 12 and 15 were used. The master networks had an average nodal degree of 4, and each family went down to an average nodal degree of just over 2. These families are presented completely in the Appendices, with the master networks displayed in Figure 11 to Figure 14. The nodal size of the networks was limited to 15 due to the significant computational resources required. The cost of each span was linearly related to its length.

For the topology design inputs, the nodal layout of the 8, 10, 12 and 15 node networks were used. Each node was connected to the nearest at minimum 50% of the other nodes. There are certain cases where a specific node is connected to more than 50% of the other nodes in the network. The 50% mark was chosen to reduce the complexity of the ILP problems while still giving valid results, as most network operators would not put a span in their network that bypasses over half the network.

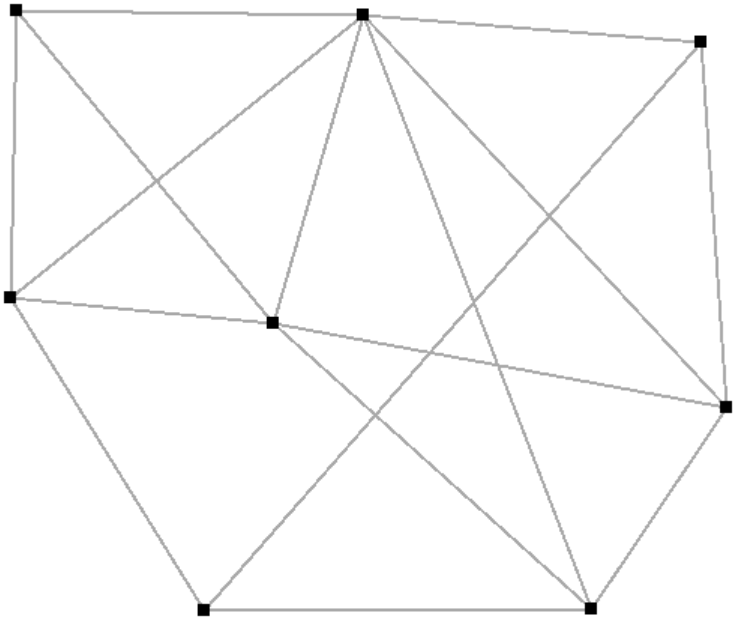


Figure 11 – 8 node 16 span master network

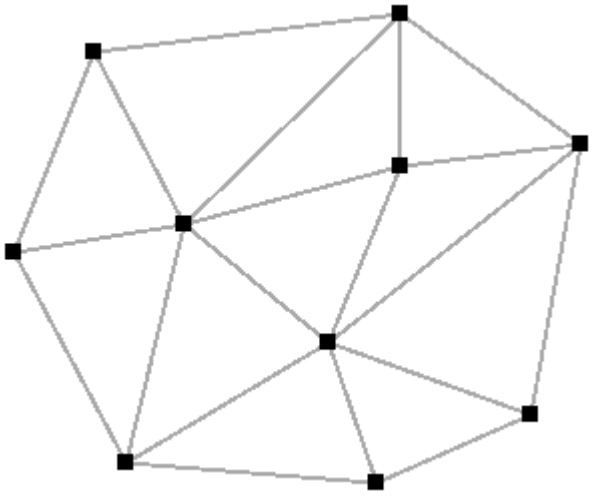


Figure 12 – 10 node 20 span master network

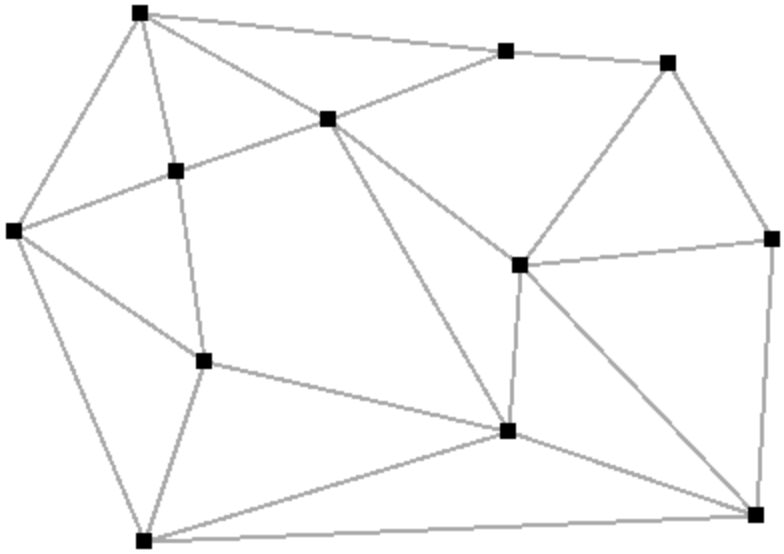


Figure 13 – 12 node 24 span master network

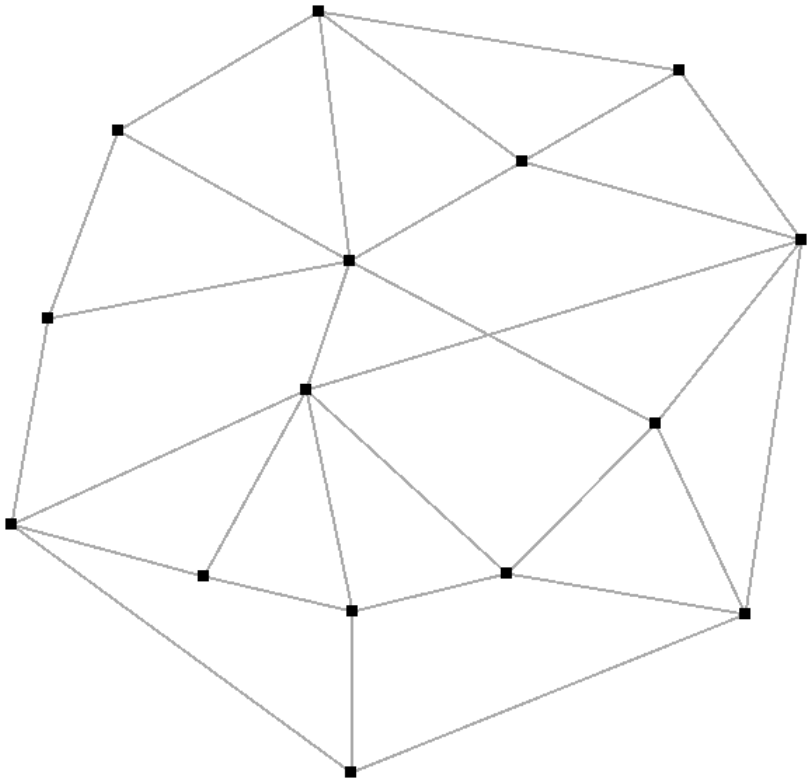


Figure 14 – 15 node 30 span master network

4.2 Assumptions in the Cost of a Network

When modeling a network there are a number simplifications that are required. When estimating the costs involved with adding capacity to a span, there were two prominent assumptions made that could have an impact on the resulting designs: linearity and modularity.

Costs were assumed to increase in a linear fashion based on the length of a span. While this is generally the case, there are many factors which reduce this linearity when establishing actual networks. Some component costs are incremental rather than linear, such as repeaters in long distance spans. When adding a unit of capacity to a span, it could be the case that the cost of transmitter and receiver equipment outweigh the cost of fiber or wire connecting the two. Along with stepped increases in equipment costs, economies of scale can also reduce the accuracy of the linearity assumption. The linearity assumption was kept in this work as the design methodologies were intended to be technology neutral.

The modularity cost assumption is important, especially when dealing with optical networks. It is often assumed that adding one unit of capacity to the span is equivalent to adding a wavelength to the optical system. The cost of the addition of wavelengths is modular, with the cost of increasing the number of wavelengths from say 3 to 4, is not necessarily equivalent to the cost of increasing the number of wavelengths from 4 to 5. The assumption of single unit increases in capacity can still provide reasonable designs, as the representation of a unit of demand can be that of a single module.

The cost assumptions used were consistent with what has been used in literature [13]. However the effects of these assumptions, especially on DSP networks could provide better designs, and is an area of potential future work.

4.3 Implementation Setup

The network survivability models were implemented using AMPL, and solved using CPLEX 11.2. The problems were run on two different machines, with the

majority of the problems run on a dual core AMD Opteron computer with 2GB of RAM running Windows XP, and the more difficult problems run on an Intel 3.0 GHz Core 2 Quad computer with 8 GB of RAM running 64 bit Windows 2003. The mip gap was unique for each problem, and will be discussed with each problem's results.

As an aside, the mip gap is a characteristic of the branch and bound method utilized to solve ILPs. The branch and bound method creates a series of *linear programs* (LP) in a tree like fashion, with each branch representing a sub problem containing the original problem with an additional constraint to determine the optimal integer solution. These branches each have a solution that may or may not meet the integer requirements. The mip gap is the difference between the best integer result and the best non-integer result that has not been fathomed. Obviously this difference could be measured in absolute terms, or as a percentage, with the percentage definition of the mip gap being used in this work.

4.4 Validation of Results

In order to validate the results, a set of programs were created to simulate the routing of all of the single and dual failure scenarios with the path and capacities set out in the ILP results. This validated the legitimacy of the designs, with the optimality validated by the mip gap, where possible. The validity of the models was affirmed by comparing results to equivalent network designs using ILP implementations already presented in literature.

The results are useful for comparison between results with similar cost assumptions. Results, however, are dependent on the network characteristics of the network families utilized in this study. While these networks were chosen as they exhibit similar generic characteristics of actual networks, they are primarily useful for comparative analysis.

In general the results were valid designs, however, they are numerically valid in the context of comparing the results between the various survivability models.

Chapter 5. Results and Analysis

5.1 DSP-TR

The key to efficient DSP network designs is the ability to route traffic on multiple paths with a similar length, and therefore depends significantly on topology. The capacity efficiency in DSP is produced by being able to reduce the required spare capacity, which is done by reducing the amount of traffic affected by a failure. If a demand has r units of capacity, the most efficient design DSP is capable of is a set of r+1 paths, where each path is of equal length. In general the total number of lightpaths, l, required for a demand with k disjoint paths is defined in (64). In practice however, the cost of an additional disjoint path between two nodes compared to an already established set of paths that are of themselves the lowest cost, grows quickly, as the number of paths approach the networks average nodal degree. Therefore the capacity reduction in the first k paths quickly becomes less than the cost of the additional span.

$$l = r + \left\lceil \frac{r}{k-1} \right\rceil \tag{64}$$

$$\left\lceil \frac{r}{k-1} \right\rceil - \left\lceil \frac{r}{k} \right\rceil \tag{65}$$

The reduction in the number of lightpaths required (65) if one more path is added decays at a rate of $\frac{1}{k^2}$. Figure 15 outlines the total units of capacity required to route 1 to 15 units of traffic demand. Since returns are quickly diminishing for utilizing additional paths, topologies that allow high volume demands to be routed

on a number of paths that have similar costs will provide the best results for DSP routing.

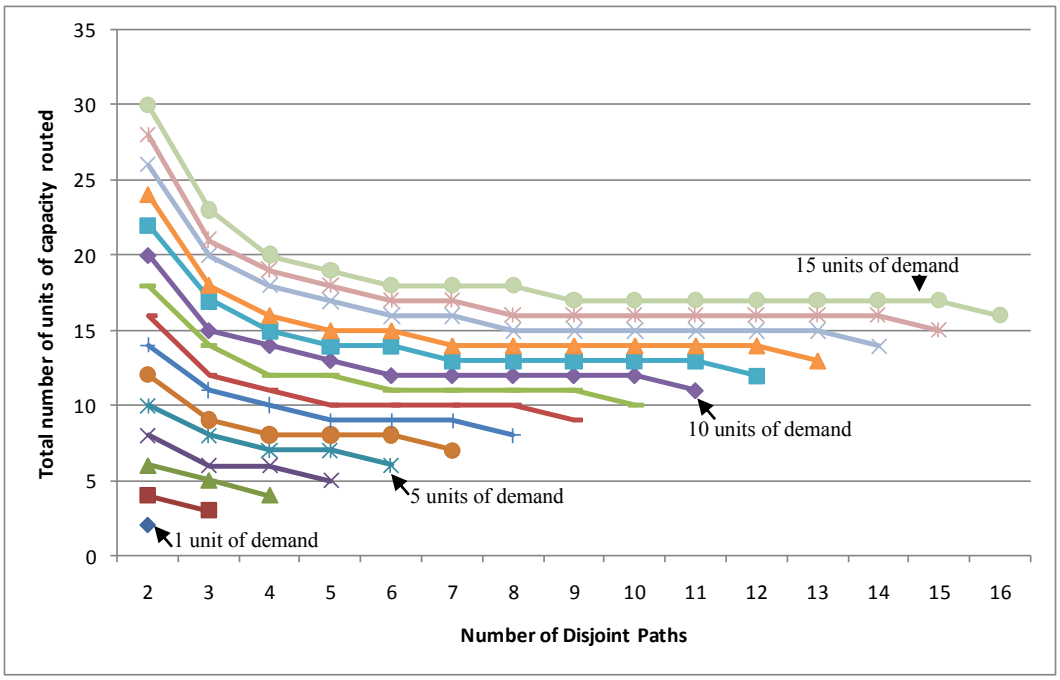


Figure 15 – Total lightpaths required to route various volumes of demand across all the possible number of disjoint paths

The DSP-TR model was run on each network with a mip gap of 0.0001, and all networks took less than a couple of minutes to find a solution within the given mip gap and as such a larger number of network families were utilized when examining path usage. These results form the basis from which the other survivability methodologies can be compared. [32] provides a comparison of DSP to other survivability techniques. Figure 16 displays the normalized total cost of the 8, 10, 12 and 15 node network families. These results leave some questions. First, since the results are normalized, why are there such differences in the low connectivity networks? Another question is why do some network families reduce the normalized cost faster than others as the connectivity of the network increases?

To answer the question as to why there are such large differences in the normalized results of low connectivity, the length of the chains in these networks is observed. A chain is a series of nodes that are connected to each other, and each only have two spans connected to them. These chains have a significant impact on the ability to find multiple diverse routes between two nodes. In the 15 node network family, the addition of a span from the 16 to 17 and 20 to 21 span networks each break up a chain in the network, and each transition have an increased reduction in normalized capacity. This phenomenon is repeated in the other networks.

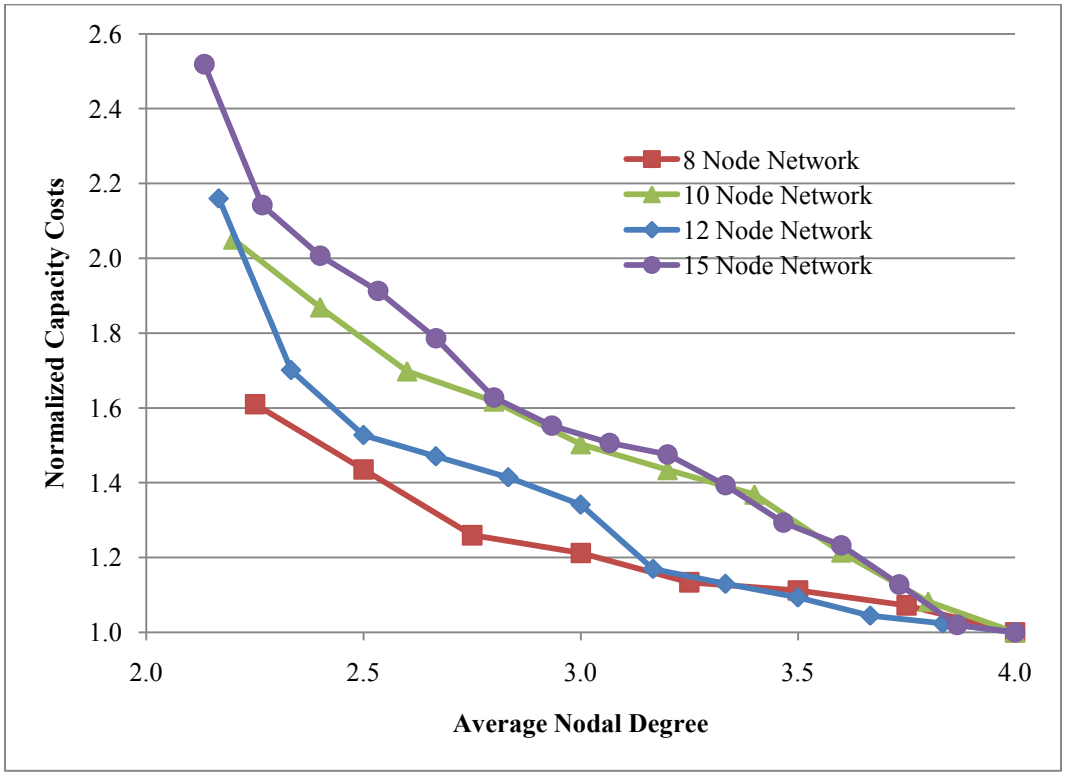


Figure 16 – Normalized costs of DSP-TR results on 8, 10, 12, and 15 node network families

Each data point in Figure 16 represents the results of the DSP-TR ILP, the capacity cost of the network, normalized by the lowest cost network in the respective network family. For example the left most data point in the 15 node network family states that this network, with an average nodal degree of 2.13, had

a total capacity cost of approximately 2.5 times that of the network with an average nodal degree of four.

It is also obvious by looking at Figure 16 that the 8 and 12 node families reduce their costs faster than the 10 and 15 node families. The main contributor to this again has to do with the topology of the networks. Both the 10 and 15 node networks have nodes that if removed, would break the network's bi-connected property, even in the networks with a nodal degree of around 3.6. This means that many of the node pairs cannot find more than two node disjoint paths, eliminating the possibility of utilizing the efficiency of DSP.

It is obvious that topology has a significant impact on the redundancy of DSP network design. Before looking at optimal topology designs, a short discussion on what helps drive some demands to utilize more than two paths. The data used in this discussion comes from the results of the DSP-TR ILP from each of the network family's master networks. In order to look at a wider range of networks, master networks that had 20, 25, 30, 35 and 40 nodes were also included. Figure 17 displays the distribution of the number of paths that demands used, and emphasizes the decreased capacity savings, as the number of paths increased.

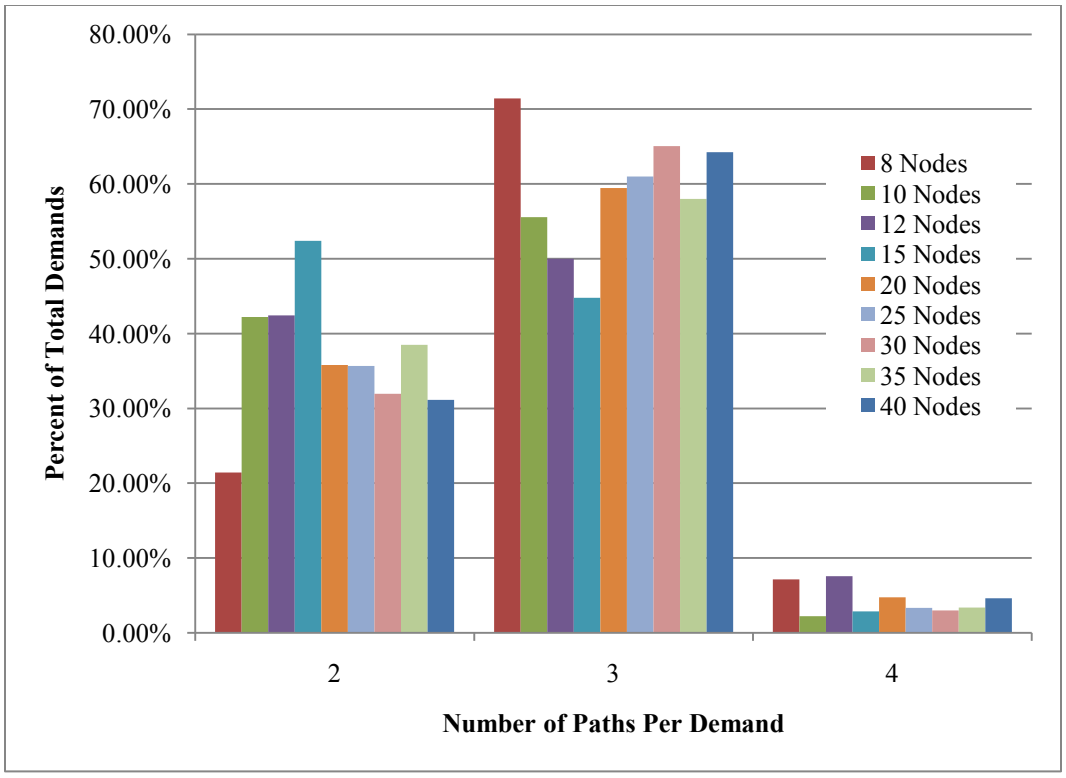


Figure 17 – Comparison of the number of paths used per demand

The volume of traffic for each demand in the 8 to 40 node networks was compared in Figure 18. Obviously no demand with a volume of 1 unit will ever use more than 2 paths, and hence the graph shows the average number of paths for demands with 1 unit of capacity as 2. The rest of the paths average 2.76 paths per demand, and there is no significant correlation between the number of paths and the volume of the demand.

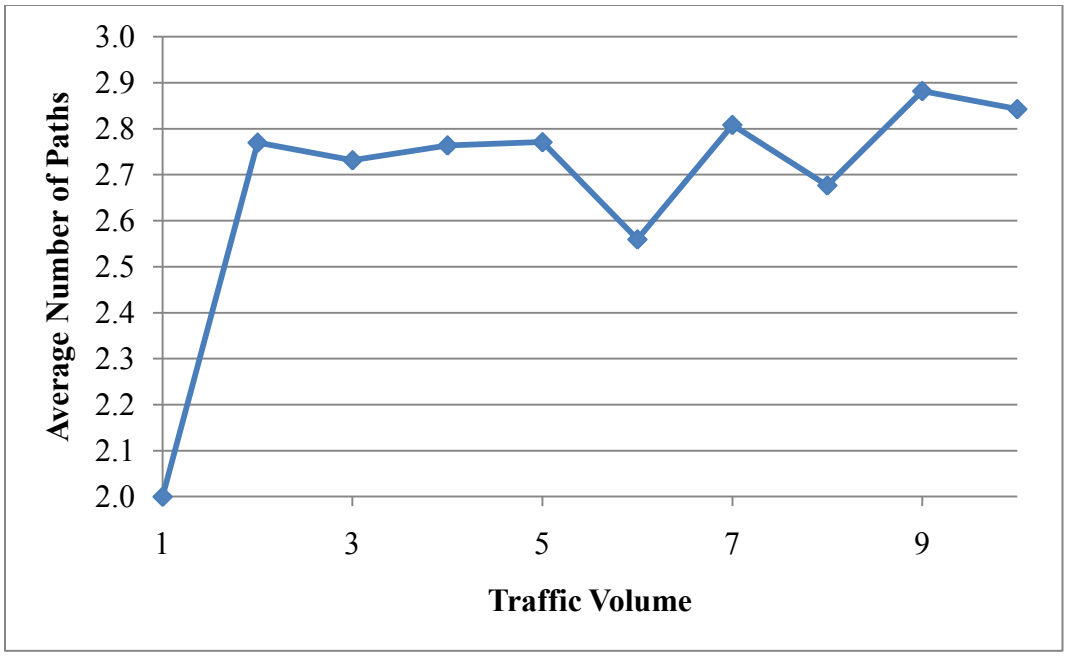


Figure 18 – Comparison of demand traffic and number of paths utilized across all master networks

Another factor that was investigated in order to determine if it had an influence on the use of multiple paths was the length of the shortest path between the two end nodes of each demand (Figure 19). The length of other possible paths were not looked at, as it would be difficult to determine these paths before the ILP was run, and the purpose of looking at the path lengths is to try to determine topology indicators of the utilization of multiple paths in DSP-based restorable networks. Generally it was the demands with longer shortest paths between the end nodes that utilized more than two paths; however, the trend is not as consistent when looking at 3 and 4 paths. Both the 8 node and the 30 node networks had on average, longer minimum length paths for demands using 3 paths compared to demands using 4 paths.

The reason that demands that are farther apart are more likely to utilize more than two spans is found in the reduction in effectiveness of DSP to reduce costs as the disparity in the length of the possible paths increase. If two nodes are close together (in terms of the span topology) the first path will be very short, and the distance will grow significantly for additional paths, as additional paths will most

likely have to travel away from the destination in order to remain disjoint, and even one hop in a direction away from destination could, in percentages, be dramatically larger than the first path (if the shortest 2 paths are 2 hops away, and the third requires 3 hops it is an increase of 50% in path length). When two nodes are relatively far apart in a network, a hop away from the wrong direction has less of an impact (if the first two paths are 10 hops away, and the third is 11, this third path is only an increase of 10% in path length).

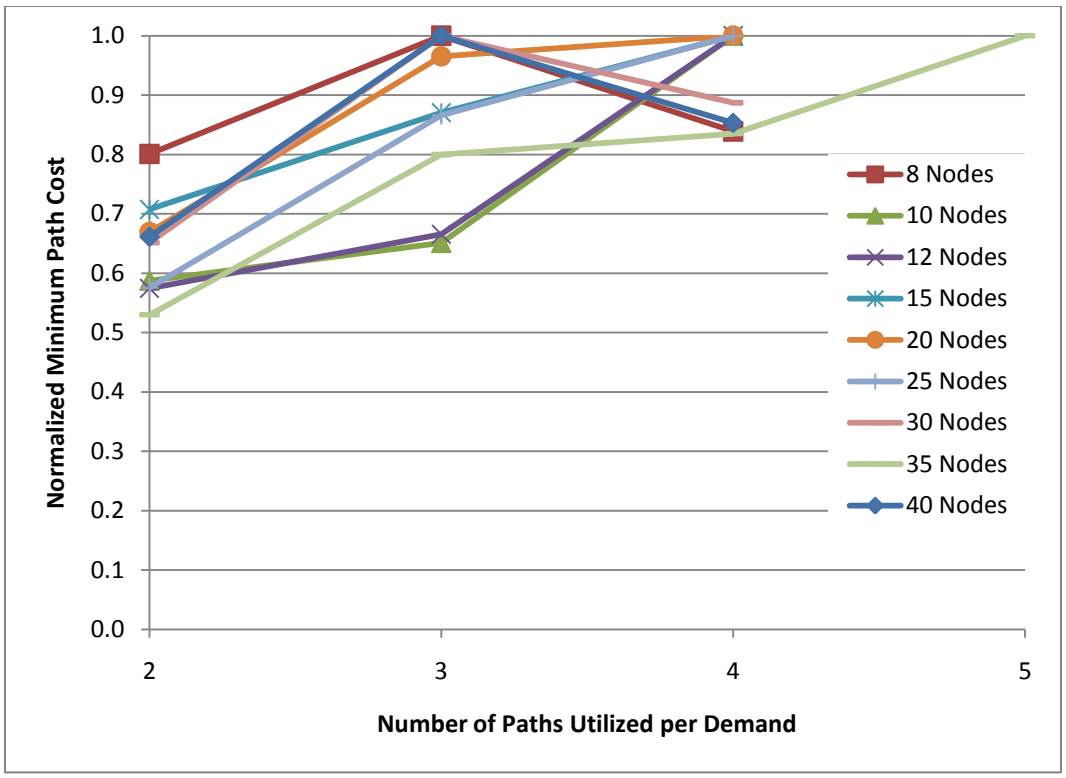


Figure 19 - Length of the shortest path for a demand vs. the number of paths the demand uses for DSP

5.1.1 Availability

One last aspect of the DSP results that needs to be looked at when discussing the design of high availability networks is the availability of the design. Of interest when looking at DSP is how is availability affected by utilizing multiple paths? Whenever overall capacity is reduced, there is an expectation that availability will decrease as well, when looking at similar methods of survivability, as there are

fewer options to route traffic when failures occur outside of what was explicitly protected against. A study that looked at utilizing multiple working and backup paths in shared backup protection showed that multiple paths had a small negative effect on demand availability [19].

When looking at the DSP designs, Figure 20 shows the average unavailability for demands with varying number of paths. This figure shows a consistent decrease in availability as the number of paths increase. This is due to a couple of reasons. First, as mentioned above, demands that utilized more than two paths tend to be further apart in the network, and therefore are more susceptible to failure due to the longer distances that the traffic must traverse. The other reason is that demands that utilize more than two paths would more readily be susceptible to dual failures. Although the impact on the amount of traffic is less severe for these demands, their occurrence would be significantly increased. This was confirmed in the results with demands that had shorter minimum path lengths but used more than two paths had a lower availability that demands with a longer shortest path but only utilized two paths.

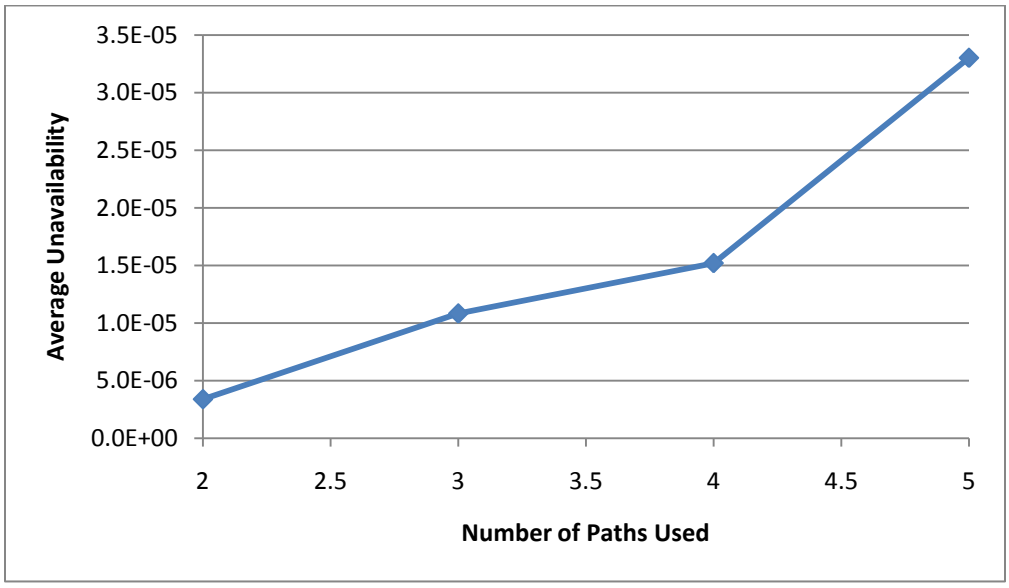


Figure 20 - Availability compared to number of paths used

5.2 DSP-Top

In order to discuss efficient topology designs, there must be a cost of utilizing a potential span between two spans in a network. If there was no cost, other than the cost of adding a unit of capacity, then the optimal networks would be fully connected. Obviously this isn't the case. In order to connect two nodes, a physical right of way must be obtained and the cables must be placed in the ground, and there is a cost involved for each. Different geographies have different costs for both factors. Rights of way can vary in cost based on a number of factors. Being able to co-locate cables with other utilities can reduce the costs, but this can make enforcing spans to not be concurrently susceptible to single failure more difficulty [33]. Also, the cost of physically placing the cables can vary due to whether the cable is in an urban or rural location, undersea, or many other factors. Since topology costs can vary, and hence vary the optimal network layout, a number of different implementation costs must be taken into account.

In order to compare the solutions of networks with varying implementation costs, three costs are presented, the implementation, capacity and unit implementation costs. The implementation cost is the cost associated with obtaining the rights to place a cable, and physically place it in the ground. This implementation cost was calculated by multiplying the cost of adding a single unit of capacity to a span by an implementation factor. The implementation factor was a multiplier that allowed the variation of the implementation cost in a consistent manner in order to investigate the implications of the implementation cost relative to the capacity costs of a network. The capacity cost is the sum of the number of lightpaths on each span multiplied by the cost of adding one unit of capacity on the span. When evaluating networks, this capacity cost is the typical cost metric. In summary, each span was assigned a fixed cost if any capacity was assigned to it, the implementation cost, and a variable cost corresponding to the volume of capacity assigned.

The normalized implementation cost shows the impact of the implementation cost on the network topology in terms of unit cost. The unit cost is indicative of the

length of service right of ways required by a network. Obviously it is expected that as the cost of implementing potential spans increase, the overall unit implementation cost of the network will go down. This metric was included in order to observe how the size of an optimal network is affected by the implementation costs.

Within the ILP, the goal is to minimize the total cost of the network, being the implementation and the capacity costs. The interaction between the number of utilized spans in a network, and the overall capacity required in a network is important when designing new networks, as well as modifying the topology of current networks.

The DSP-Top ILP was run on four nodal layouts. These layouts were taken from the network families, and have 8, 10, 12, and 15 nodes, identical to the corresponding network families (Figure 11-Figure 14), with the same demand profiles as each of the corresponding networks. The set of eligible spans for each network, as mentioned, was created such that each node had the possibility of being connected to at least the nearest 50% of the remaining nodes in the network. The nearest 50% of the remaining nodes was chosen to reduce the overall problem size as well as eliminate possible spans that would not be practical in actual networks. Each network was run with costs of implementing a given span at 20, 50, 100, 200 and 500 times the cost of adding a single unit of capacity to the network. These multiples are referred to as the implementation factor and were chosen in order to evaluate the topology design across a variety of implementation costs.

There were some significant difficulties with solving the DSP-Top ILP's. Inherent in topology design is a problem structure that is very difficult for linear programming to obtain optimal results. Because changing topology can drastically change the paths chosen for each demand, topology ILP's have many local minima, and can be difficult to solve when using the branch and bound solution methodology. This problem affected running DSP-Top ILP's. Two different techniques were used in order to achieve the best results in the minimal

amount of time. First, the method of finding solutions was to alter the ILP, and for second method the ILP's were run to a time limit (if necessary).

In order to get solutions in a reasonable amount of time (less than 24 hrs per network), a two-step process was chosen to come up with a topology and routing solution. The DSP-Top ILP had the integrality requirement for the $\omega_{i,j}^{r}$variable relaxed. By doing this, the solution time was drastically reduced, as running the ILP for 3 hours would provide reasonable solutions with the integrality relaxed, and full ILP results would run out of memory before a solution could be found. The impact on the results however was that paths had fractional amounts of capacity assigned to them. The structure of the paths were valid results, however, assigning capacity must be done in integral amounts. The topology was taken from the relaxed ILP, and the network capacities work established by running the DSP-TR ILP on the new topology.

Since the ILP run times were limited, some results are non-optimal. The 15 node network results were interrupted at 3 hours, and the remaining mip gap is outlined in Table 2. All other results were solved with a 1% mip gap. A 12 node and 15 node ILP model were each left to run for over one week without reaching the required mip gap. It was found that the optimal integer solution did not differ significantly between the results after two to three hours, and those that were available after a week. Although these results have significant mip gaps, there is a reasonable confidence in the quality of the results.

Implementation factor	Mip Gap
20	0.2094
50	0.3150
100	0.3681
200	0.3797
500	0.3357

Table 2 – Mip Gaps from DSP-Top results on 15 node network

Figure 21 through Figure 24 graph the unit cost (green triangles), implementation cost (blue diamonds), and capacity cost (red squares) of each of the 8, 10, 12 and 15 node networks. The unit implementation cost is the total implementation cost divided by the implementation factor. Because the unit cost of a network was on a different scale as the implementation and capacity costs, it was labeled on the secondary vertical axis.

Of note, the relationship between the implementation and capacity costs are affected by the total area of the networks. Hence in Figure 21 and Figure 24, the implementation and capacity costs intersect, while they do not in Figure 22 and Figure 23. The 8 and 15 node networks cover a larger area, and hence have higher unit implementation costs, as well as different ratios between the capacity and implementation costs, when compared to the 10 and 12 node networks.

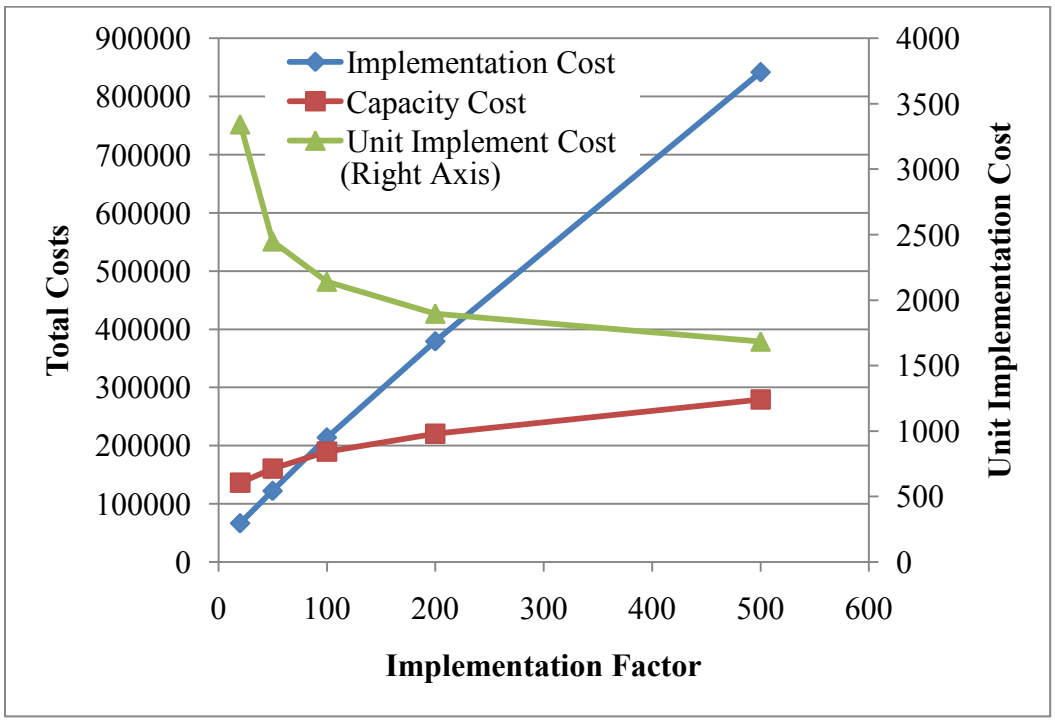


Figure 21 – Costs from DSP-Top 8 node network

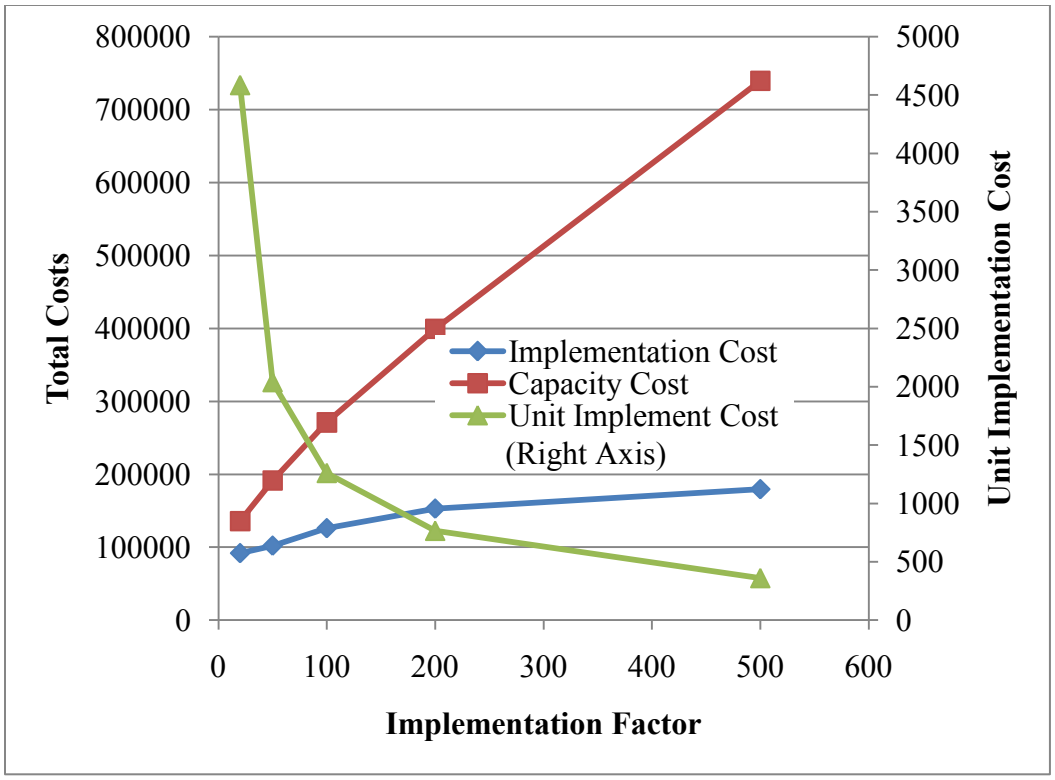


Figure 22 – Costs from DSP-Top 10 node network

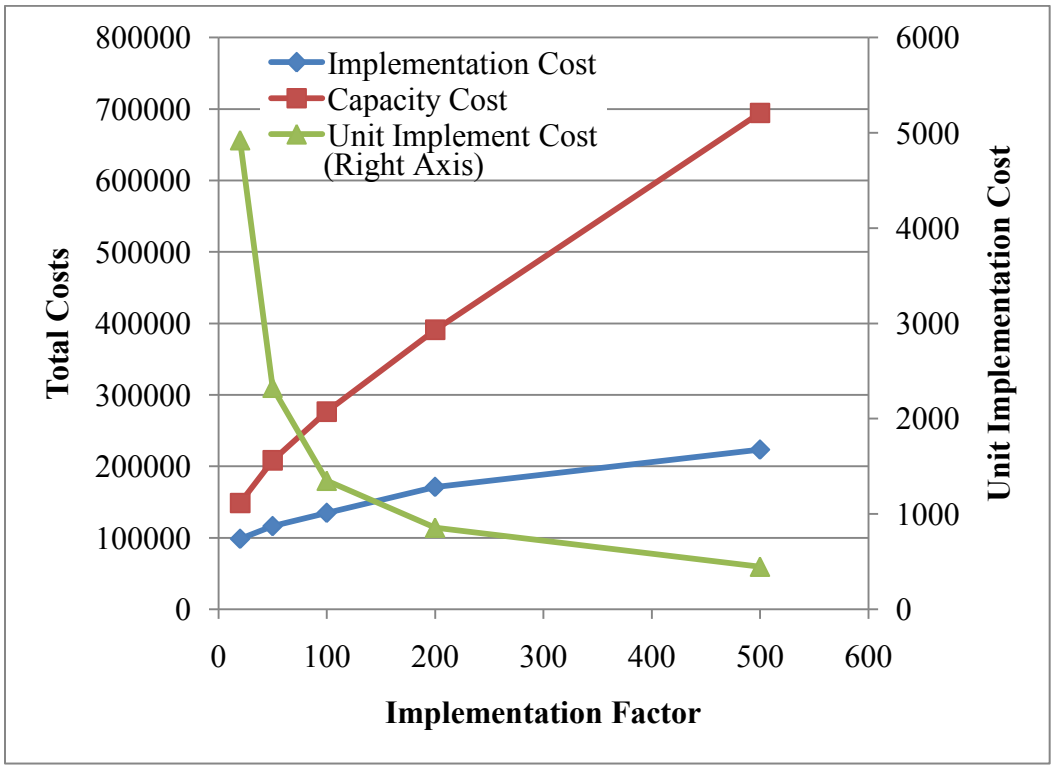


Figure 23 – Costs from DSP-Top 12 node network

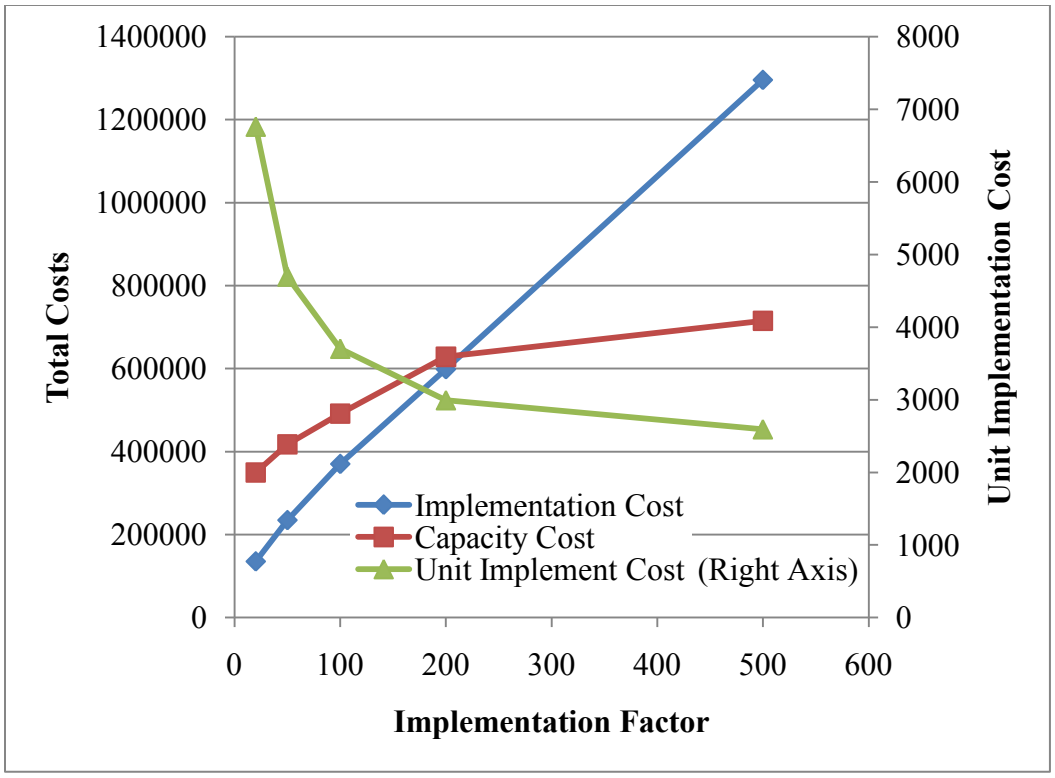


Figure 24 – Costs from DSP-Top 15 node network

All four networks had similar properties in their cost graphs. The unit cost was generally exponentially decreasing. The reason for this is found in the diminishing returns of adding paths in DSP combined with the requirement that a network be bi-connected. For networks that have a relatively low implementation cost, the optimized topology of the network is obviously large, however, as the implementation cost increases, the efficiencies gained through utilizing multiple paths are quickly eliminated by the increasing cost of implementing each span. This is evident in looking at the distribution of the number of demands using multiple paths.

Figure 25 shows the percentage breakdown of how many demands use only 2 paths as the implementation factor increases. The anomaly is in the 10 node network results. The 50 and 100 times implementation factor results have the same number of demands using more than two paths. The topologies are different

for each network, and different demands utilize more than two paths for each design. The nodal layout and the demand distribution allow for this special case. The capacity and unit implementation costs in Figure 22 however remain consistent with the other networks. The multiple paths used in the 10 node -100x network design were longer and less efficient than what was used in the 10 node -50x design, and hence the unit costs remain exponentially declining.

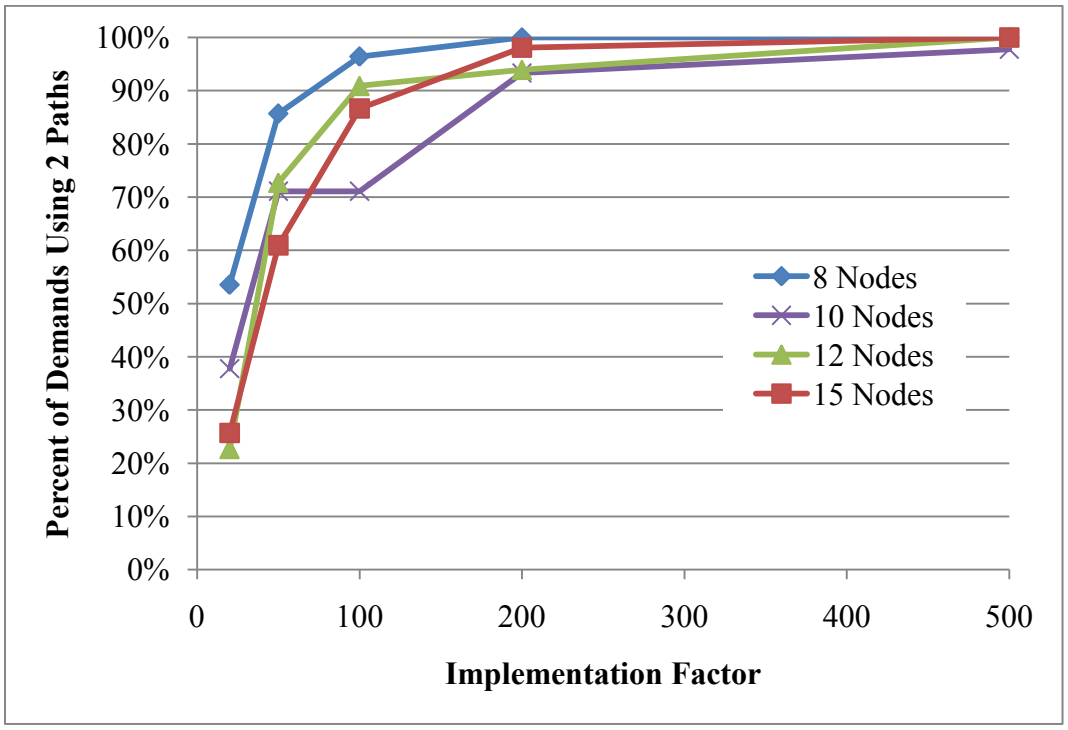


Figure 25 – Percent of demands using exactly 2 paths for DSP-Top as the implementation factor increases from 20 to 500

The implementation costs appear nearly linear as the implementation factor increases. However, there is a small exponential component. The increasing installation cost dominates the overall network cost, and since the relationship to the implementation factor is linear, the implementation cost also appears to be linear. Dividing the implementation cost by the implementation factor derives the unit capacity cost, and hence there is a small exponentially decreasing component

in the overall installation cost due to the reduction of the number of spans in the network.

When comparing the normalized implementation costs of all the networks, there is a noticeable difference between the networks that solved optimally, and those that were time limited, as shown in Figure 26. It was expected that the normalized costs would follow the same rate of increase, as the factor affecting the implementation costs should have affected the network designs in similar manners. The cause of this discrepancy cannot be fully investigated due to computing limitations, and could have been due to the time limited solutions for the 12 and 15 node networks not being able to find optimal topologies, especially at lower implementation costs.

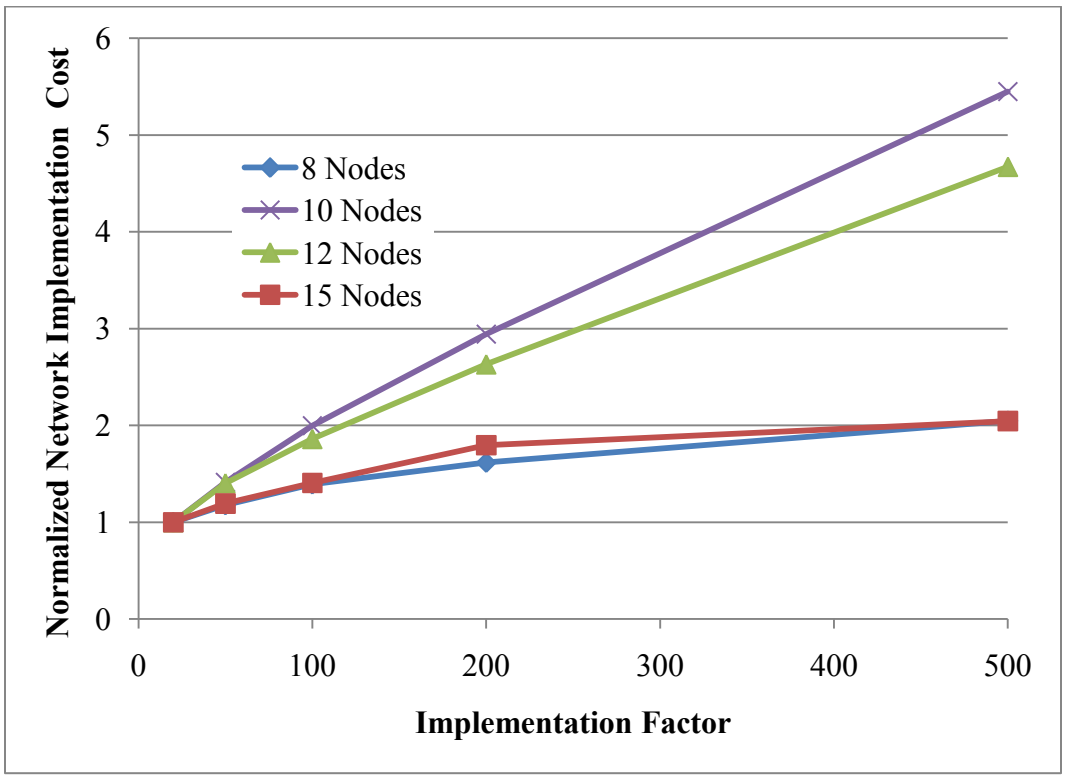


Figure 26 – Network costs normalized by network nodal count across varying implementation factors

Overall, costs of all 4 networks are very similar, despite having significantly different demand and nodal topology configurations (Figure 27). As a note, the capacity costs were normalized to the minimum capacity cost networks for each

nodal configuration. Therefore while the results are only from a small sample, it does appear that the impact of the implementation factor is consistent across various networks.

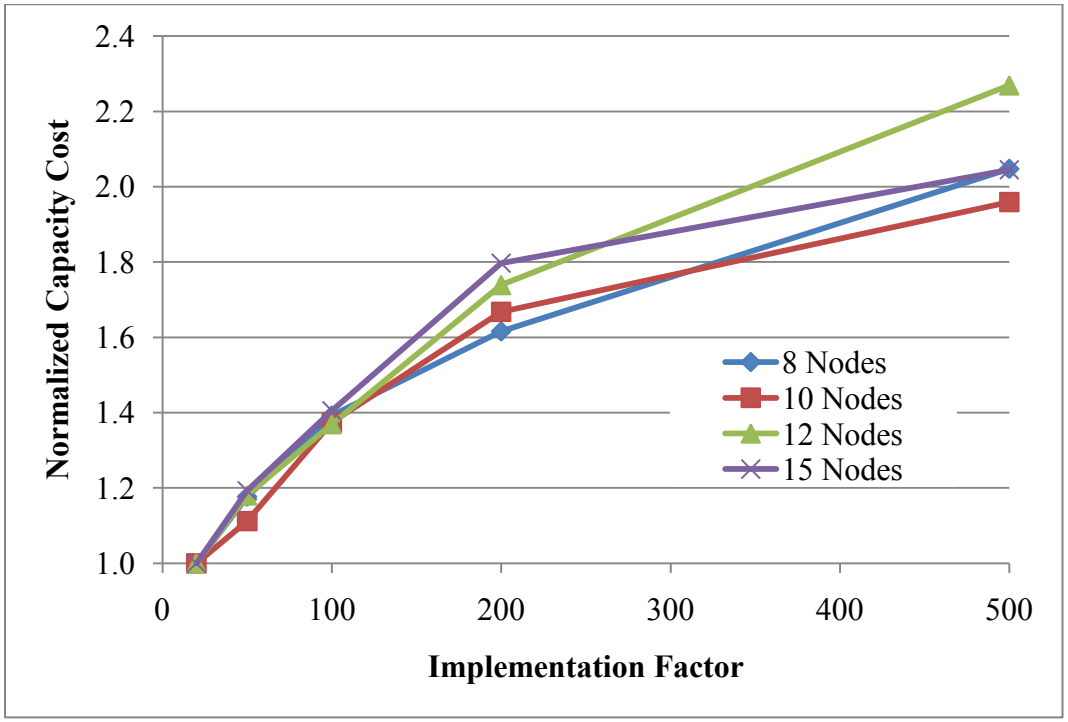


Figure 27 – Comparison of Normalized Capacity Costs for DSP-Top results for 8, 10, 12 and 15 node networks

5.2.1 Availability

Although the DSP-Top does not take into account availability or dual failure restorability requirements, it does ensure 100% single failure restorability, which does significantly enhance availability compared to an unprotected network. The availability results show the networks divided into two groups (Figure 28). The cause behind the higher availability in the 10n20s1 and 12n24s1 networks is the overall network size. Figure 26 shows the implementation costs for each network normalized by the nodal count of each network. In this figure it is obvious that the 10 node and 12 node networks have an increased nodal density when compared to the other two networks. Having nodes relatively closer together

implies that the distance traffic must travel is reduced, and hence less prone to failure.

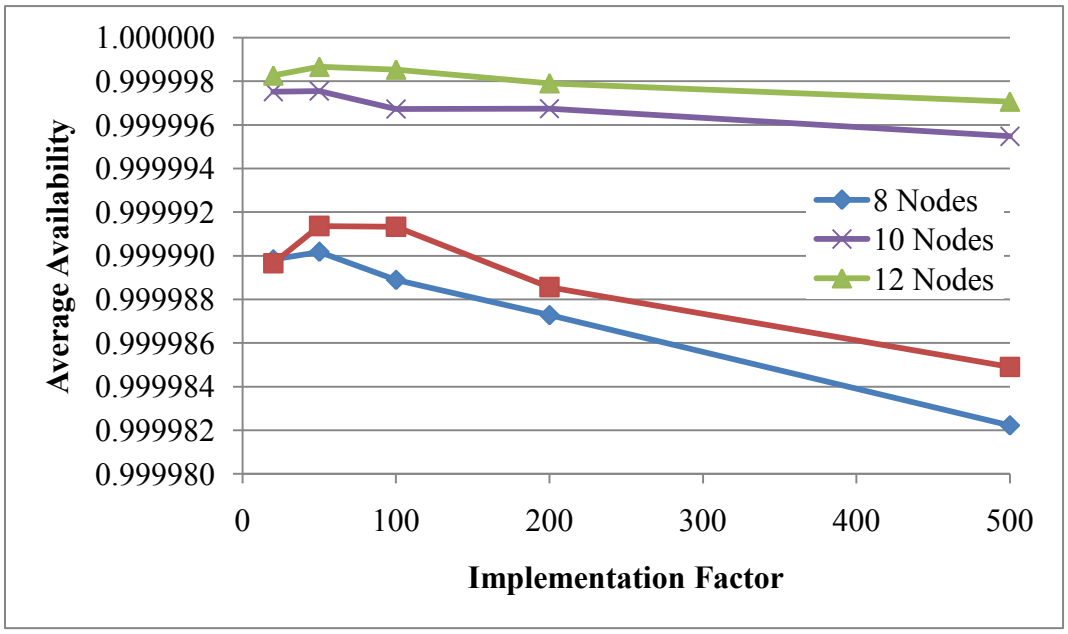


Figure 28 – Average availability of topology designed networks across various implementation factors

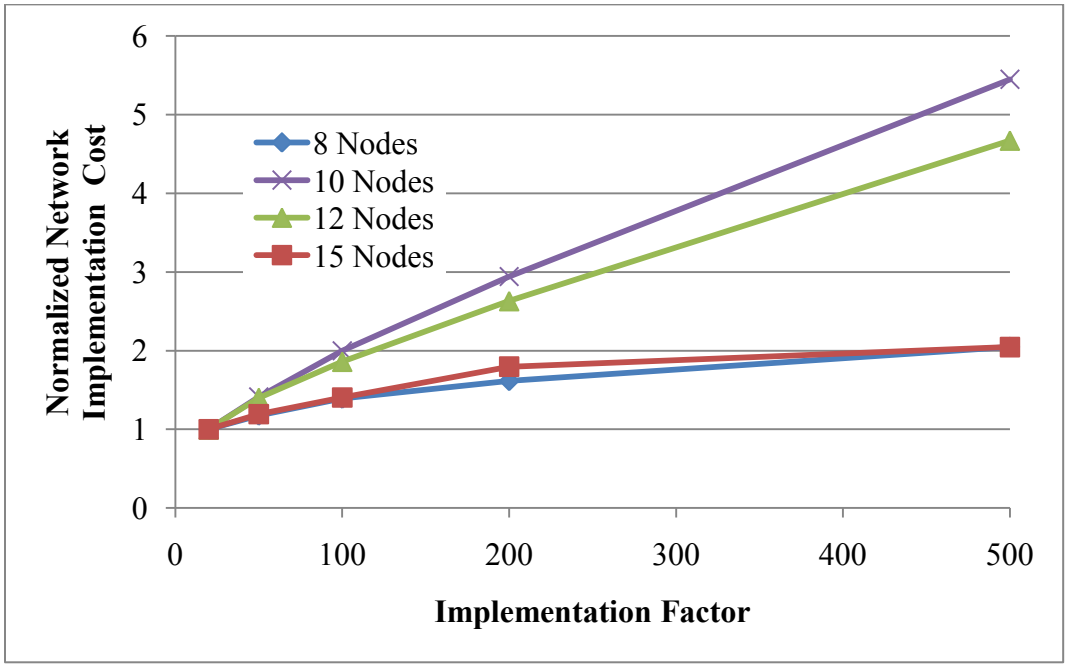


Figure 29 – Normalized Capacity Costs of the DSP-Top results for the 8, 10, 12 and 15 node networks across varying implementation factors

While it is interesting to look at availability on these networks, the results are based on estimates of real world parameters and the value at looking at availability is comparative to other designs with similar parameters. The next section compares the topology designs with the results from the network families designed using DSP-TR.

5.2.2 Comparison of DSP-Top and DSP-TR results

The topologies that were obtained from the ILP models could not be labeled as optimal due to the limited solution timelines, and the relaxation of the integrality requirements. This raises concerns of how well the two-step methodology utilized to come up with the final DSP-Top designs compare to results of non-optimized topologies from the DSP-TR results. If the topology design results were to have costs higher than the DSP-TR results, the effectiveness of this topology could be called into question.

The implementation costs were added to the capacity costs from the DSP-TR results for each of the network families in order to compare to the DSP-Top results. Figure 30 through Figure 33 display this comparison.

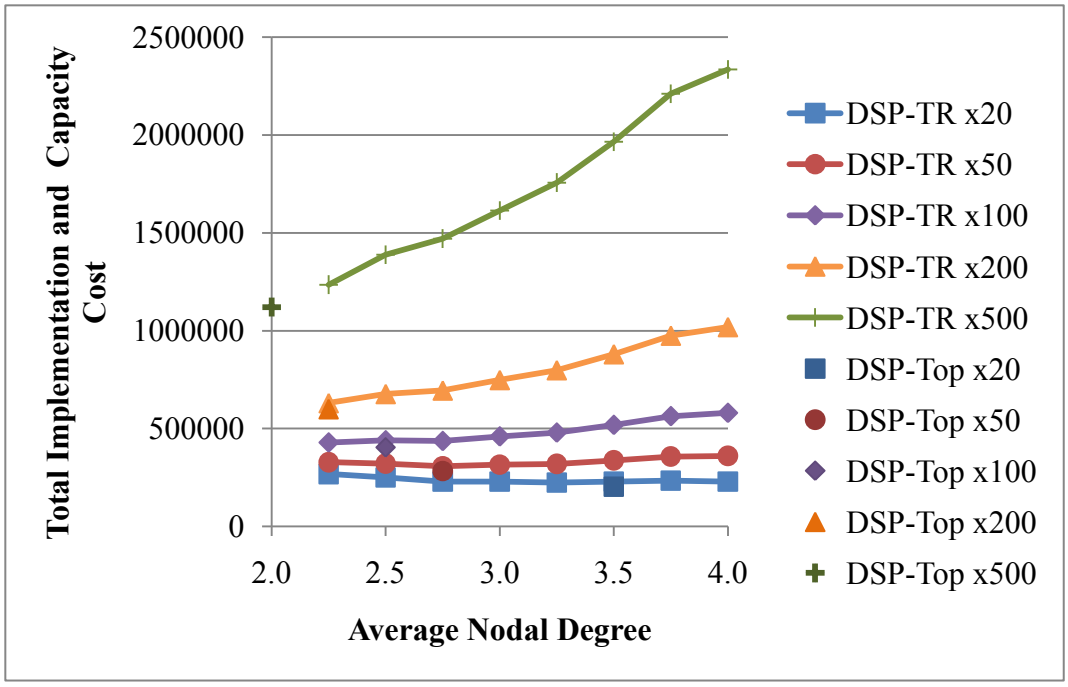


Figure 30 – Comparison of DSP-TR and DSP-Top costs for the 8 node network family

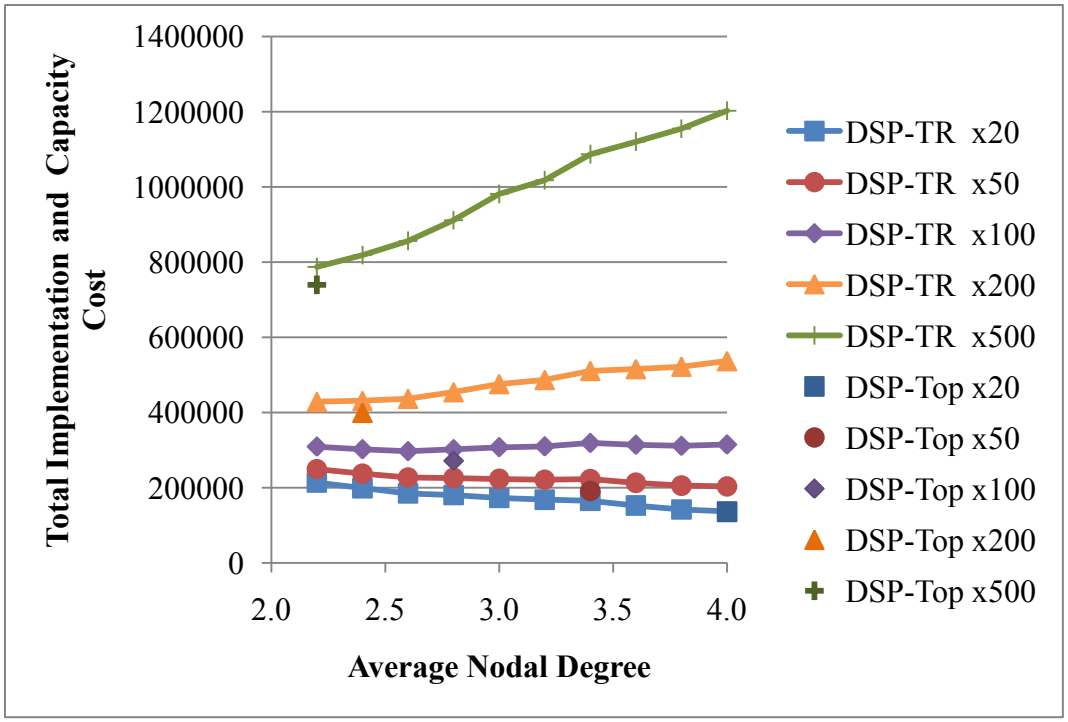


Figure 31 – Comparison of DSP-TR and DSP-Top costs for the 10 node network family

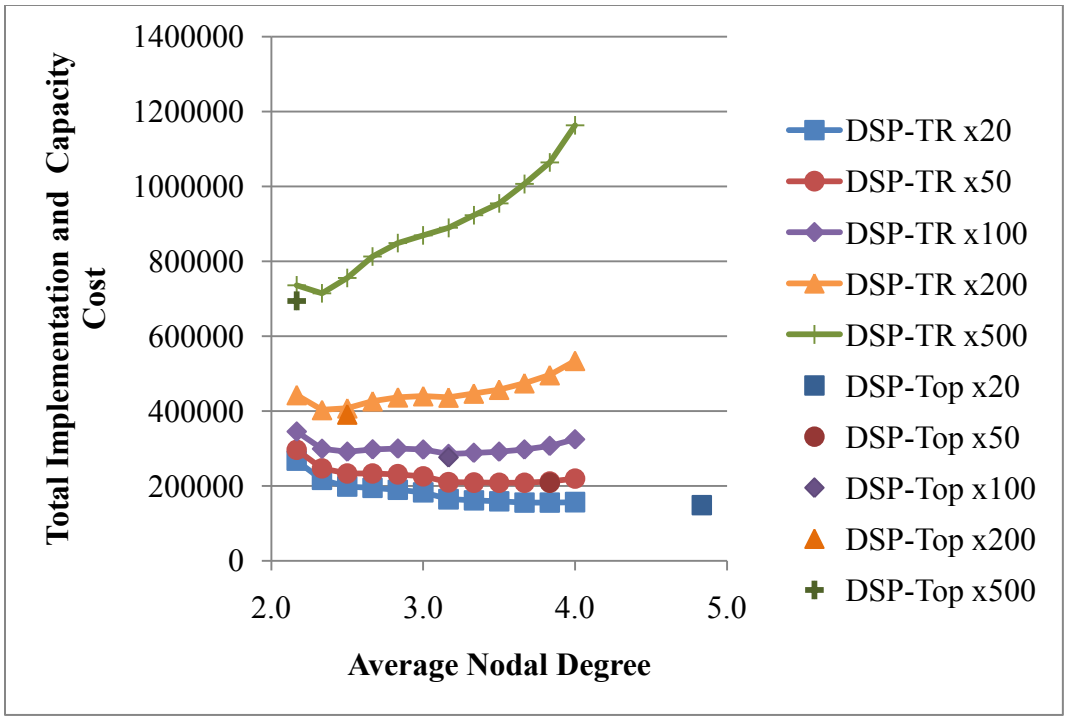


Figure 32 – Comparison of DSP-TR and DSP-Top costs for the 12 node network family

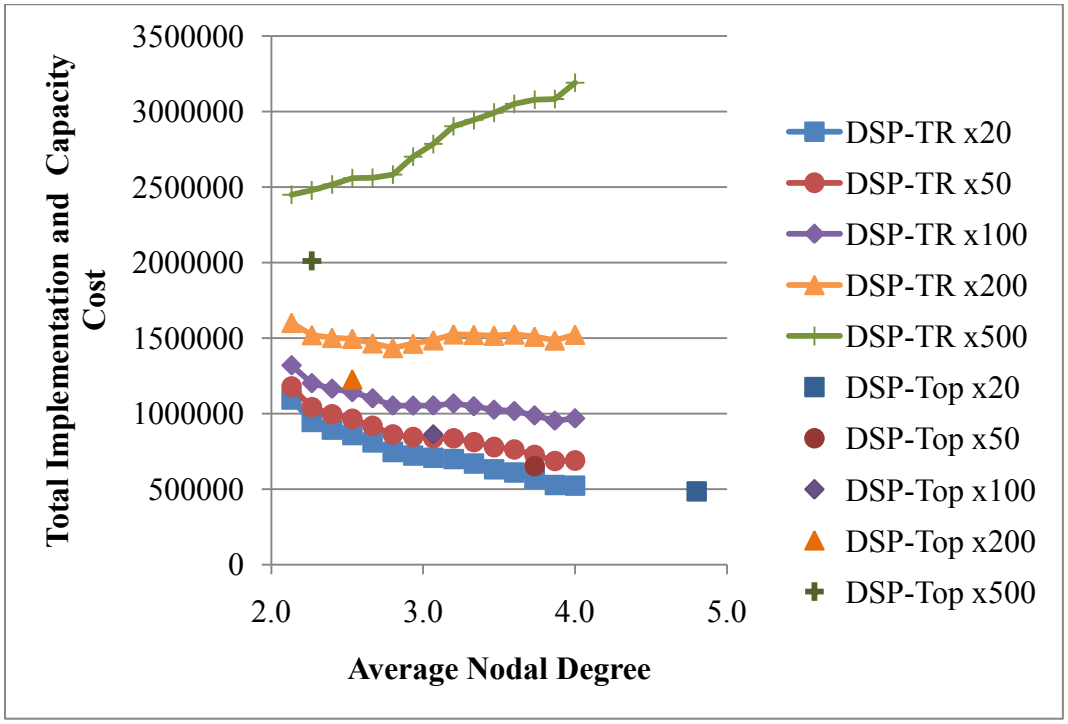


Figure 33 – Comparison of DSP-TR and DSP-Top costs for the 15 node network family

In all cases the results from the topology designed networks were less in total cost than the DSP-TR results. The amount of improvement (Figure 34) ranges from 0.007% to 17.86%, with the mean improvement being 6.64%.

It is interesting to note that a network whose topology is designed to meet the demand requirements of a network does not provide a significant cost reduction as compared to network topologies that have not been arranged with any foresight into the survivability mechanism and design. However, the fact that none of the topology designs had a higher cost than the network families can provide a level of confidence in the efficiency of these topology designed networks.

In Figure 32 many of the cost graphs for DSP-TR networks have a clearly defined minimum. The other networks have this as well; however, they are not as well defined. The optimized topology designs all had average nodal degrees very similar to the minima in the DSP-TR graphs.

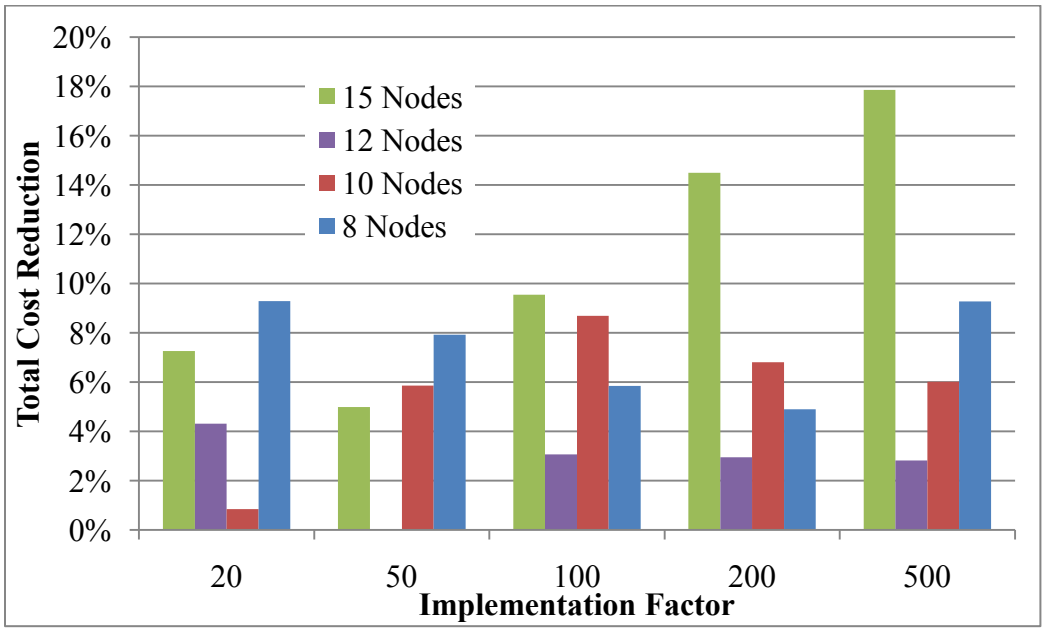


Figure 34 – Percent total cost reduction of topology designed networks and minimal cost network family results for implementation factors of 20, 50, 100, 200 and 500

All of the topology designs are provided in the Appendix; however, a few examples are provided in order to provide an example of topologies at each implementation factor. The savings for each pair of networks are 9.3% for Figure 35, 5.86% for Figure 36, 8.69% for Figure 37, 2.95% for Figure 38, and 17.86% for Figure 39.

It is interesting to note that the networks that are most similar (Figure 38) with a difference of two spans also had the smallest improvement (2.95%). This suggests that this network already had a very efficient topology in the network family. Overall, though, the topologies did not contain many common design patterns, either amongst themselves or with the network families.

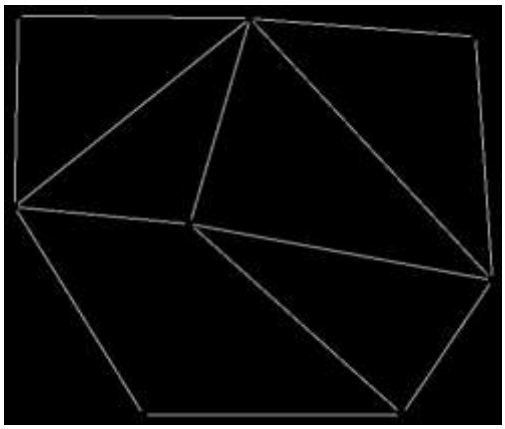
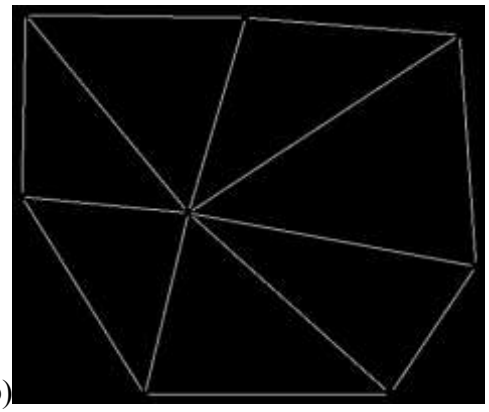
a)  b) 

Figure 35 – a) Minimum total cost network for the 8 node network family with an implementation factor of 20: the 8 node, 13 span network, and b) DSP-Top network topology from 8 node configuration with an implementation factor of 20

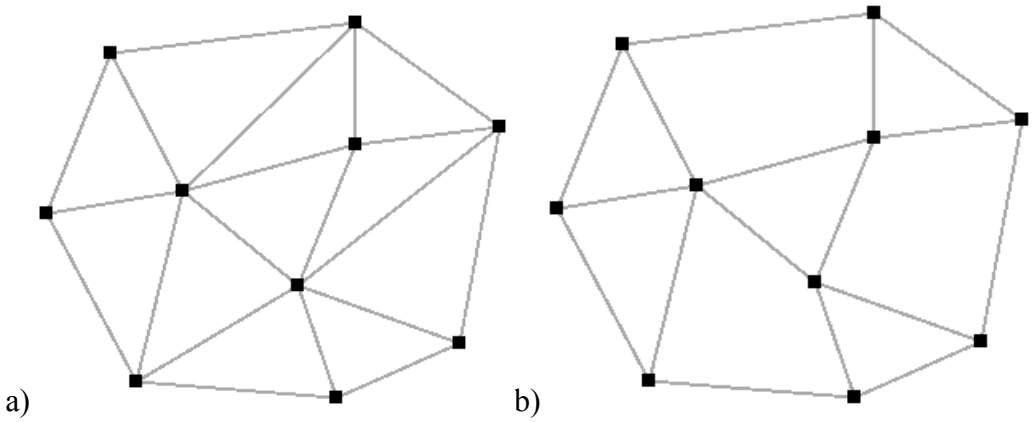


Figure 36 – a) Minimum total cost network from the DSP-TR problem for the 10 node network family with an implementation factor of 50: the 10 node, 20 span network, and b) DSP-Top network topology from 10 node configuration with an implementation factor of 50

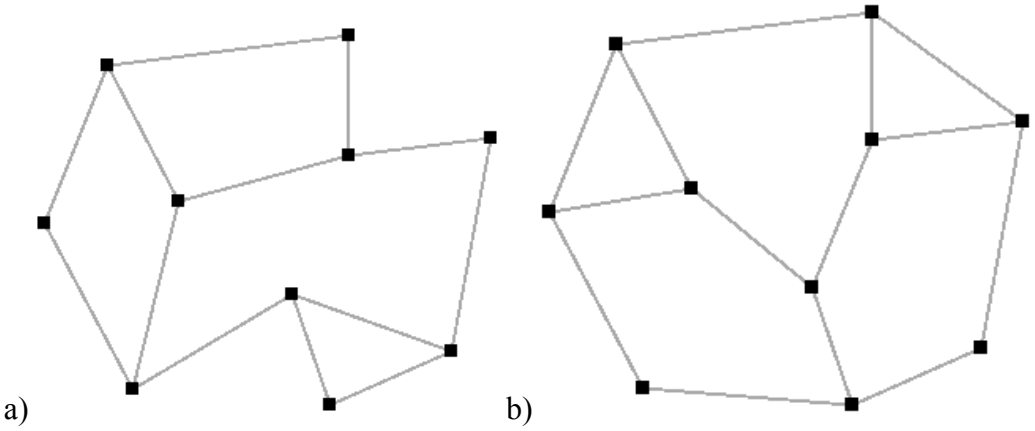


Figure 37 – a) Minimum total cost network for the 10 node network family with an implementation factor of 100: the 10 node 13 span network, and b) DSP-Top network topology from 10 node configuration with an implementation factor of 100

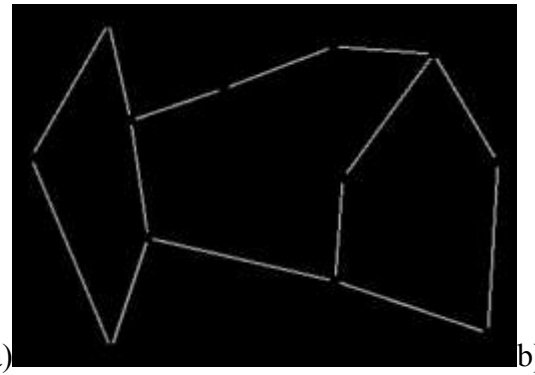
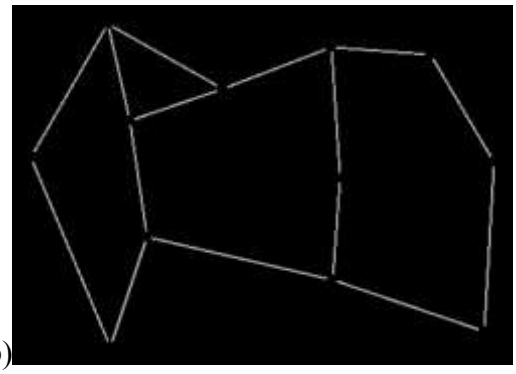
a)  b) 

Figure 38 – a) Minimum total cost network for the 12 node network family with an implementation factor of 200: the 12 node 14 span network, and b) DSP-Top network topology from 12 node configuration with an implementation factor of 200

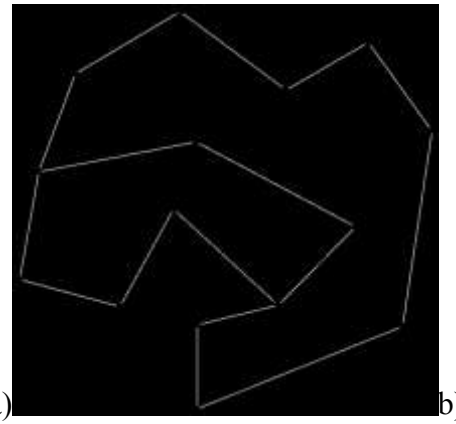
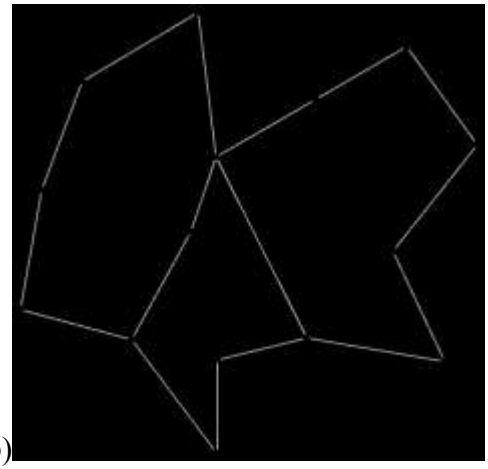
a)  b) 

Figure 39 – a) Minimum total cost network for the 15 node network family with an implementation factor of 500: the 15 node 16 span network, and b) DSP-Top network topology from 15 node configuration with an implementation factor of 500

5.2.3 Investigation into the causes varying nodal degree

The networks were also analyzed to determine if there is a correlation between the nodal degree of each node in the designed topologies, and the amount of demand that originated or terminated at it. Figure 40 charts the average portion of demand that originated and terminated at each nodal degree for the topologies designed using the DSP-Top ILP. There is no significant correlation between the amount of traffic in or out of a node, and the nodal degree. The main factor in what

causes some nodes to have a higher nodal degree than others depends on the relative location of the nodes and overall demand patterns, rather than traffic characteristics of a particular node.

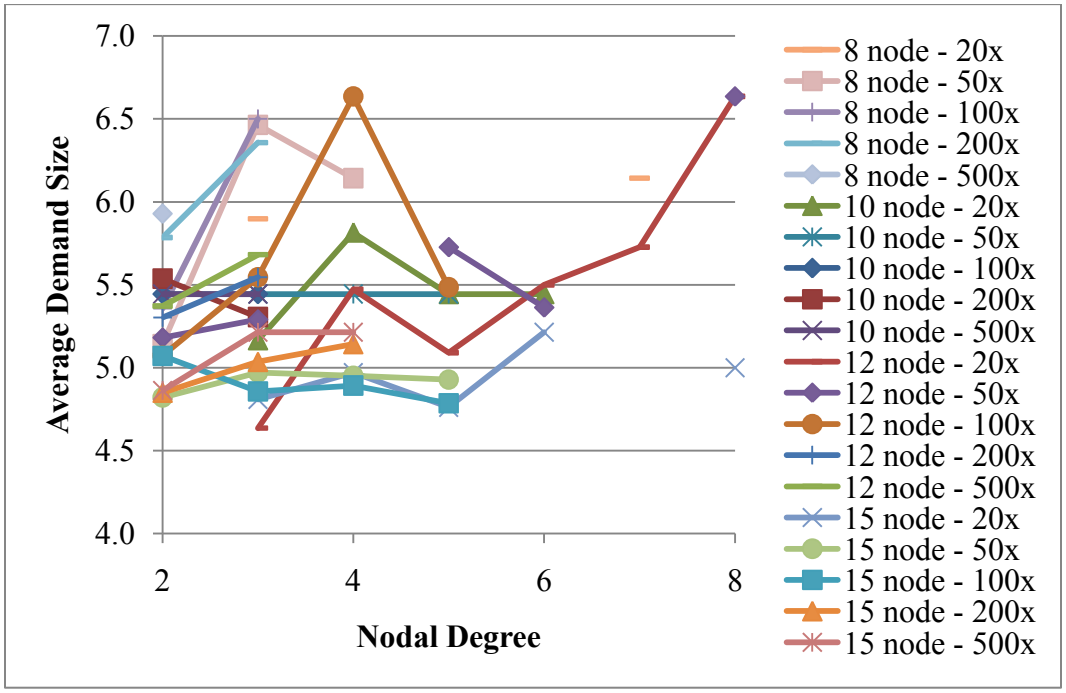


Figure 40 – Average percentage of total capacity that originated or terminated at nodes of various degrees from DSP-Top based topologies

5.2.4 Availability Comparison of DSP-TR and DSP-Top

When comparing the availability of the DSP-TR and DSP-Top network designs, there are two competing factors affecting relative performance. The topology designed networks had less capacity assigned to them, which commonly decreases network availability. The other factor, which would increase availability, is that the topology designed networks could arrange the spans in order to more directly route demand traffic that has relatively high volume requirements.

Figure 41 through Figure 44 compare the availability of the DSP-TR designs with the DSP-Top networks. In all but three networks (8 node -20x, 10 node-500x, and 12 node-200x) the DSP-Top network designs had a higher availability than DSP-TR network designs of a similar nodal degree. These results emphasize the impact of the topology on network availability, with efficient topologies giving a boost to availability, while reducing overall costs.

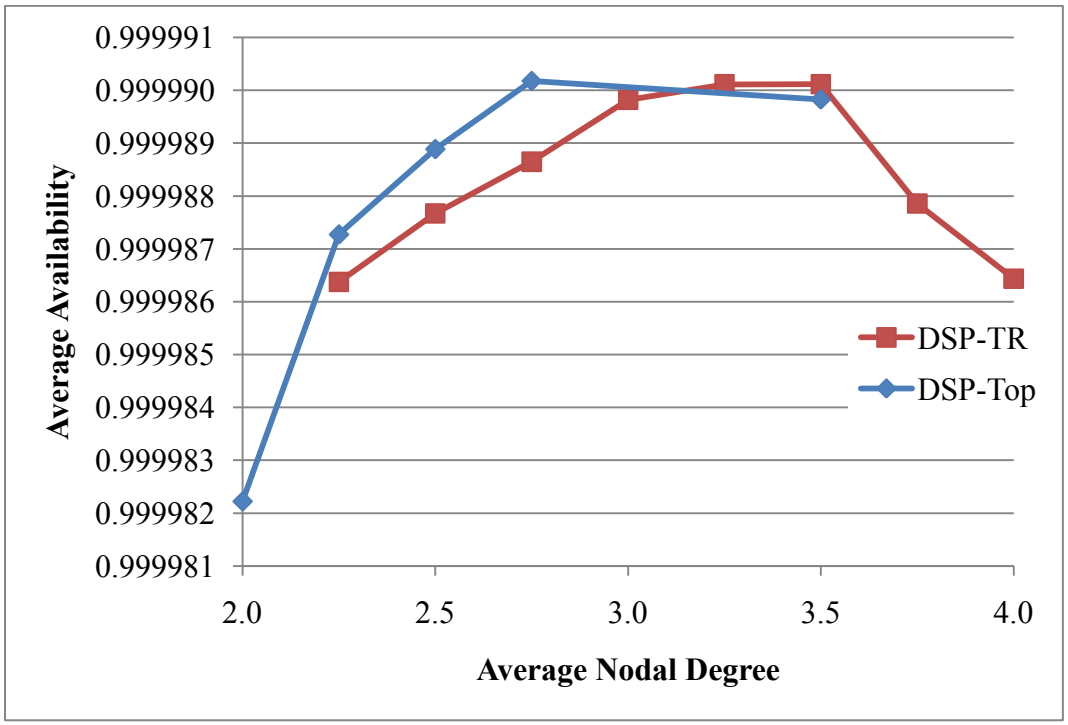


Figure 41 – Availability of DSP-TR and DSP-Top designs for 8 node networks

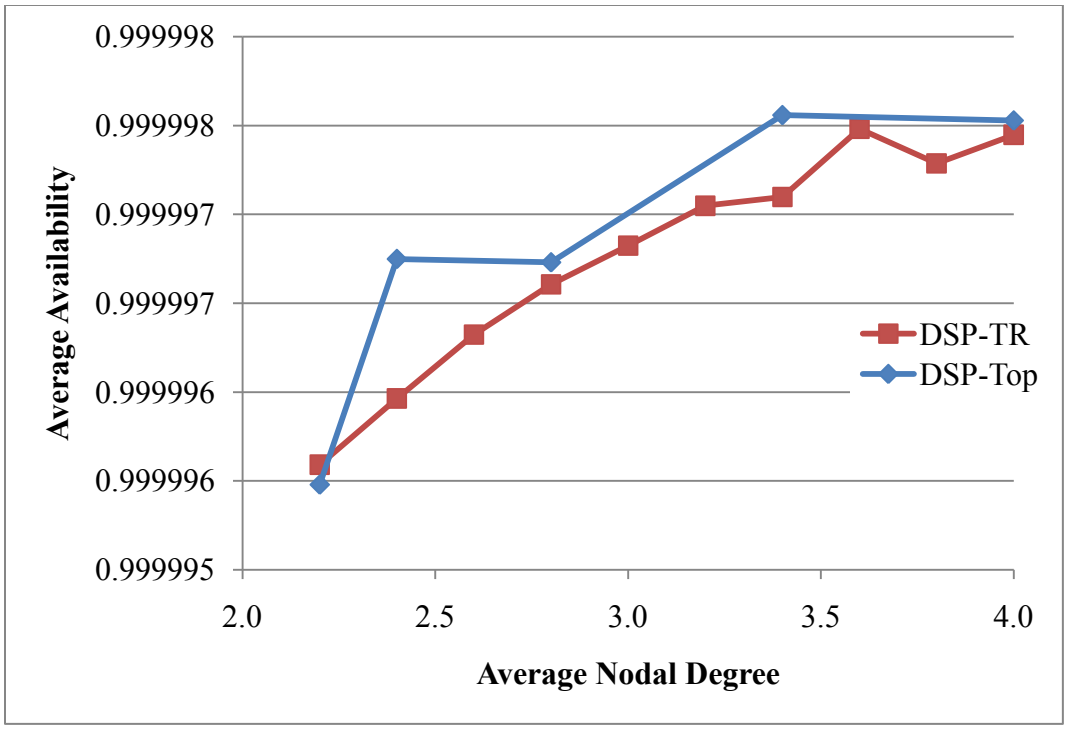


Figure 42 – Availability of DSP-TR and DSP-Top designs for 10 node networks

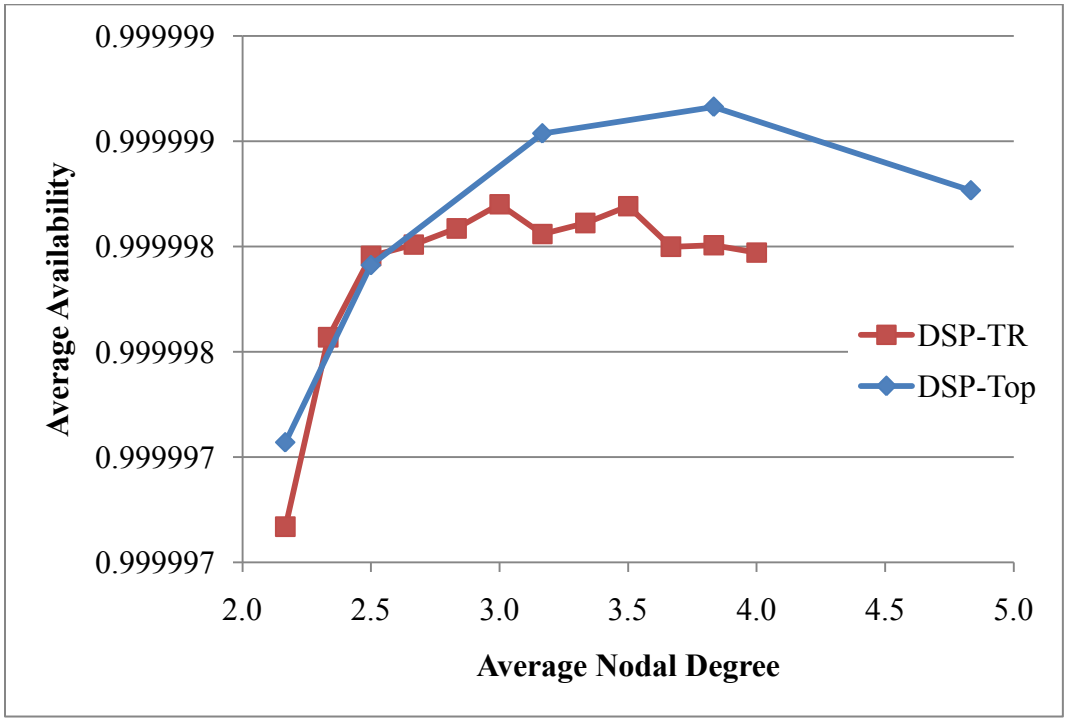


Figure 43 – Availability of DSP-TR and DSP-Top designs for 12 node networks

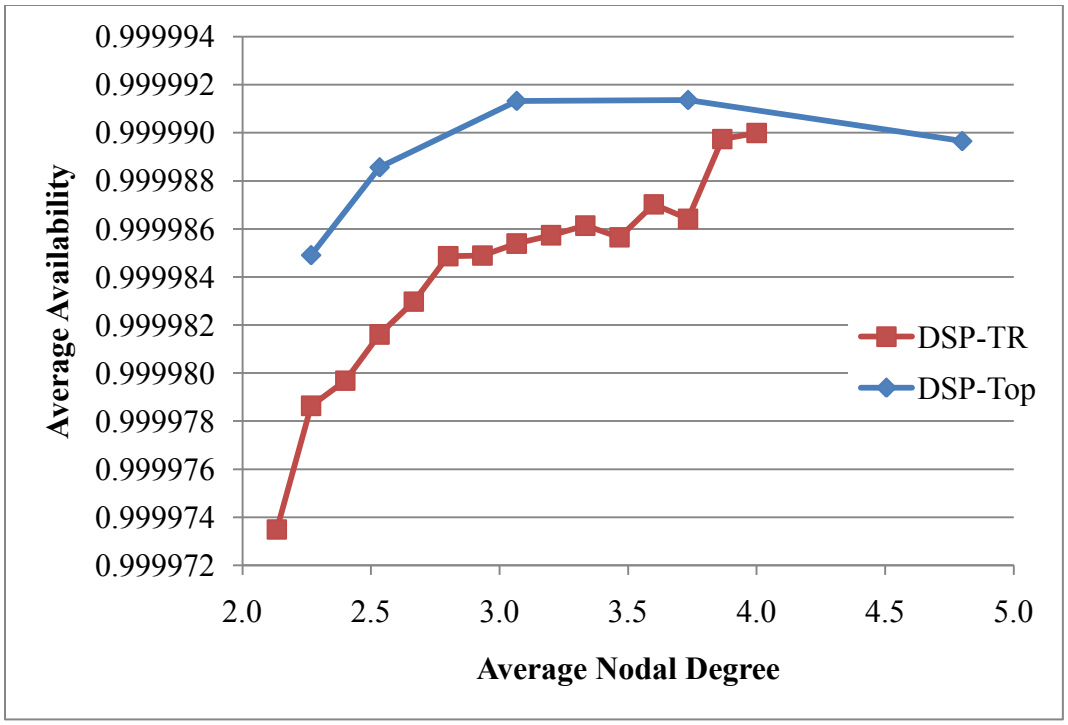


Figure 44 – Availability of DSP-TR and DSP-Top designs for 15 node networks

5.3 DSP-TR-R2

The purpose of the DSP-TR-R2 ILP was to design a network for a predefined minimum dual failure restorability (R2). This ILP was run over the same networks as where used for DSP-TR inputs, with each having a range of required R2 levels. R2 levels ranging from 50% to 100% were used. The range was limited to 50% and higher, because the point of the study was to examine high availability networks, and as will be seen, dual failure restorability levels below 0.5 adversely affect availability. Looking at the networks with an average nodal degree of 4.0, the average R2 for the demands were between 22-38%. Therefore an R2 of 50% was chosen as a starting point for next generation networks that required significantly more availability.

In DSP, to enable a demand to withstand more than n failures, there must be n+1 disjoint paths. As such, many of the test network demands do not have adequate

connectivity to allow more than single failure restorability, especially in sparse networks. When it was not possible to achieve any level of dual failure restorability for a given demand, then the R2 was set to 0. Figure 45 through Figure 48 show how many demands in each network were able to meet the R2 requirements.

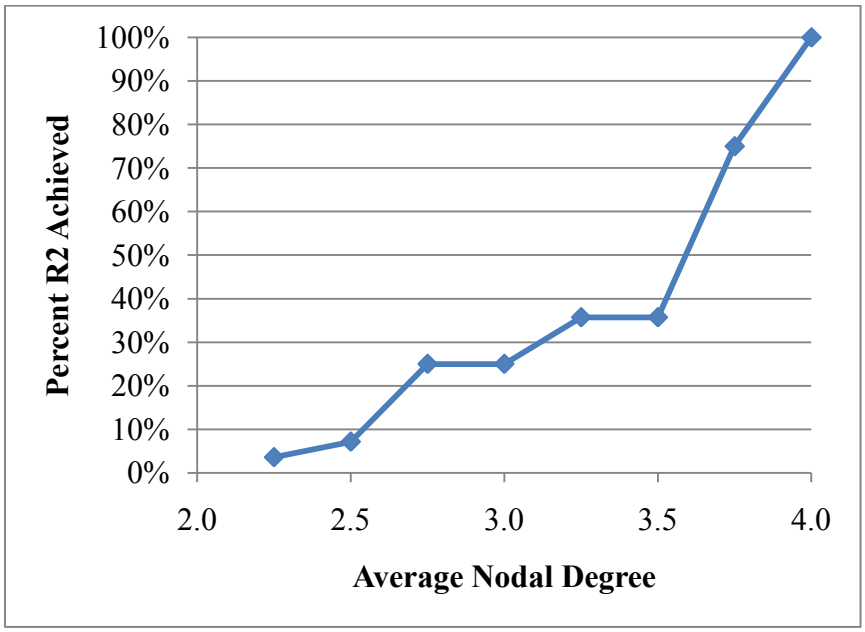


Figure 45 – Percentage of demands which met R2 requirements for the 8 node network family

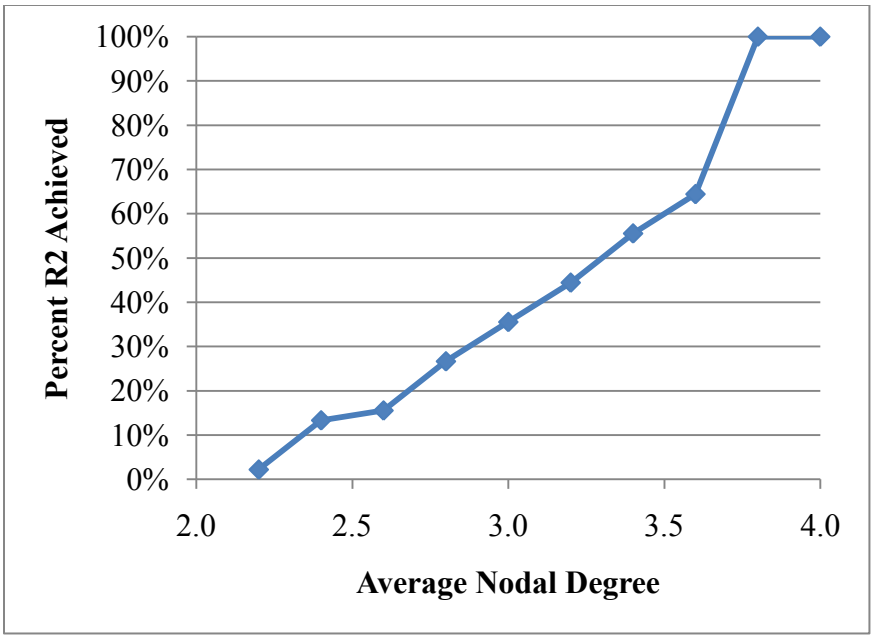


Figure 46 – Percentage of demands which met R2 requirements for the 10 node network family

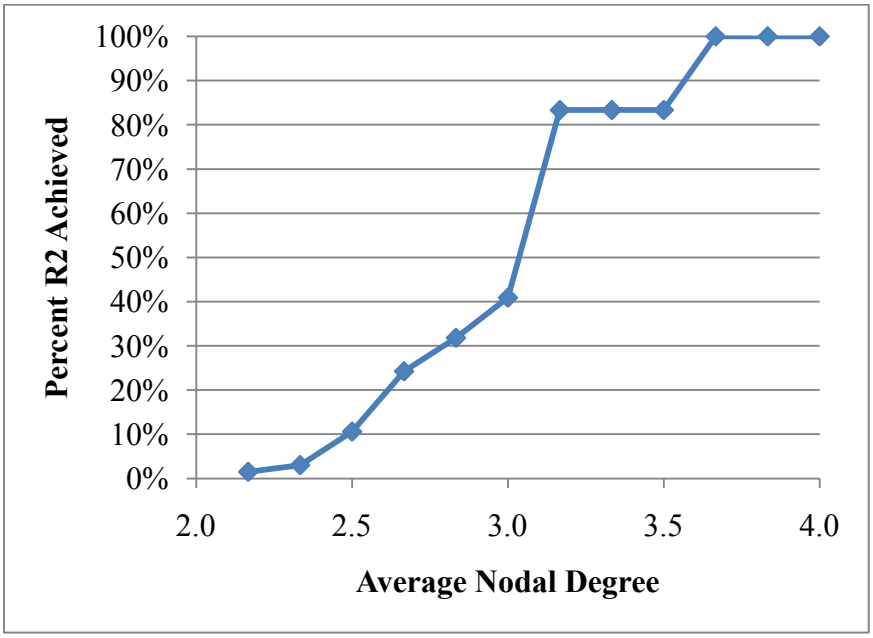


Figure 47 – Percentage of demands which met R2 requirements for the 12 node network family

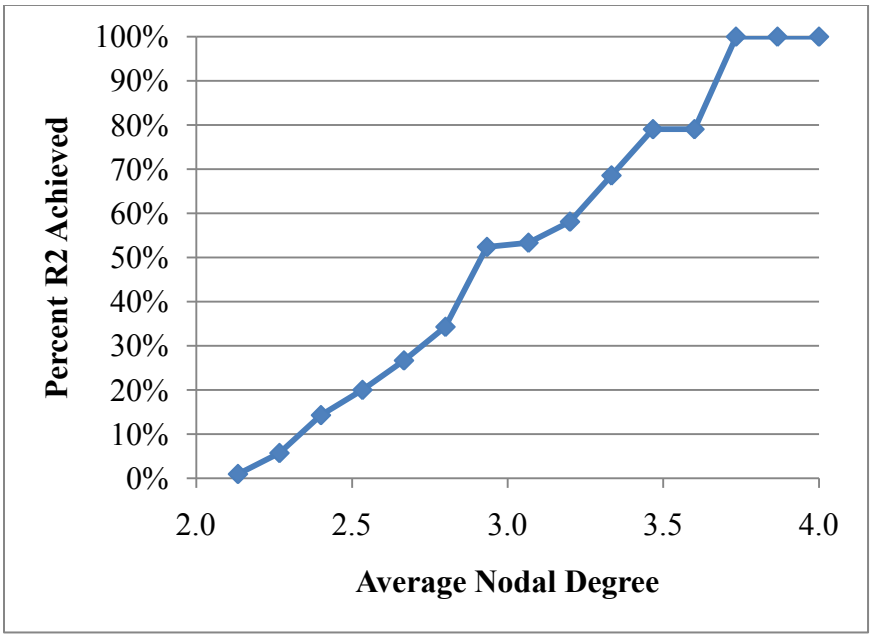


Figure 48 – Percentage of demands which met R2 requirements for the 15 node network family

The costs of the dual failure restorable networks are presented in Figure 49 through Figure 52. The designs for a required R2 of 0.9 and 1.0 had very similar capacity costs. When looking at the actual R2 of the demands, most of the demands have an R2 of 1.0 when the requirement was only 0.9. In the 15 node 29 span network, 98 out of 105 demands had an R2 of 1.0, when the required R2 was 0.9. By comparison, the design for a required R2 of 0.8 on the same network had only 49 out of the 105 demands exceeding an R2 of 0.9. For a generic example, if a demand had 3 disjoint paths connecting the origin and destination and a volume of 6 units of capacity, allocating 5 units per path would be adequate for a required R2 of 0.8, but 6 units per path would be required for a required R2 of 0.9, which gives the demand an actual R2 of 1.0. For other increments of required R2, besides between 0.9 and 1.0, the number of demands over achieving in terms of R2 is balanced out by those that are between the required R2 and the next increment. It therefore appears that the similarity in the cost between the 0.9 and 1.0 R2 designs is due to the routing of only whole units of capacity, and the fact that the maximum demands were 10 units of capacity.

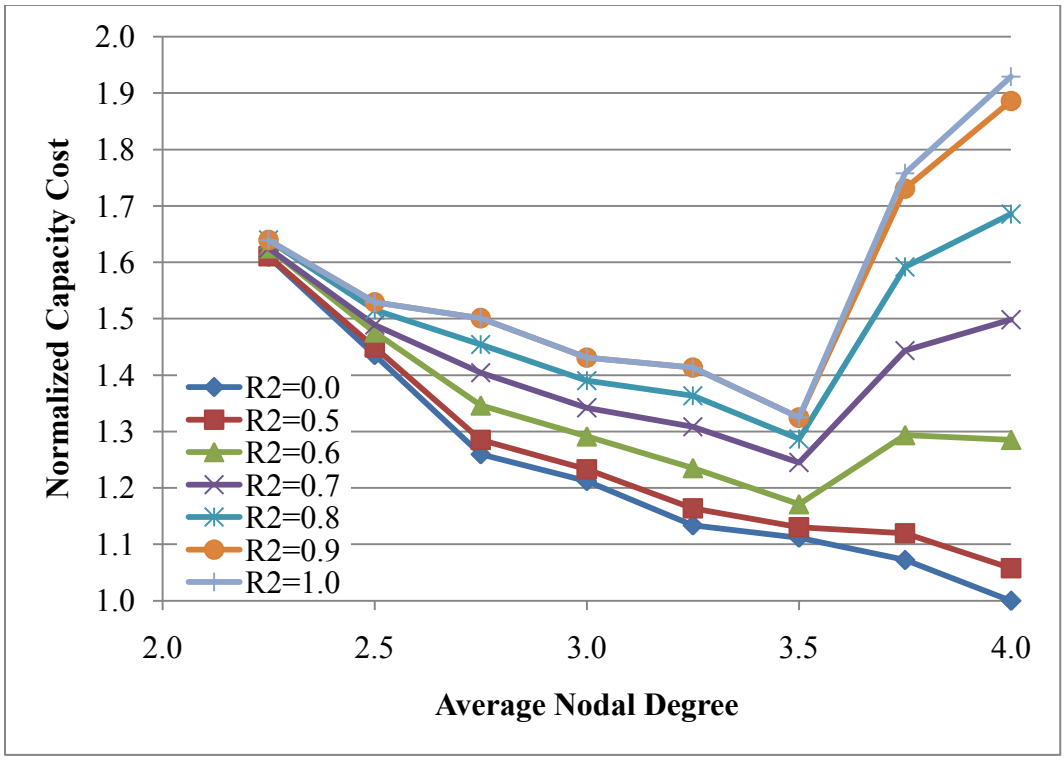


Figure 49 – Capacity cost of 8 node network family for DSP-TR and DSP-TR-R2

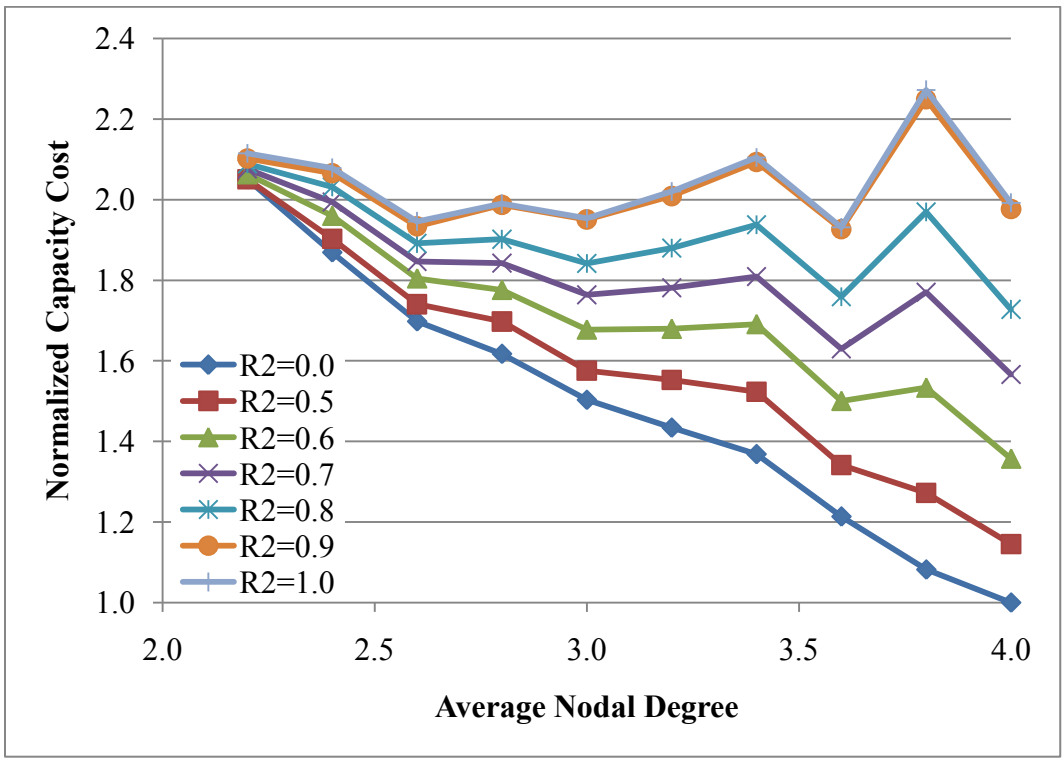


Figure 50 – Capacity cost of 10 node network family for DSP-TR and DSP-TR-R2

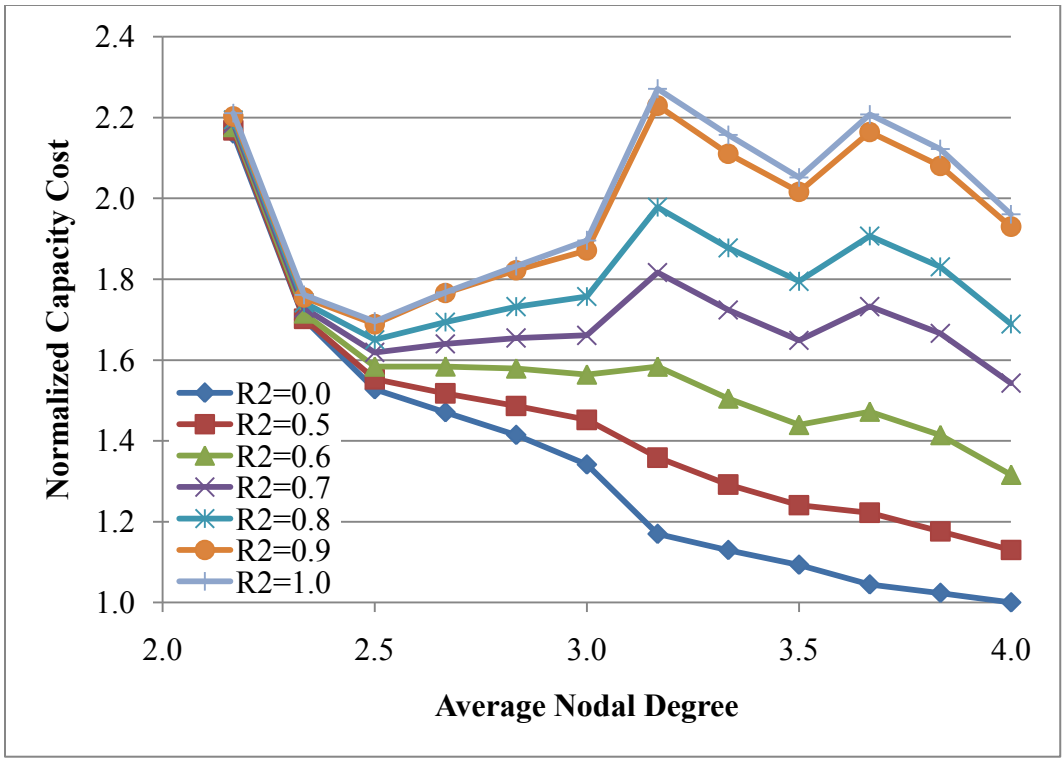


Figure 51– Capacity cost of 12 node network family for DSP-TR and DSP-TR-R2

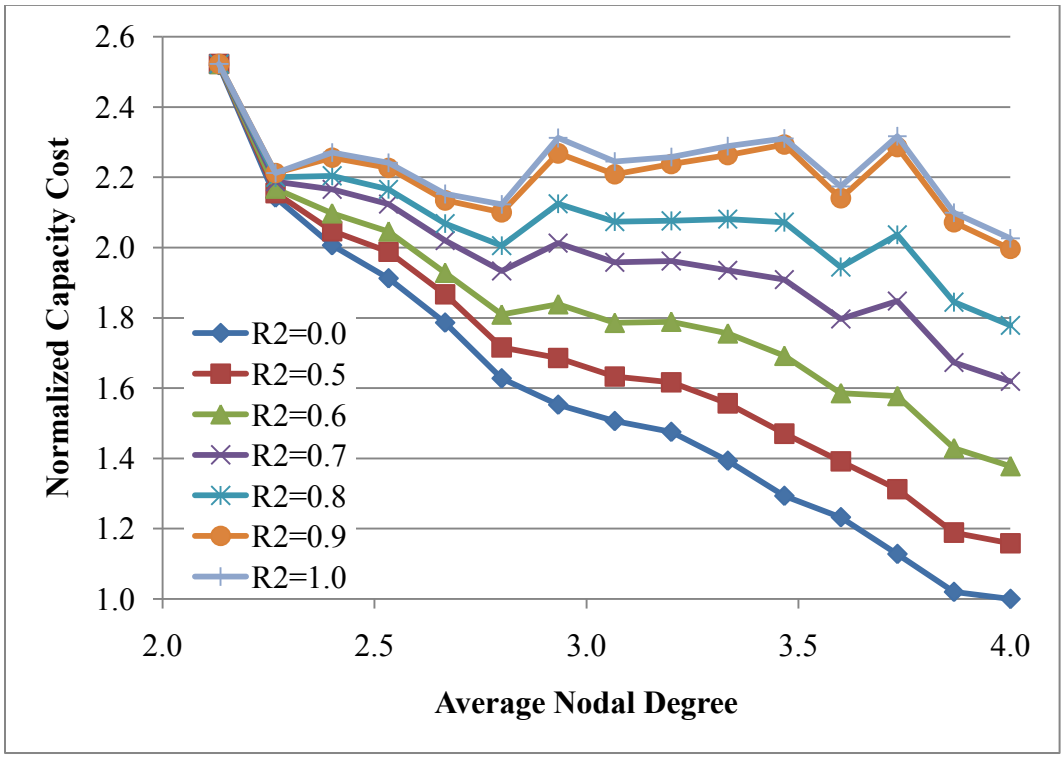


Figure 52– Capacity cost of 15 node network family for DSP-TR and DSP-TR-R2

5.3.1 Availability Analysis of DSP-TR-R2

The purpose of designing networks with predefined levels of restorability is to ultimately increase the overall availability of the demands served by the network. Figure 53 compares the average availability of each network family for the DSP-TR and the DSP-TR-R2 networks. Each network actually experienced a drop in overall availability when designed for a minimum R2 of 0.5, even though the total cost of the networks increased. The increased exposure to failure is not offset by the addition of capacity in order to meet the dual failure requirements. Demands of an even number of units of volume will already have a dual failure of 0.5 when using 3 paths and an odd number of paths only need a single unit more capacity on one of its paths to meet that requirement. As seen in Figure 20, availability is reduced as DSP utilizes more paths, and hence the drop in availability is observed between the DSP-TR and DSP-TR-R2 results with an R2 of 0.5.

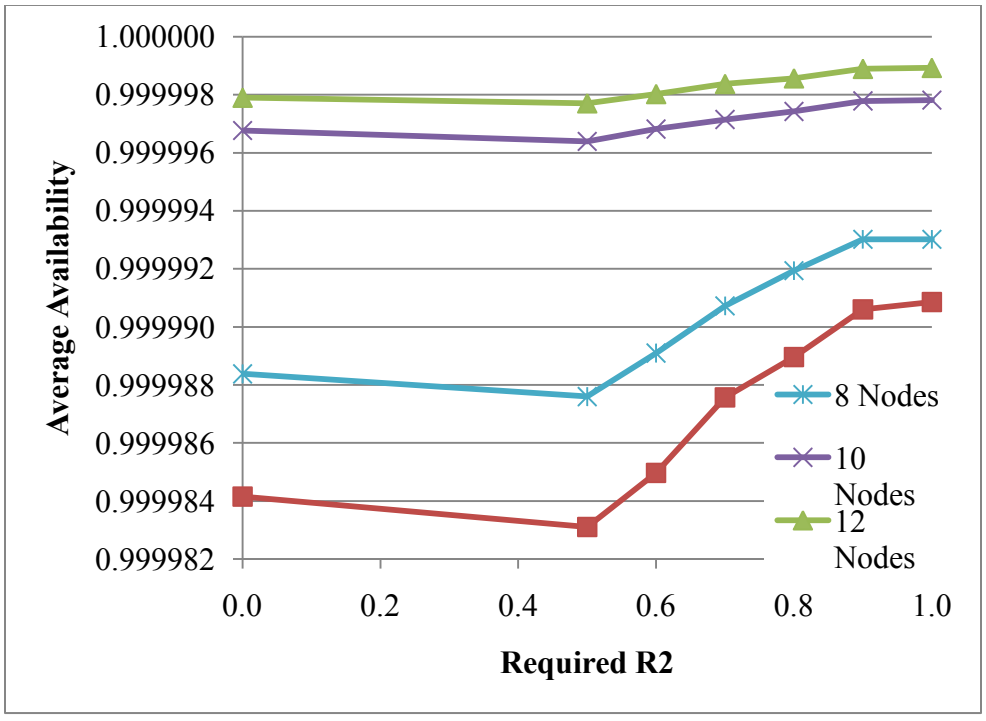


Figure 53 – Average availability of DSP-TR-R2 networks

5.4 DSP-Top-R2

The DSP-Top-R2 ILP attempts to optimize the topology as well as the placement of capacity in a network that achieves a predefined level of survivability from two failures. The DSP-Top-R2 ILP inputs utilized nodal configurations of 8, 10, 12, and 15 nodes, which were the same as what was used in the DSP-Top designs. The ILP was solved using a range implementation costs and required minimum dual failure survivability's, as outlined in Table 3.

Input Parameter	Range
Number of nodes	8, 10, 12, 15 nodes
Required R2	0.5 – 1.0 (0.1 increments)
Implementation Costs	20, 50, 100, 200, 500 times unit implementation costs

Table 3 – DSP-Top-R2 input parameters

A two-step process was also used for the DSP-Top-R2 solutions, as in the DSP-Top solutions, with first step solving for an optimized topology with the integrality of the routed capacity relaxed. The topologies resulting from this optimization were then solved using the DSP-TR-R2 ILP to route capacity in integral quantities. While this process meant that the results are not strictly optimal, it was required in order to reduce the run time and memory requirements of the solutions to sizes manageable by the available computing resources (section 4.2). These relaxed ILPs still contained integer variables as the variables representing direction of traffic flow were still required to be integer in order for the ILP results to be meaningful. If these variables had their integer requirements relaxed, the properties of path disjointedness and traffic flow from origin to destination could not be enforced.

As in the DSP-Top results, the DSP-Top-R2 solutions were time limited, and hence the results cannot be labeled as optimal. A run time of 2 hours (7200 seconds) was chosen for the 12n24s1 solutions, and 3 hours (10800 seconds) for the 15n30s1 networks. These run times were chosen after examining the best integer solutions of a sample of networks, and the time in which they appear

approach a minimal result. For each time limit, doubling the run time would only result in a minor improvement in the overall cost (less than 1%).

Network	R2	Mip gap	Time Run(s)		Network	R2	Mip gap	Time Run(s)
12n24s1-500x	0.5	0.4331	7200		12n24s1-500x	0.8	0.3777	7200
12n24s1-200x	0.5	0.30019	7200		12n24s1-200x	0.8	0.3132	7200
12n24s1-100x	0.5	0.2633	7200		12n24s1-100x	0.8	0.206	7200
12n24s1-50x	0.5	0.1887	7200		12n24s1-50x	0.8	0.1759	7200
12n24s1-20x	0.5	0.1339	7200		12n24s1-20x	0.8	0.1286	7200
12n24s1-500x	0.6	0.498	7200		12n24s1-500x	0.9	0.3579	7200
12n24s1-200x	0.6	0.1462	7200		12n24s1-200x	0.9	0.3446	7200
12n24s1-100x	0.6	0.2467	7200		12n24s1-100x	0.9	0.2425	7200
12n24s1-50x	0.6	0.1867	7200		12n24s1-50x	0.9	0.1847	7200
12n24s1-20x	0.6	0.1301	7200		12n24s1-20x	0.9	0.1392	7200
12n24s1-500x	0.7	0.3542	7200		12n24s1-500x	1	0.3136	7200
12n24s1-200x	0.7	0.3218	7200		12n24s1-200x	1	0.2497	7200
12n24s1-100x	0.7	0.2652	7200		12n24s1-100x	1	0.2448	7200
12n24s1-50x	0.7	0.2062	7200		12n24s1-50x	1	0.207	7200
12n24s1-20x	0.7	0.134	7200		12n24s1-20x	1	0.1441	7200

Table 4 – Mip gap and run time for DSP-Top-R2 solutions on 12n24s1 networks

Network	R2	Mip gap	Time Run(s)		Network	R2	Mip gap	Time Run(s)
15n30s1-500x	0.5	0.3866	10800		15n30s1-500x	0.8	0.3731	10800
15n30s1-200x	0.5	0.3642	10800		15n30s1-200x	0.8	0.3658	10800
15n30s1-100x	0.5	0.3264	10800		15n30s1-100x	0.8	0.3272	10800
15n30s1-50x	0.5	0.2819	10800		15n30s1-50x	0.8	0.2802	10800
15n30s1-20x	0.5	0.2028	10800		15n30s1-20x	0.8	0.2003	10800
15n30s1-500x	0.6	0.376	10800		15n30s1-500x	0.9	0.3877	10800
15n30s1-200x	0.6	0.3648	10800		15n30s1-200x	0.9	0.3669	10800
15n30s1-100x	0.6	0.3254	10800		15n30s1-100x	0.9	0.3302	10800
15n30s1-50x	0.6	0.2816	10800		15n30s1-50x	0.9	0.2679	10800
15n30s1-20x	0.6	0.2058	10800		15n30s1-20x	0.9	0.2064	10800
15n30s1-500x	0.7	0.3963	10800		15n30s1-500x	1	0.3834	10800
15n30s1-200x	0.7	0.3675	10800		15n30s1-200x	1	0.3596	10800
15n30s1-100x	0.7	0.3287	10800		15n30s1-100x	1	0.3332	10800
15n30s1-50x	0.7	0.2715	10800		15n30s1-50x	1	0.2845	10800
15n30s1-20x	0.7	0.1982	10800		15n30s1-20x	1	0.2081	10800

Table 5– Mip gap and run time for DSP-Top-R2 solutions on 15n30s1 networks

The 8 node and 10 node networks were able to solve with a mip gap of 0.001 within reasonable amounts of time (less than 1 hour per network). There is a noticeable difference between the networks that solved optimally and the ones that didn't, as is displayed in the quality of the results. It would be expected that the average nodal degree for networks with the same implementation cost would only increase or stay the same as the required R2 is increased. Looking at Figure 54 through Figure 57 it is apparent that the two networks that were able to solve optimally in the first stage of the solution process do indeed follow this property, with the 8 node networks designed for an R2 of 0.9 and 1.0 with an implementation cost of 20 were the only networks to increase the number of spans in the network. This is what is expected, as the lower implementation cost makes the tradeoff between path length and cost savings more likely to encourage larger

networks, and with high R2, there is more of an advantage of utilizing more paths, as the total required capacity for each demand is increased to accommodate multiple failure restoration.

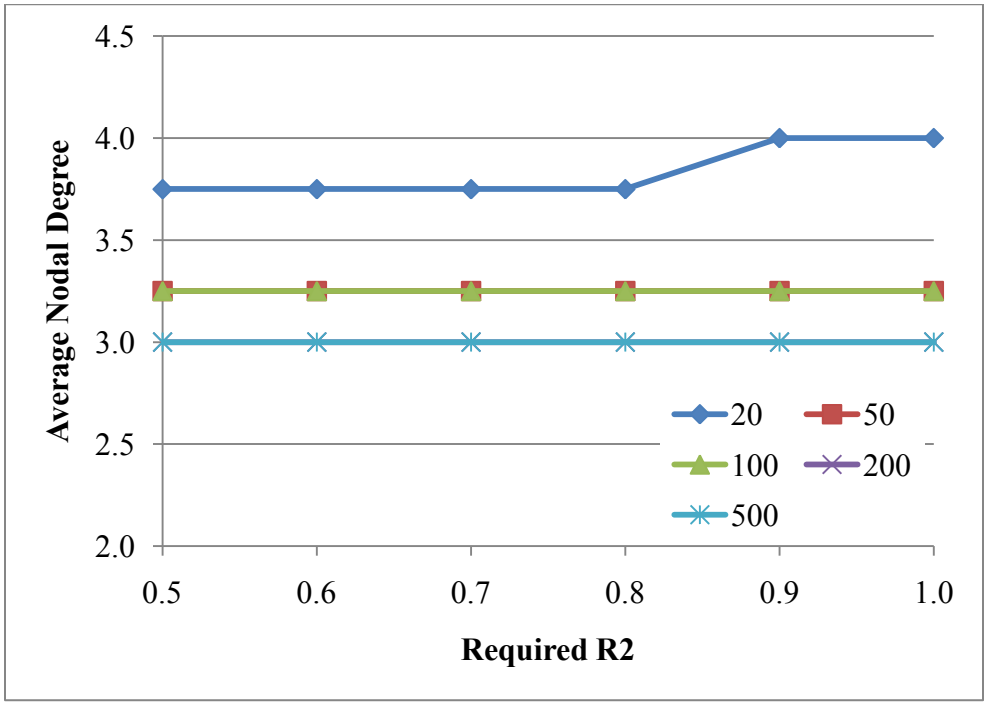


Figure 54 – Average nodal degree for the DSP-Top-R2 ILP results from the 8 node base network

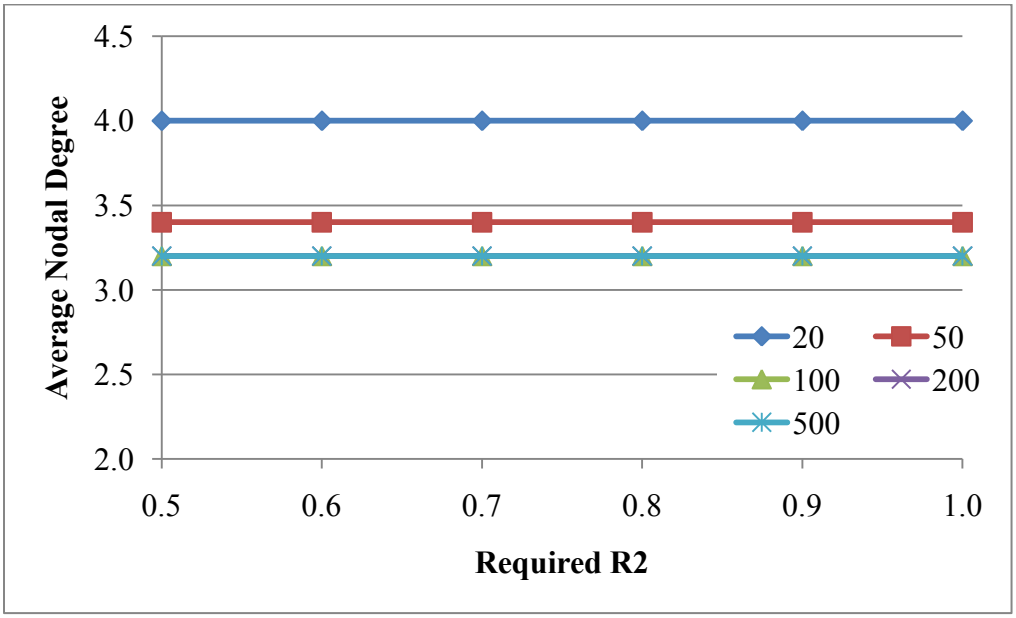


Figure 55 – Average nodal degree for the DSP-Top-R2 ILP results from the 10 node base network

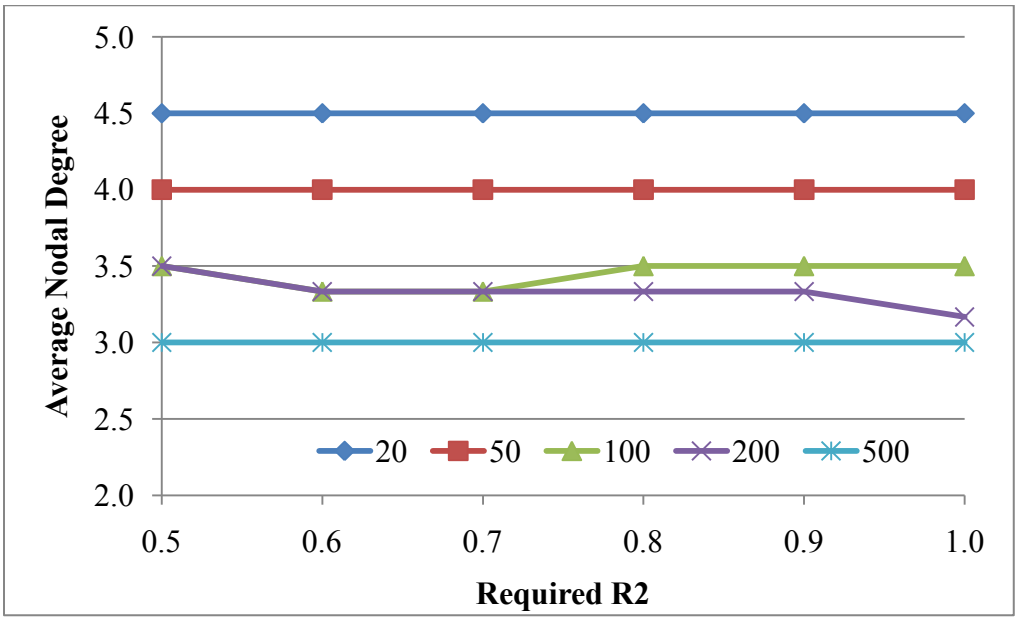


Figure 56 – Average nodal degree for the DSP-Top-R2 ILP results from the 12 node base network

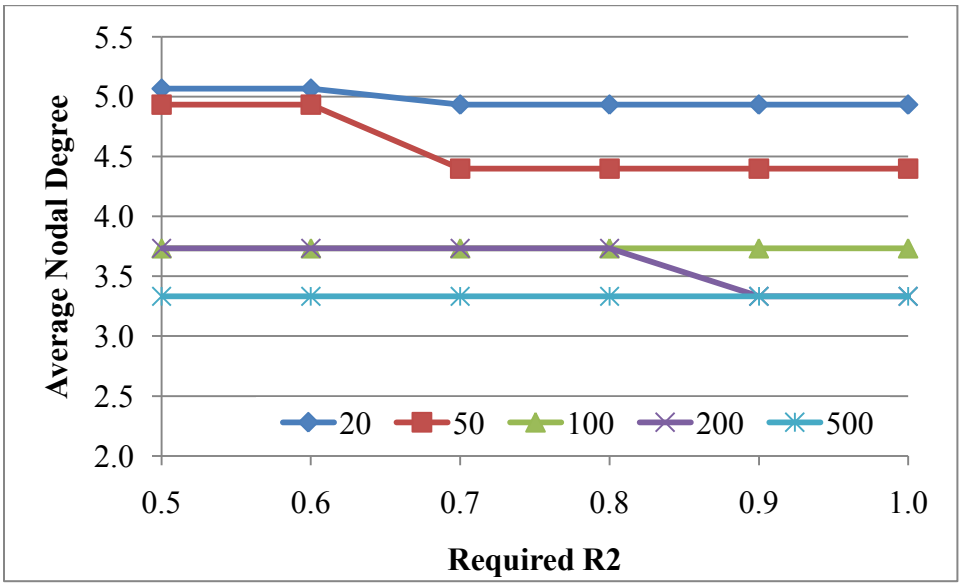


Figure 57 – Average nodal degree for the DSP-Top-R2 ILP results from the 15 node base network

The networks that were not able to solve optimally do have networks that get smaller as the required R2 increases. Since this runs contrary to the theory that networks should be adding spans as the R2 increases, the networks used in the 12

node test cases for an implementation factor of 100 and 200 were rerun. The networks used for these re-tests were the networks used of other required R2 values, with the purpose of investigating whether some networks would provide better results for other required R2 values than what they were designed for. If the above theory holds, it would be expected that some of the results of the original solutions would be found to be a higher cost.

After the re-tests were done, the lowest cost candidate network for each required R2 was chosen. The average nodal degree from these lowest cost sets are presented in Figure 58. These indeed follow the trend of networks not decreasing in average nodal degree as the required R2 is increased. The reason that optimal topology designs cannot be reduced in average nodal degree as the required R2 increases is found in the fact that DSP only has more incentive to utilize more paths as the capacity that is routed increases. As such, it would not make sense to reduce network size, as this would have the effect of reducing the number of paths each demand could utilize.

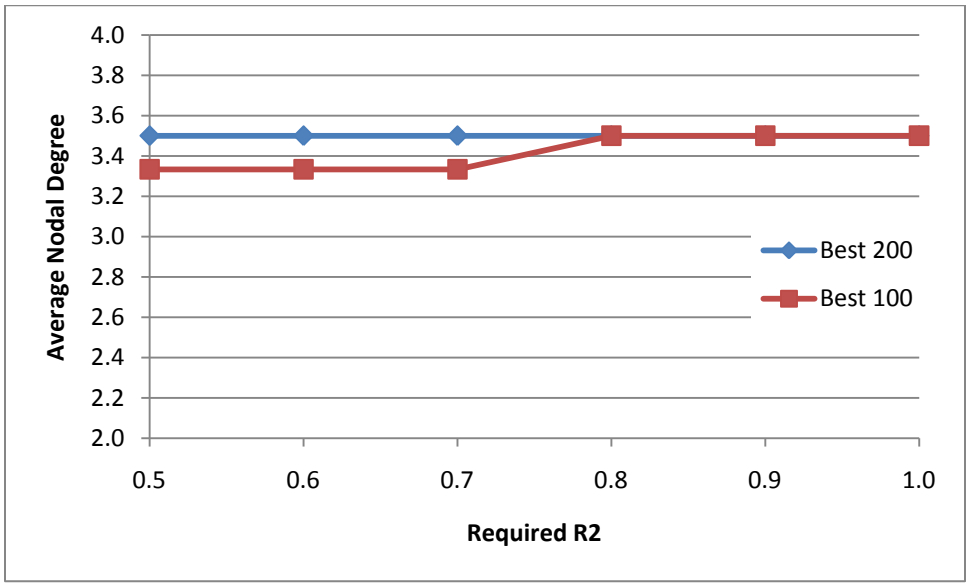


Figure 58 – Average nodal degree of the lowest cost networks including re-tests with topologies designed for other required R2 levels

The 12 node and 15 node results are sub-optimal; however, the results are reasonable, in terms of the relative average nodal degree of the networks

compared to the 8 and 10 node results and with relation to other implementation factors. These original results will be used in the analysis of characteristics of dual failure restorable networks with topology design.

5.4.1 Costs Comparisons of DSP-Top-R2

The cost of the network designs from the DSP-Top-R2 ILP can be viewed to observe the effect of the implementation factor or the increasing R2 requirements. When comparing results across various implementation factors, it is the effect of the implementation factor that dominates the total cost, partially masking the effect of topology and dual failure restorability has on the total cost. Figure 59 through Figure 62 show the linear effect of the implementation factor on the total cost. The impact of topology and dual failure restorability can be more easily observed in the capacity costs.

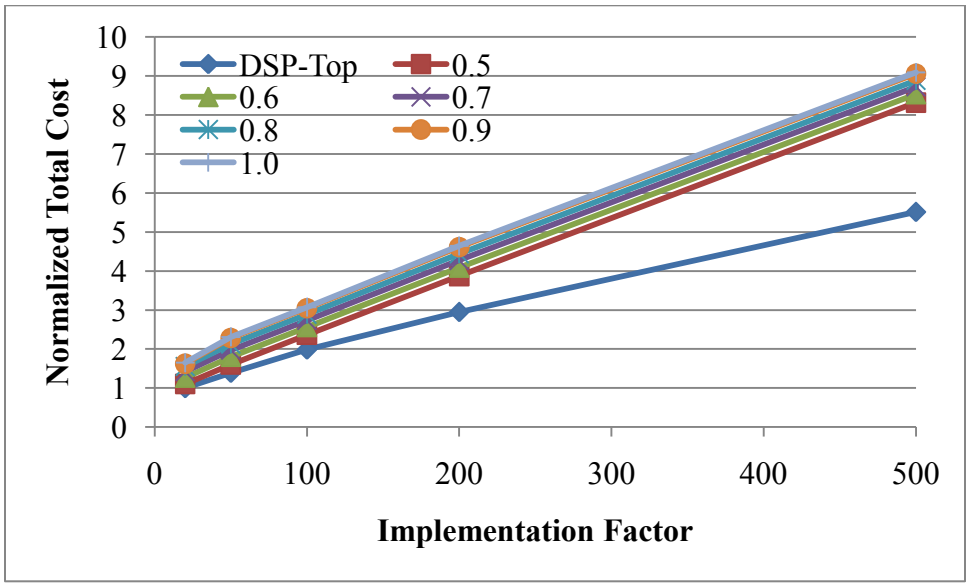


Figure 59 – Total cost of the 8 node DSP-Top-R2 network designs

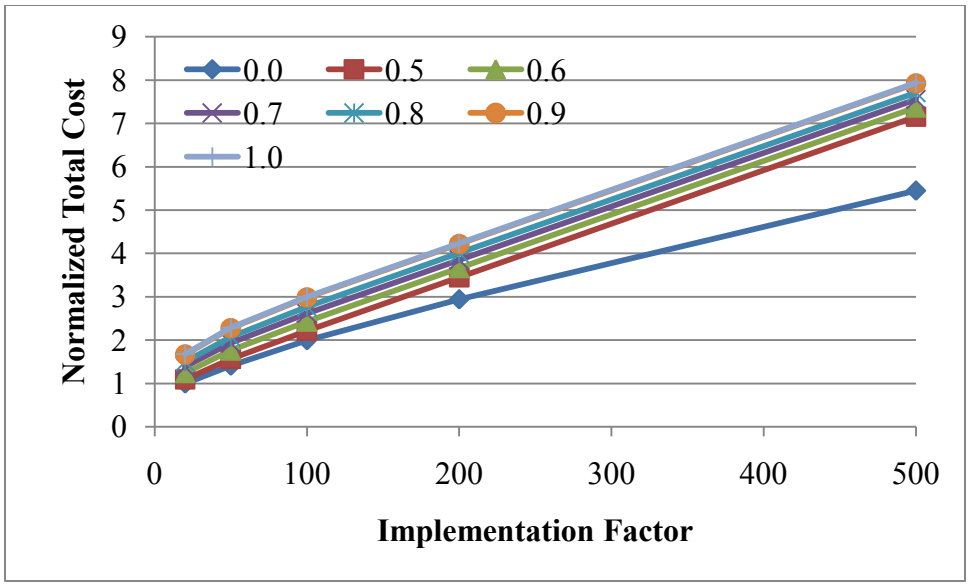


Figure 60 – Total cost of 10 node DSP-Top-R2 network designs

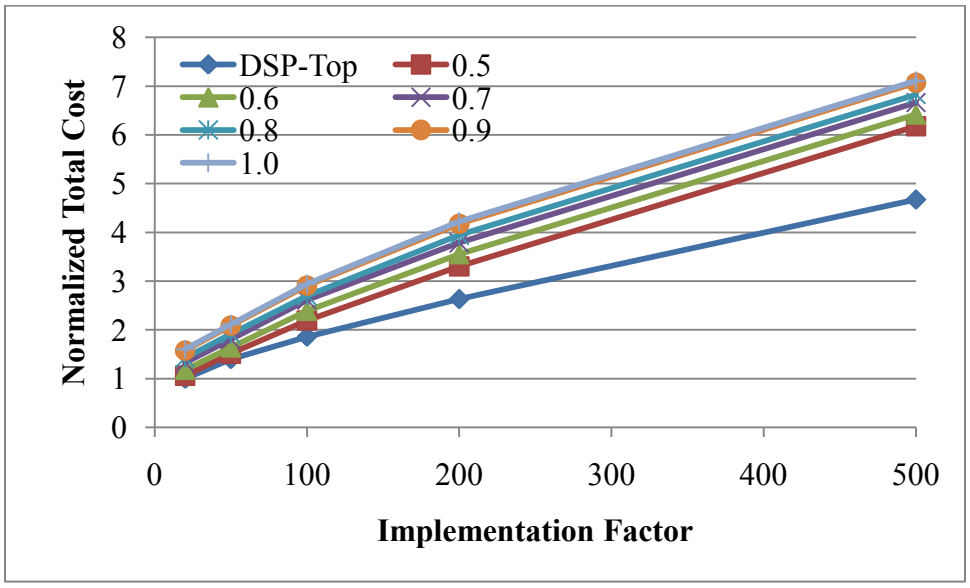


Figure 61 – Total cost of 12 node DSP-Top-R2 network designs

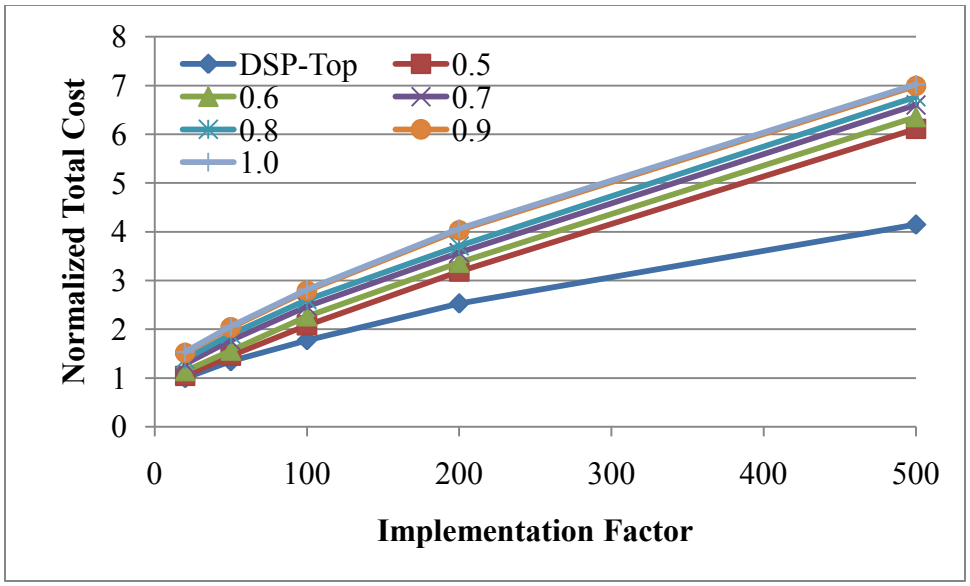


Figure 62 – Total cost of 15 node DSP-Top-R2 network designs

Removing the implementation costs from the total costs highlights the effect of the implementation factor on the cost of the networks. The capacity cost graphs (Figure 63 through Figure 66) highlight the consistency in topology between various required R2 values, as well as the limited savings associated with DSP with dual failure restorability, as the networks have a consistent capacity cost generally for networks with an implementation factor of 200 and 500. The differences between how the costs are affected by the implementation factor are minimal. Each R2 holds a similar shape, with a difference in a scaling factor. This indicates that topology is affected more by the implementation factor, and very little by the required R2.

In Figure 65 the capacity cost for the network designed with an implementation factor of 200 and an R2 of 0.5 is less than the capacity cost of the network with an implementation factor of 100. This is another indication that the network designs for the 12 node network configurations are not optimal. In general, though, the capacity costs behave as expected.

The capacity costs for the networks designed for an R2 of 0.9 are very similar to the networks designed for an R2 of 1.0, similar to what was found with the DSP-Top results. As with DSP-Top, in order to meet the minimum requirements of an

R2 of 0.9, most demands had to have enough capacity added to produce fully dual failure protected network design.

It appears that topologies reach a limit in their possible efficiency gains vs. the number of spans in the network around the implementation factor of 200. Most capacity costs for networks designed with an implementation factor of 200 are similar to networks designed with an implementation factor of 500. This is because networks designed for an implementation factor of 200 were already nearing the average nodal degree of 3.0, which is a minimum for dual failure restorable networks. The DSP-Top results were still able to reduce the number of spans in the network between the networks with an implementation factor of 200 and 500.

When DSP is used to protect against dual failures there is a change in the balance between the cost of adding more paths, and the cost reduction available. In most networks that are relatively sparse in terms of nodes per total area, paths are forced to be routed significantly out of the way compared to a straight path between two nodes. The implications of this is that although the efficiency gains of utilizing more paths, in terms of lightpath count, remain the same, the ability to route more paths with relatively low cost differences is reduced. Hence, the networks approach the minimum average nodal degree at lower implementation factors for DSP-Top-R2 networks, than when compared to DSP-Top networks.

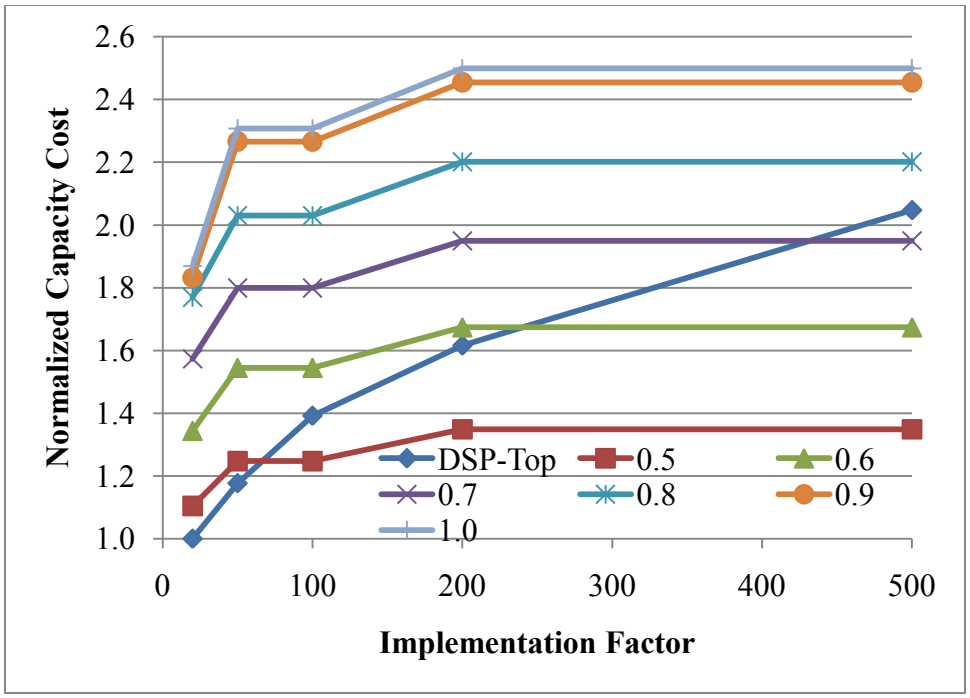


Figure 63 – Capacity cost of 8 node DSP-Top-R2 network designs

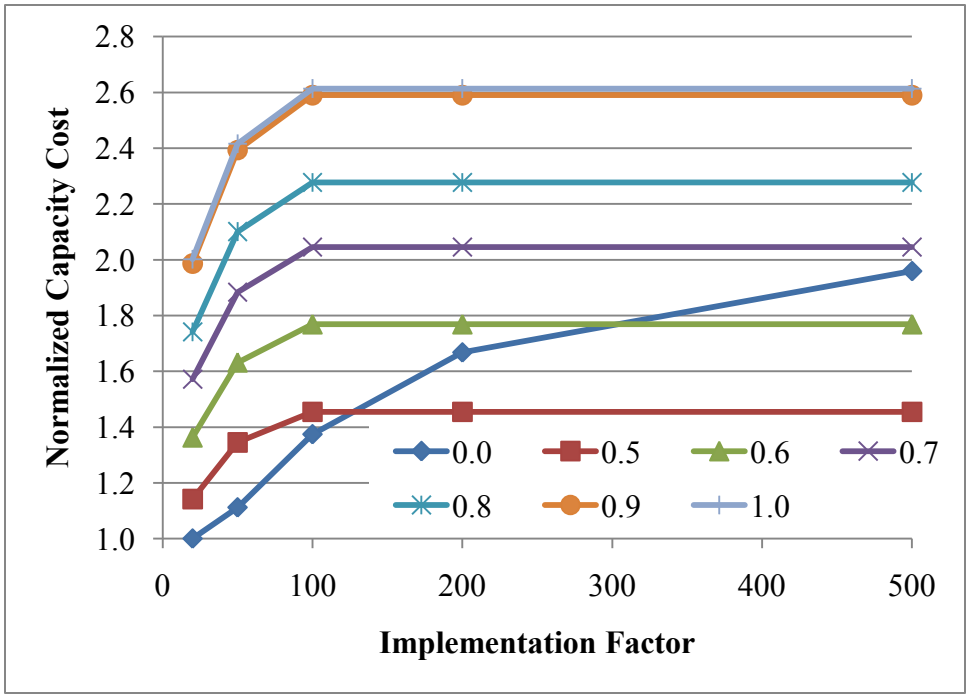


Figure 64 – Capacity cost of 10 node DSP-Top-R2 network designs

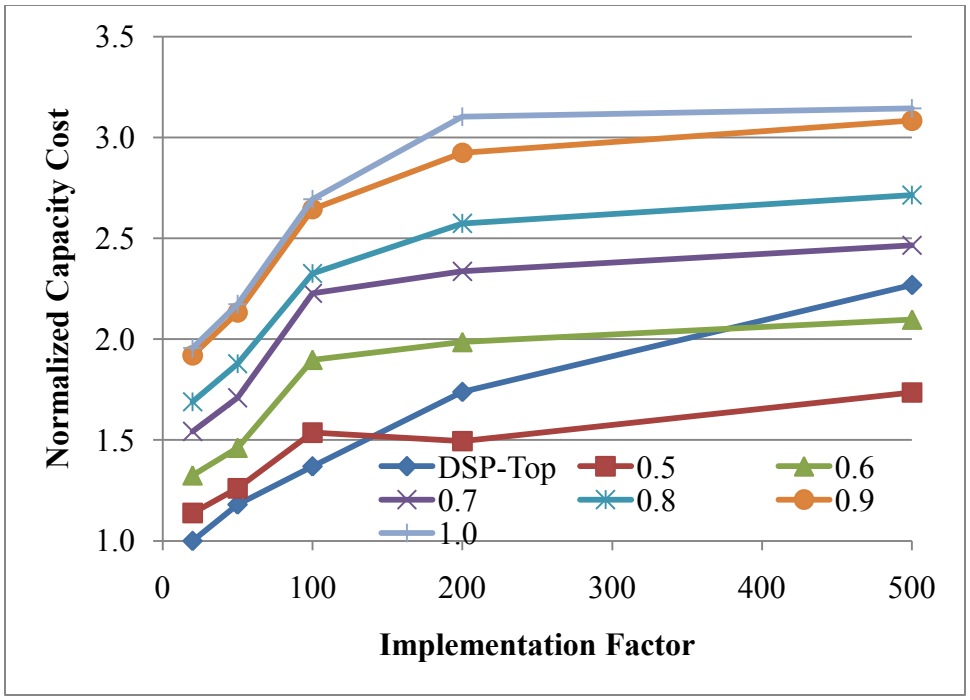


Figure 65 – Capacity cost of 12 node DSP-Top-R2 network designs

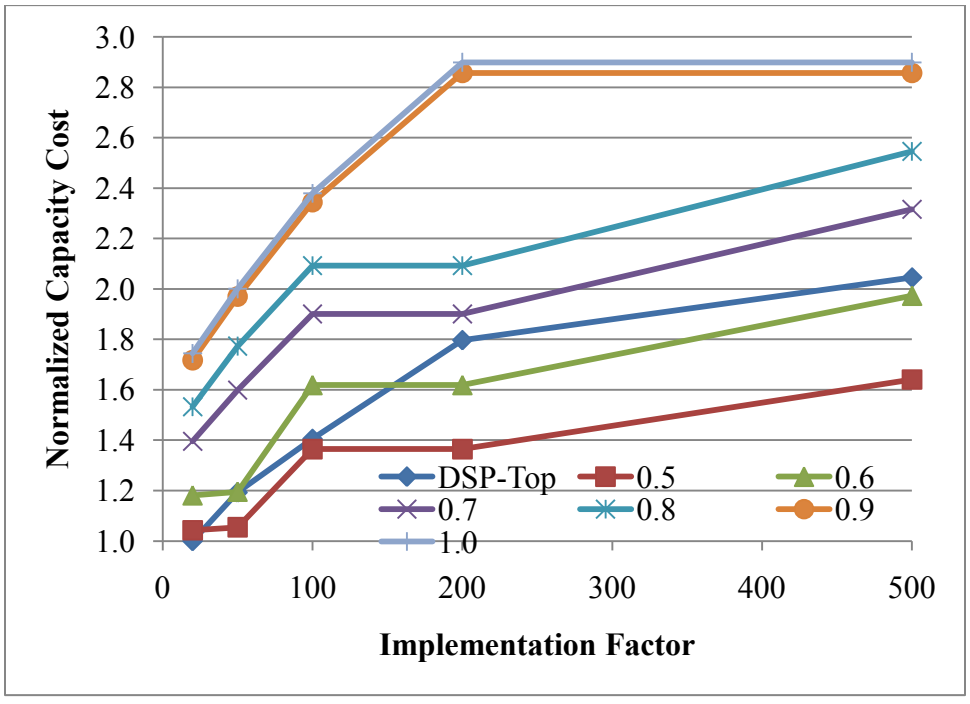


Figure 66 – Capacity cost of 15 node DSP-Top-R2 network designs

The dual failure restorable network designs are affected by both the required dual failure restorability and the implementation costs. At minimum all nodes in the network topologies produced by the DSP-Top-R2 ILP must be tri-connected.

The resulting topologies from the 12 node base network show some signs that the resulting designs could be further optimized. In Figure 68 one span is assigned that nearly runs over top of two other spans for its length. Upon initial inspection it would appear that this span is redundant within this network. Removing this span altogether, however, produced a total cost that was higher than with the span. While this may be counter intuitive, it does show that the results, although not optimal, are reasonable. In the topologies for other R2 requirements the span count remained the same but the topology was modified as so there were no spans that ran nearly identical routes as other spans in the network. This topology was used in a re-test of the 0.5 R2 requirement, and indeed produced a total cost that was 1% lower than the results using the network presented in Figure 68. This again is a result of the 12 node and 15 node networks not being able to solve optimally with available computer resources.

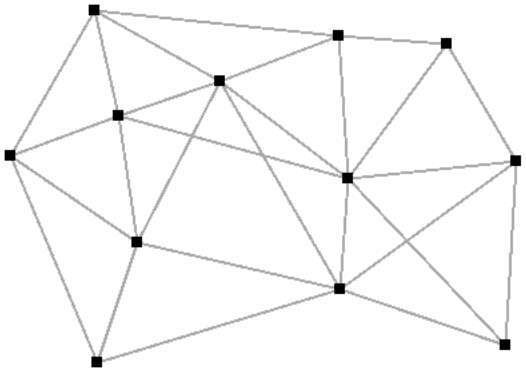


Figure 67 – DSP-Top-R2 12 node network design with an implementation factor of 20 and a required R2 of 0.5

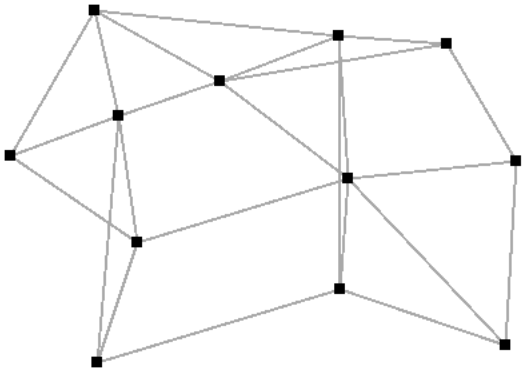


Figure 68 – DSP-Top-R2 12 node network design with an implementation factor of 50 and a required R2 of 0.5

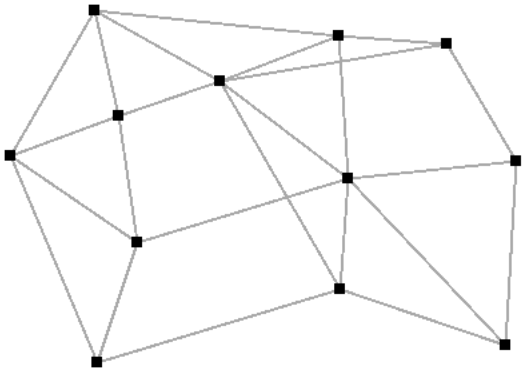


Figure 69 – DSP-Top-R2 12 node network design with an implementation factor of 50 and a required R2 of 0.6

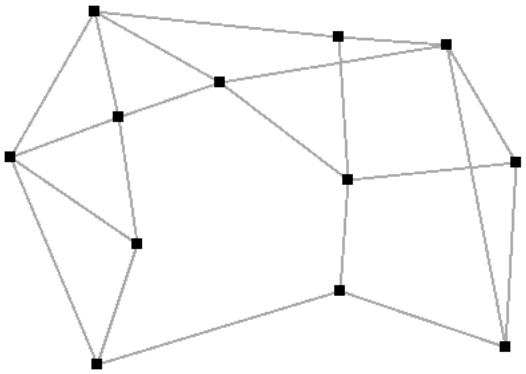


Figure 70 – DSP-Top-R2 12 node network design with an implementation factor of 100 and a required R2 of 0.5

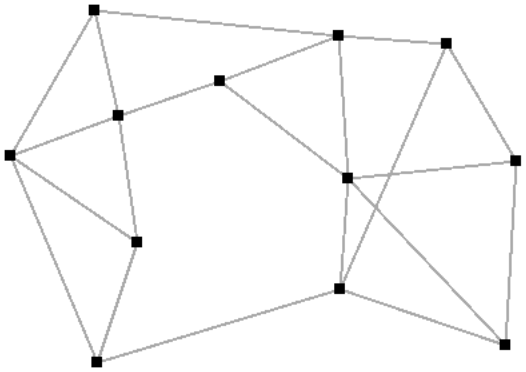


Figure 71 – DSP-Top-R2 12 node network design with an implementation factor of 200 and a required R2 of 0.5

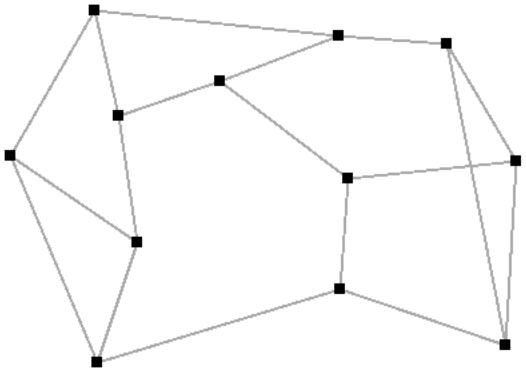


Figure 72 – DSP-Top-R2 12 node network design with an implementation factor of 500 and a required R2 of 0.5

5.4.1.1 Costs Comparisons of DSP-Top-R2 with DSP-TR-R2

Comparing the costs of the DSP-Top-R2 ILP results to the DSP-TR-R2 results is not straight-forward (Figure 73 through Figure 76). The majority of the network family topologies are not fully tri-connected, and therefore some demands were not able to withstand dual failures under any circumstances. This meant that the capacity costs were reduced because these demands did not get routed with the extra capacity required for dual failure restorability, and the topology results could be higher than the network family results for a given average nodal degree. When looking at the total cost of the network designs for a significant

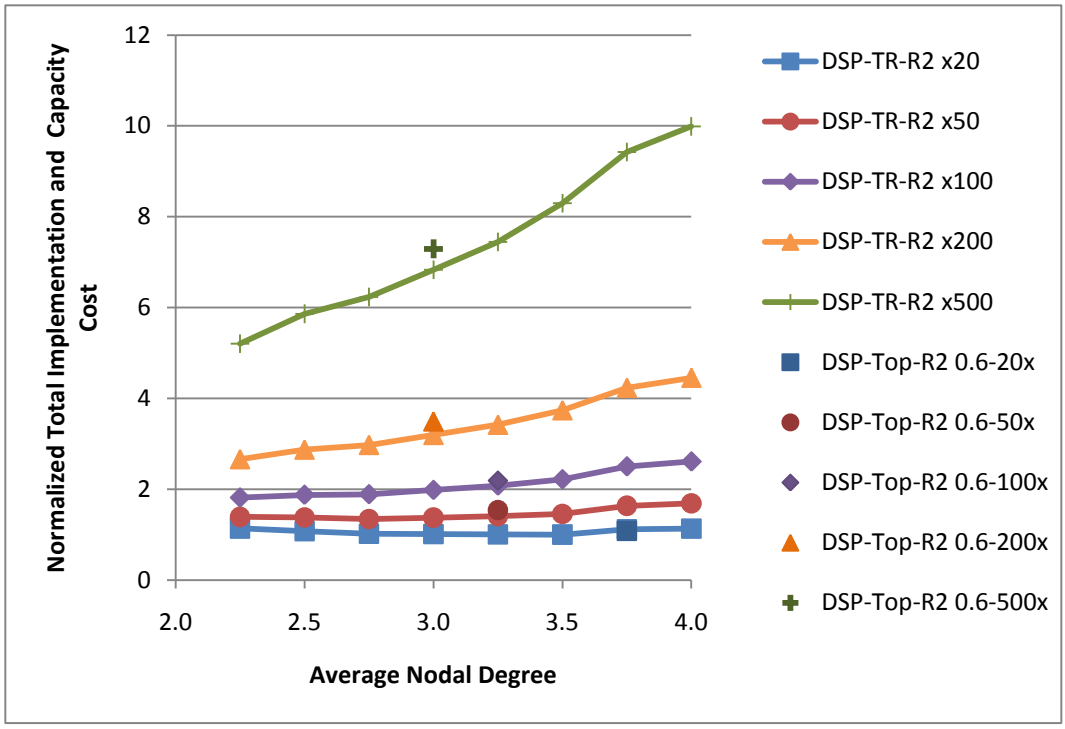


Figure 73 – DSP-Top-R2 8 node network design with a required R2 of 0.6

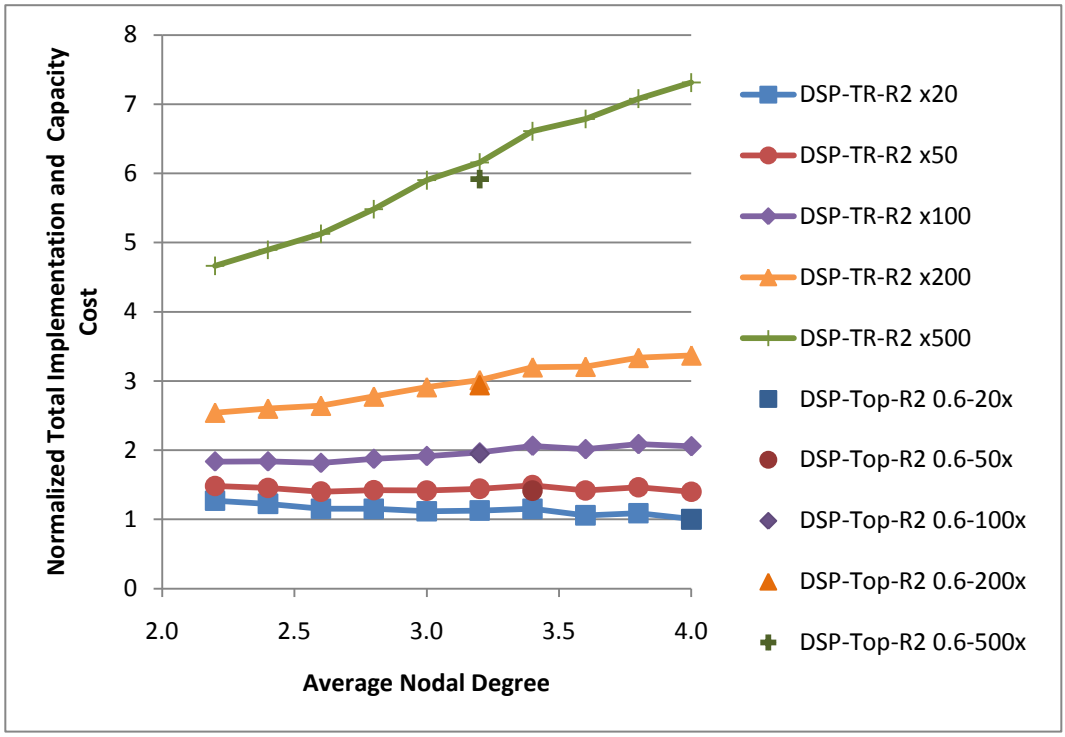


Figure 74 – DSP-Top-R2 10 node network design with a required R2 of 0.6

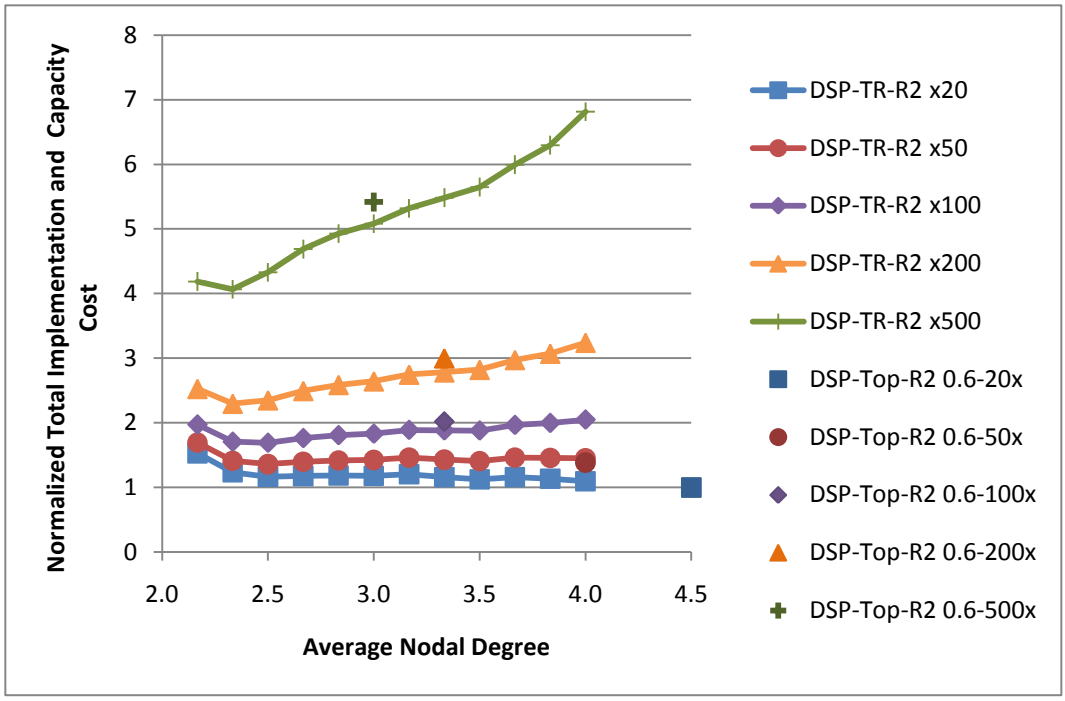


Figure 75 – DSP-Top-R2 12 node network design with a required R2 of 0.6

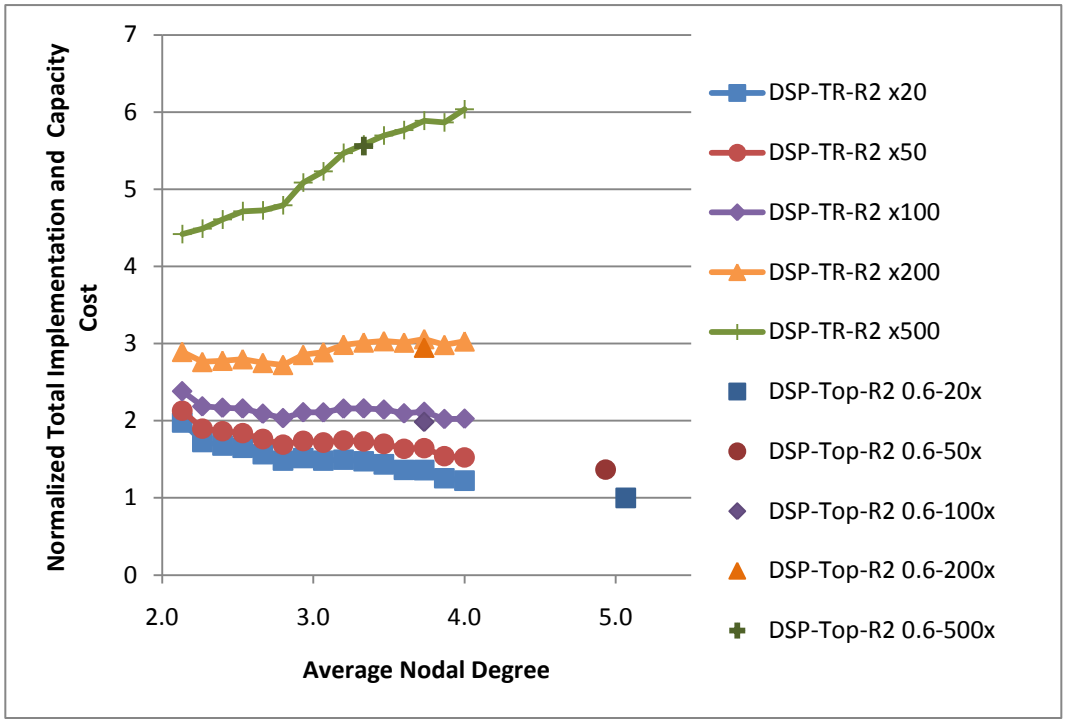


Figure 76 – DSP-Top-R2 15 node network design with a required R2 of 0.6

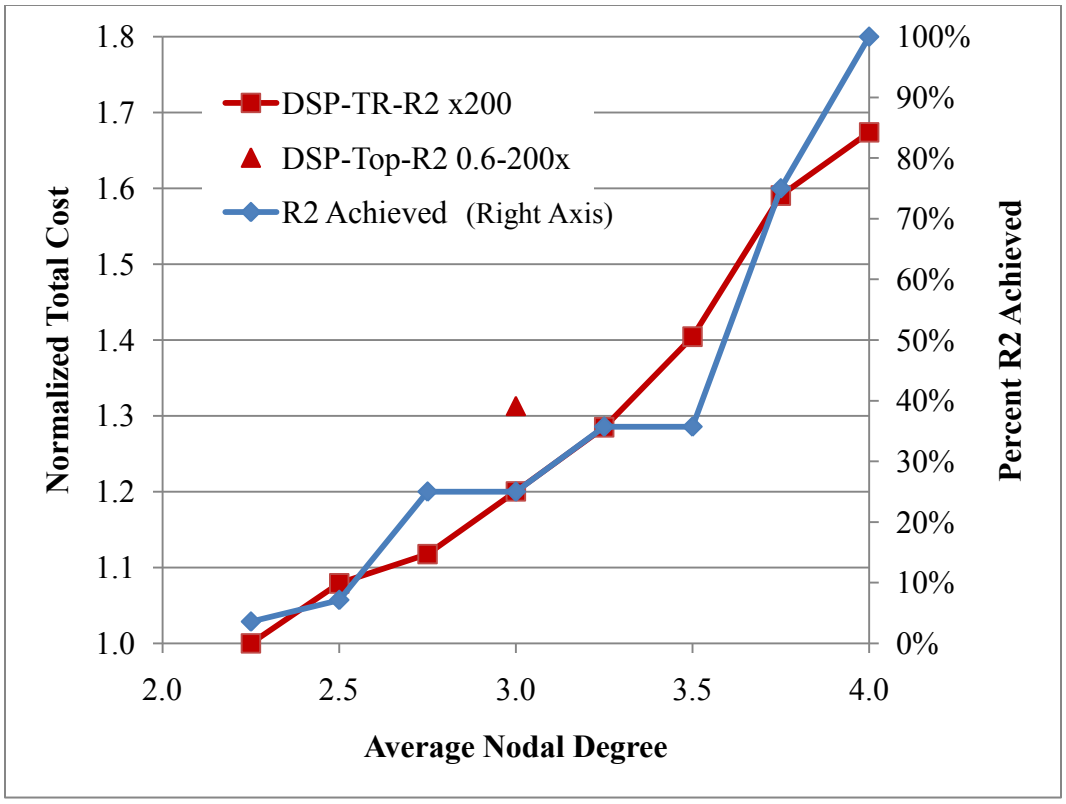


Figure 77 –8 node total cost comparison of the DSP-TR-R2 and the DSP-Top-R2 designs for a required R2 of 0.6 and an implementation factor of 200.

5.4.2 Availability Analysis of DSP-Top-R2

The availability of DSP-Top-R2 based network designs is a key indication of how well designing for dual failure restorability enables high availability networks, as the ultimate purpose of designing the network to resist more failures is to increase the network availability. Since the required R2 is met for all demands, the availability is consistently affected by only two or more simultaneous failures.

When increasing the minimum level of dual failure restorability, the availability asymptotically approached 1 as the unavailability contributions transitioned from dual failures to three or more failures (Figure 78 to Figure 81). There are some interesting characteristics that show up in these figures, namely the ranking of the availability level at various R2 levels. The availability at a required R2 of 0.5 has the network with an implementation factor of 20 with the highest availability, and

the network designed with an implementation factor of 500 with the lowest availability. At a required R2 of 1.0, the order is reversed.

This reordering of the ranking of the availability of the networks as the required R2 increased demonstrates the transition of what factors are impacting availability the most. At low R2 values, it is the length of the paths that is pushing the availability lower, as the networks (networks designed with a high implementation factor) would have to route each path along longer distances. The increase in availability due to the partial dual failure restorability is not significant does not negate the impact of these longer paths.

As the required R2 reaches complete dual failure restorability, it is the multipath impact on availability that dominates the factors pushing availability lower. As previously documented in section 5.1.1 , availability is generally adversely affected when DSP increases the number of paths utilized. This transition can be observed in all the network results.

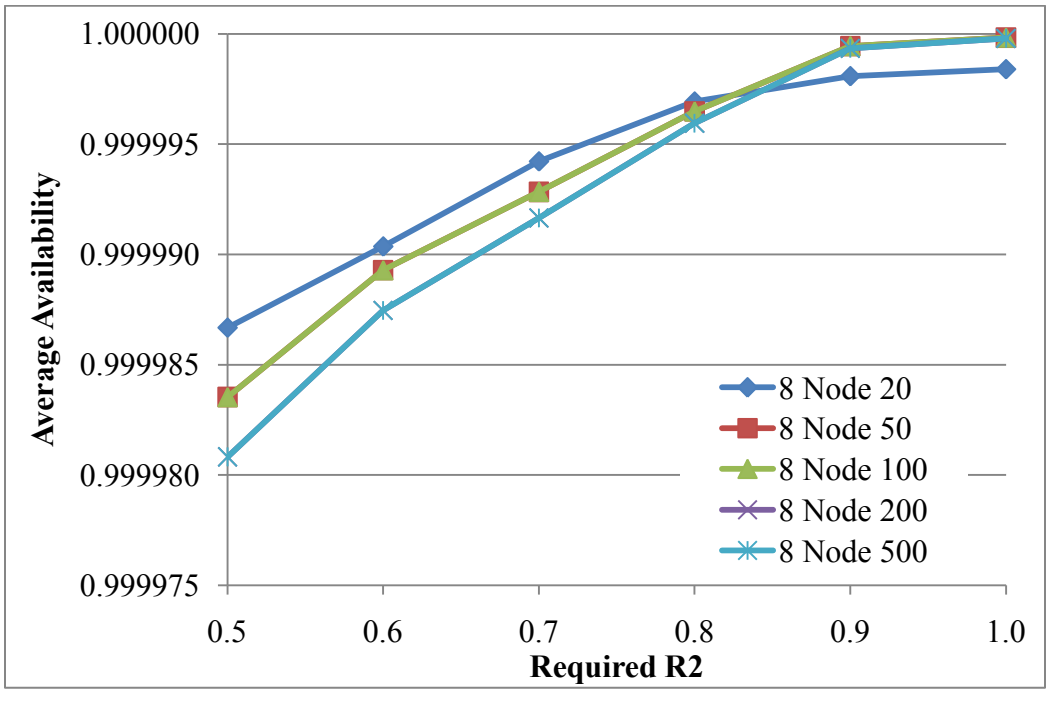


Figure 78 – DSP-Top-R2 availability results for the 8 node network by implementation factor

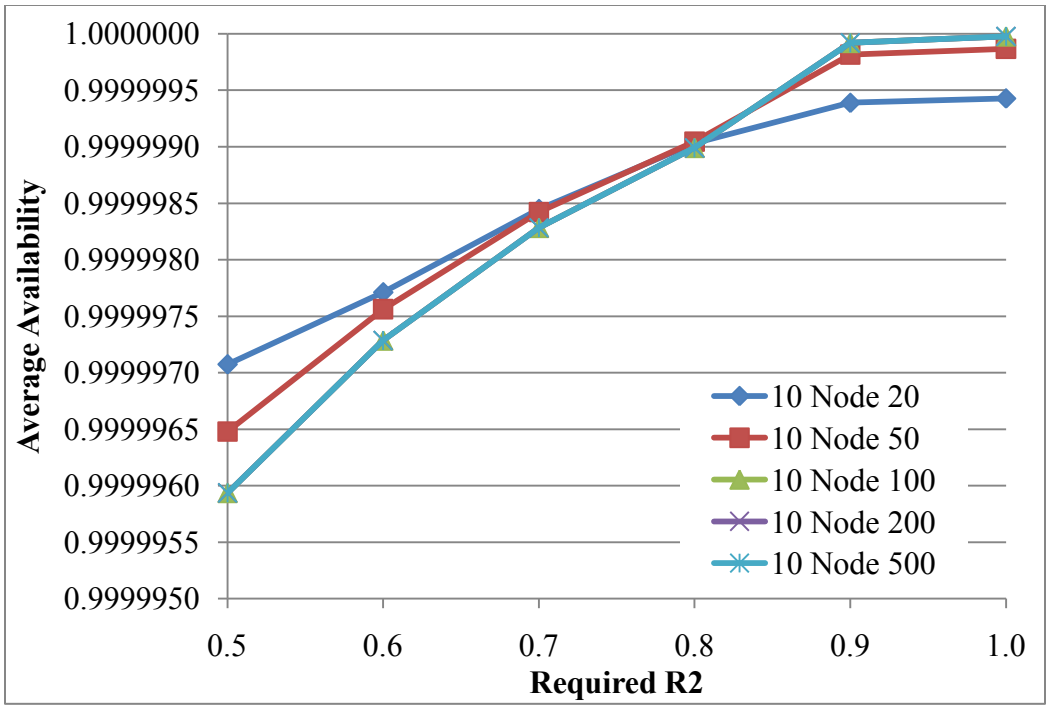


Figure 79 – DSP-Top-R2 availability results for the 10 node network by implementation factor

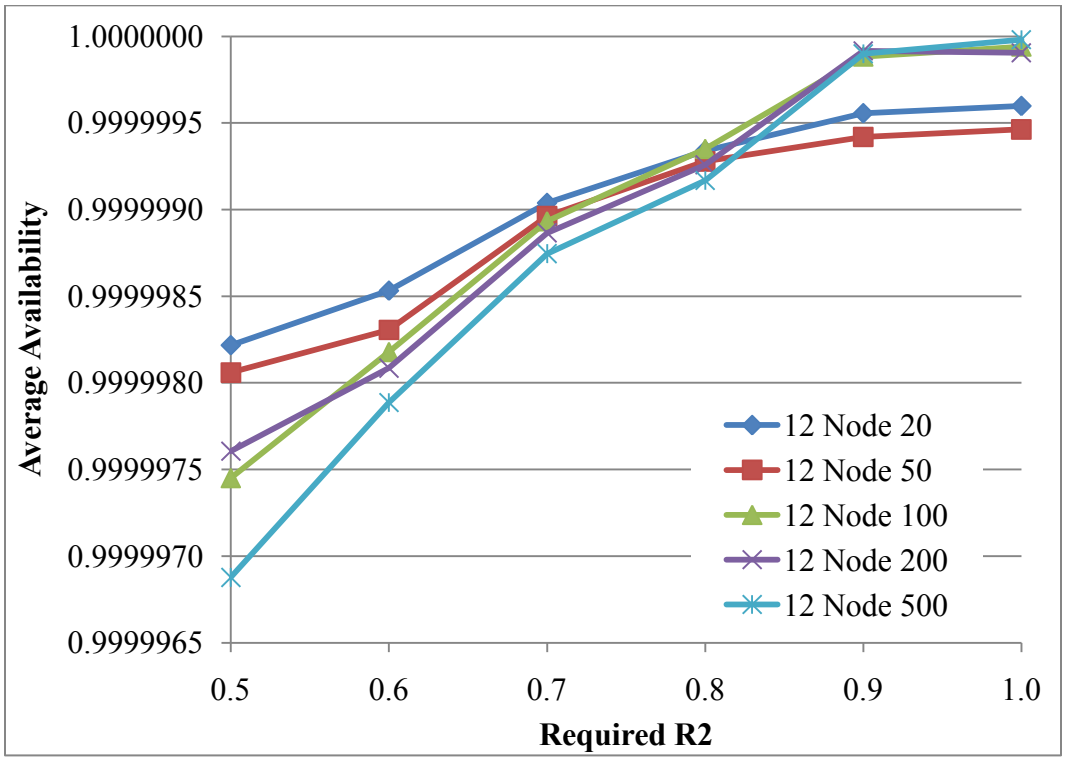


Figure 80 – DSP-Top-R2 availability results for the 12 node network by implementation factor

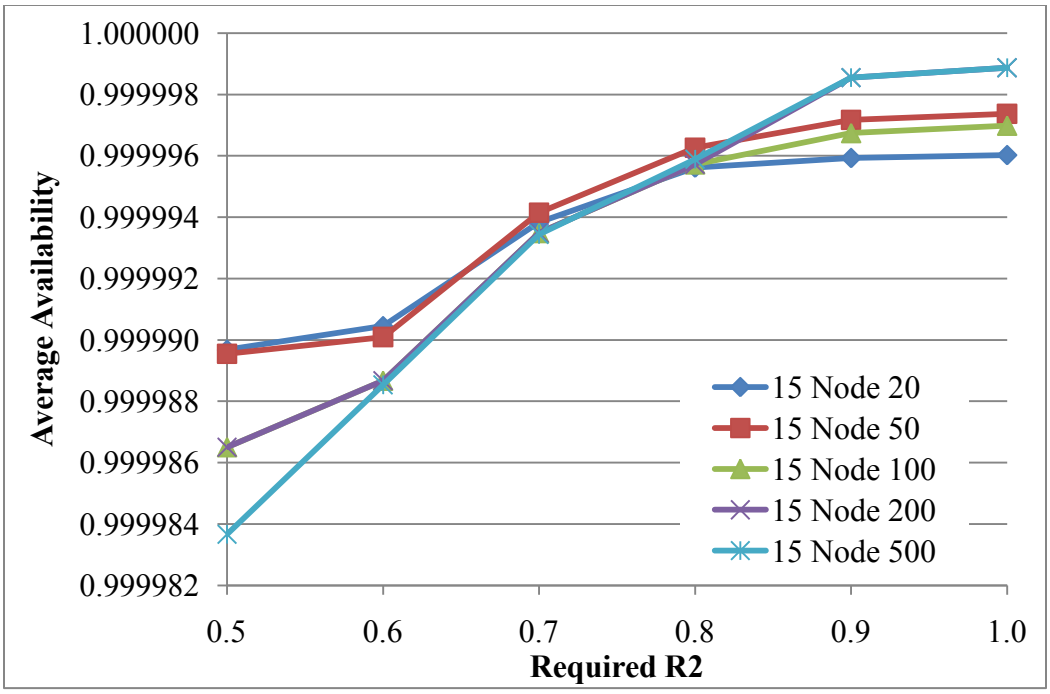


Figure 81 –DSP-Top-R2 availability results for the 15 node network by implementation factor

Comparing the availability of the DSP-Top and DSP-Top-R2, there is a stark drop between the DSP-Top availability and the availability of DSP-Top-R2 with the required dual failure of 0.5. Because many demands use only two paths, and the ones that do use more, generally have a dual failure restorability near 0.5 inherently, the DSP-Top results are somewhat higher than the 0.5 results.

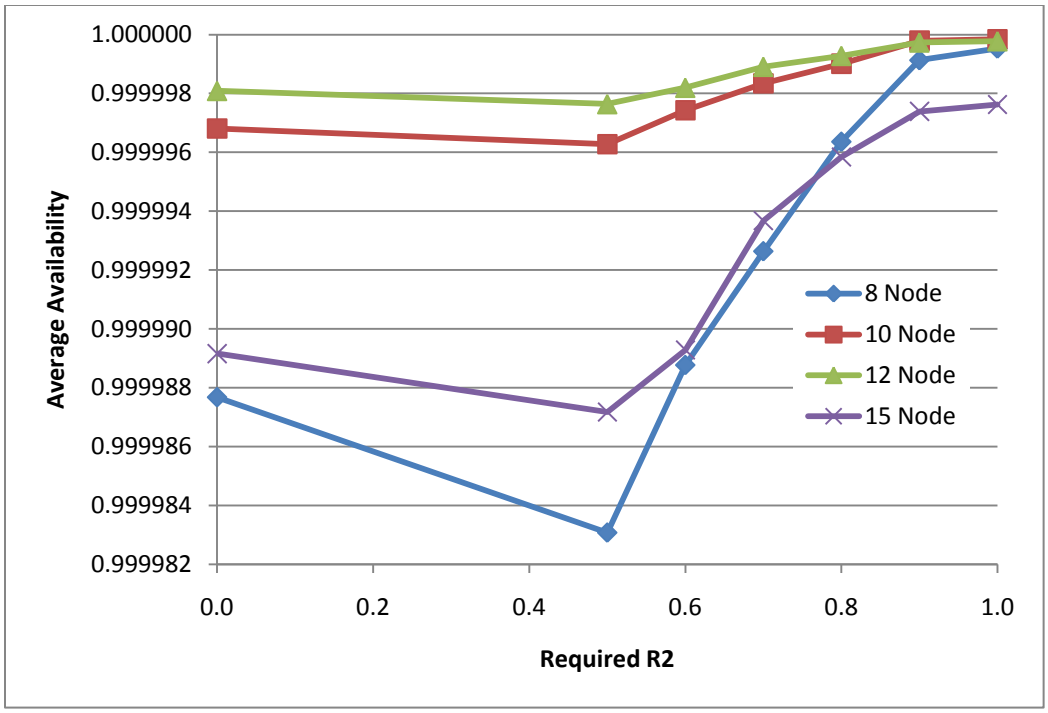


Figure 82 – DSP DSP-Top-R2 average availability across all implementation factors including DSP-Top results (a required R2 of 0.0)

Designing for a given dual failure restorability can have measured positive effect on availability, so long as the required level of restorability is above 0.6. Below this mark, availability is actually reduced due to increased exposure to possible failures.

The DSP-Top-R2 survivability model provided reasonable topology designs and demand routings even though for some networks the designs were not optimal, and a two stage process was required to establish the topology of the network and the routing for each demand. When compared to the network family DSP-TR-R2 results, the topology design has a significant impact on reducing the cost of implementing a network for any given level of dual failure restorability.

5.5 DSP-TR-A

The goal of incorporating dual failure, or multiple failure survivability, into a network design is to ultimately increase the availability of that network. As

previously discussed explicitly designing a network to survive multiple failures may end up decreasing the availability of particular demands in a network. This begs the question of why isn't a network designed to meet a certain availability requirement instead. A network survivability model was developed to do just that. However, upon implementation of the DSP-TR-A and the DSP-Top-A models as ILPs, the complexity of these problems became apparent.

The computational time to solve even the DSP-TR-A ILP for the 8 node networks were quite significant, and networks would have to be solved one demand at a time, with each demand taking hours, if not days to solve. This computational complexity made it impossible to obtain results from the DSP-Top-A model with current ILP solution techniques and available computing power. As such for the DSP-TR-A results, three networks were designed using the DSP-TR-A model. These networks were the 18, 24 and 30 span networks of the 15 node network family.

The results from the three works that were solved with the DSP-TR-A ILP were compared to the DSP-TR and the DSP-TR-R2 results. The comparison highlights the benefit of solving for a required availability level is apparent when looking at these results (Figure 83 through Figure 85). As discussed enforcing partial dual failure restorability can adversely affect availability due to increasing the path length, and the overall exposure of a given demand's traffic to failure.

For the DSP-TR-R2 networks to achieve the same level of availability as the DSP-TR-A networks, the capacity costs would increase by 5%, 21% and 42% for the 18, 24 and 30 span networks. The tradeoff for the reduced costs of the DSP-TR-A survivability designs is the significant increase in computing costs.

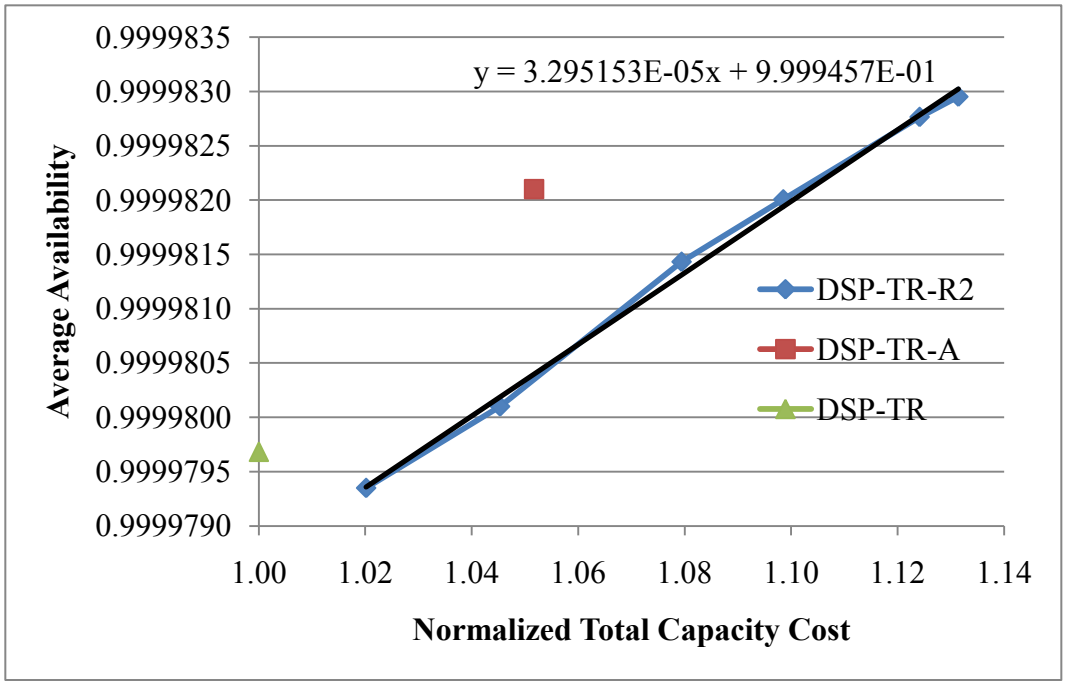


Figure 83 – Comparison of the costs and availability of the DSP-TR, DSP-TR-R2 and the DSP-TR-A results for the 15 node 18 span network with trend line for the DSP-TR-R2 results

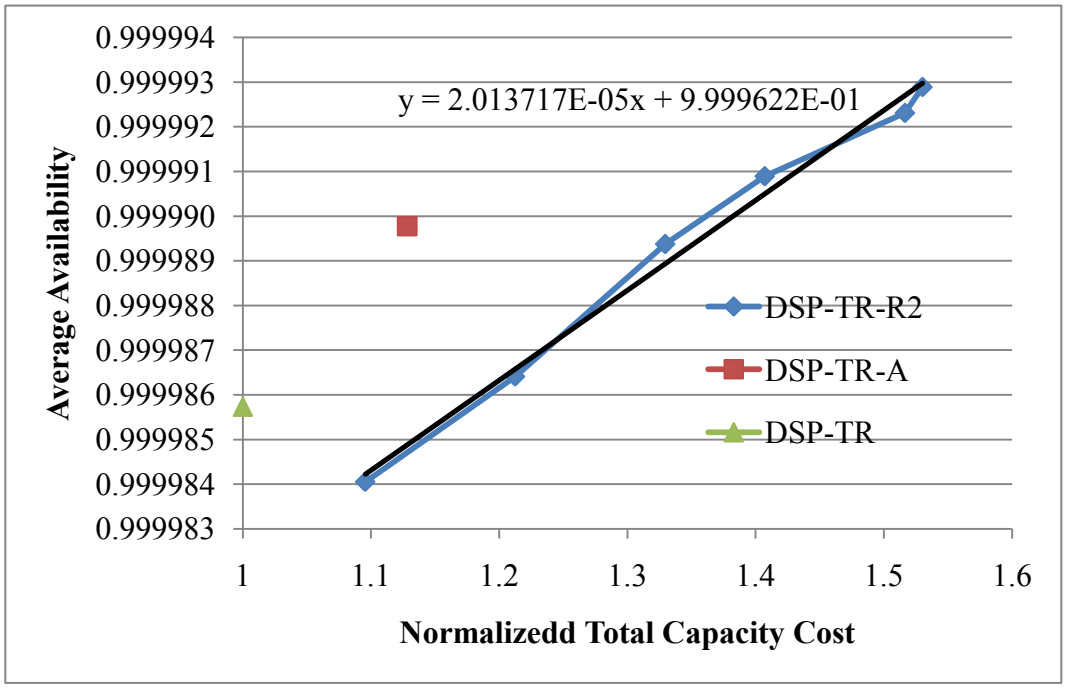


Figure 84 - Comparison of the costs and availability of the DSP-TR, DSP-TR-R2 and the DSP-TR-A results for the 15 node 24 span network with trend line for the DSP-TR-R2 results

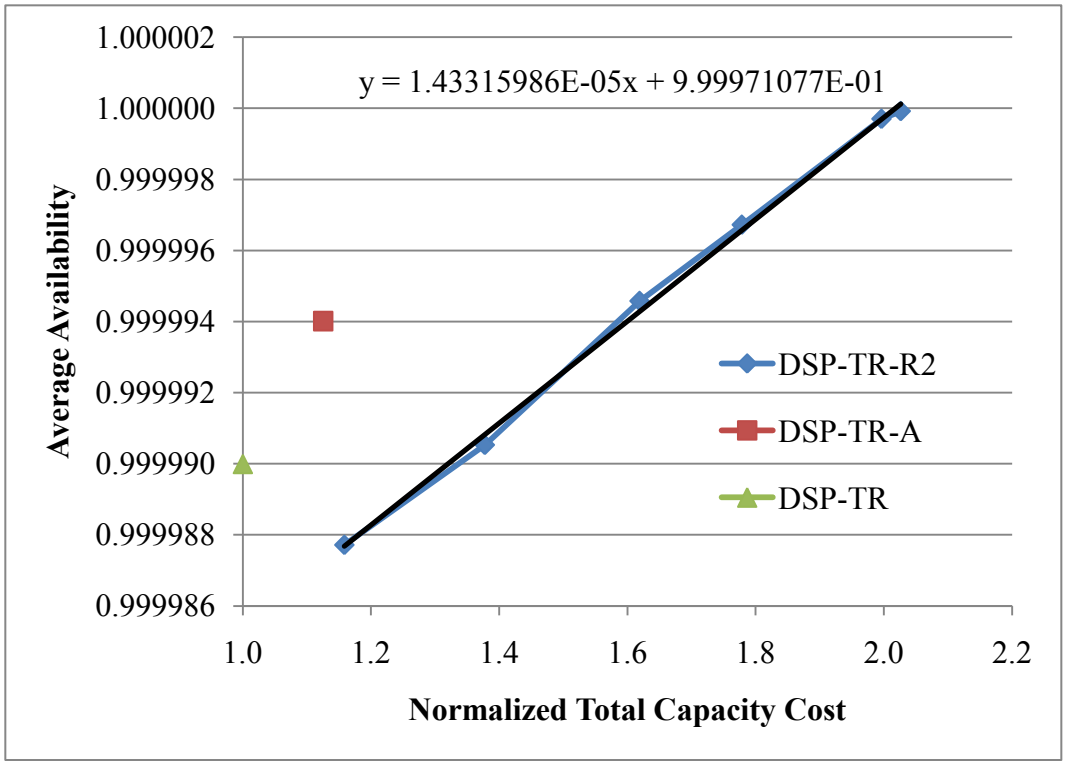


Figure 85 - Comparison of the costs and availability of the DSP-TR, DSP-TR-R2 and the DSP-TR-A results for the 15 node 30 span network with trend line for the DSP-TR-R2 results

The basis for the DSP-TR-A designs comes from the estimation of the availability of each of the spans. The accuracy of these availability estimations is a critical component of these designs. If availability of a span is over estimated, the DSP-TR-A model will under capacitate the network. Conversely, if a span has its availability over estimated, the network would be over capacitated. This is a significant risk in designing networks. It would be subject to further study analyzing the error of the availability estimations, and the impact that had on network designs. This would provide more insight to be able to mitigate the risks involved with designing an availability constrained network.

Each span may have individual characteristics that affect its availability, and if the span does not have a long history to base the availability estimation, or the external conditions affecting a span change, the adequacy of the DSP-TR-A designs to meet the availability requirements could be called into question. This

risk of being unable to confidently estimate availability may outweigh the cost savings of this design when compared to designing explicitly for a certain level of failure survivability.

Chapter 6. Discussion, Conclusions, and Recommendations for Further Work

The impact of failure in core communication networks is increasing significantly as more and more aspects of society come to rely on it, even for basic services. Companies looking to centralize the control centers of their plants, healthcare providers trying to increase their effectiveness by moving to electronic health records, government planning, even retail stores with the electronic payment methods and the automated supply chain management systems, all require a communication system with high availability, where even a few minutes of outages can prove to have a significant monetary and social cost.

The purpose of this work has been to design a network survivability model that will allow DSP designed networks to achieve a high level of availability and to optimize the network's topology. This investigation led to insights in the interaction between DSP cost savings and availability, the dual failure restorability levels and their impact on availability, and on the impact of topology design on availability.

ILP models were created based on the concept of DSP presented in [32] that utilized an arc-flow methodology to fix the path routing. The models that were created attempted to cover six different design paradigms. The first model allocated capacity on a fixed topology network that provided full single failure survivability, which was based off the originally published DSP model. This ILP model was modified to create a second DSP survivability model that concurrently optimized the topology of the network and allocated capacity. These designs would provide a baseline to compare the high availability designs.

The high availability models attempted to directly design networks to achieve a given level of availability. The DSP survivability model was modified in order to design networks for an arbitrary availability by utilizing a technique that was developed to calculate availability using linear equations. This technique proved to be computationally intensive, and was not viable to provide solutions for the topology design with the resources available. However, in limited cases, the results were adequate, and with better ILP and heuristic solution methods could be a promising basis on which to design high availability networks.

The other option to designing networks to achieve a given availability was to dictate the how much traffic survives a given number of failures. This was the approach that was taken when the first generation of survivability techniques were developed, and was the motivation behind developing the DSP model. Most survivability design techniques protect the network again single failures, with this providing a significant boost in the availability of the network (generally from around 2-3 nine's (2 nine's is an availability of 99% and 3 nines is an availability of 99.9%) up to 4-5 nine's [7]). The next step was to develop a survivability mechanism that could handle arbitrary levels of dual failure survivability, since this extra failure resiliency can be expensive, and not all traffic requires this level of protection. The DSP model using a transportation problem approach was developed to protect a portion of each demand from dual failures. This model was also adapted to concurrently optimize network topology.

The results from the simulated designs that resulted from implementing and running the ILP models representing the various survivability models demonstrated a number of characteristics about the use of DSP in designing high availability networks.

The first obvious note was that topology design problems do not easily solve with optimal solutions. Due to characteristics of the topology ILP problems, the LP solvers have a difficult time locating the global minimum using the branch and bound technique. In order to get results that were reasonable, a two-step approach was taken in designing the topology and capacity allocation of a network. The

first step was to remove some integer constraints, and solve the network for the best topology. This topology was then fixed, and the ILP models for the appropriate survivability mechanisms that did not optimize topology were used to determine path routing and span capacities. This proved effective, as most of the results were reasonable; however, they could not be called optimal.

The purpose of DSP is to reduce the total capacity cost of the network by allowing disjoint paths serving the same demand to share spare capacity. This use of multiple paths to share backup capacity had a significant negative impact on availability. When a demand routed capacity on multiple paths, the exposure to failure for the demand was significantly increased, and hence, even though there were only partial failures, these would occur with twice or more frequency. This effect was evident in all of the survivability models.

When each demand has a portion of its demand forced to be restorable to dual failures, the demand must utilize at least three paths. This use of multiple paths has an inherent level of dual failure restorability; however, if the dual failure restorability level is not sufficient, the availability of the demand will actually decrease. It was found that an R2 of 0.6-0.7 was usually required to increase the availability of a demand. Below that, and the increased exposure to failure outweighs the benefit of partial dual failure restorability. Above that and the average availability went from the 4-5 nine's level with single failure restorability up to 6-7 nine's.

Optimizing topology provided a small improvement on the total cost of a network compared to the network families when there were no dual failure restorability requirements; it did, however, provide an indication of what an optimal average nodal degree for a network should be. When designing for dual failure restorability, the results from the topology design solutions were significantly better than the results from the network families. This was because most of the network families were not tri-connected until their average nodal degree approached 4. The topology designs were able to fully protect all demands from

dual failures when the network families could only protect about half of the demands at the same average nodal degree.

The results presented have limitations that revolve around the assumptions made in the linearity and modularity of the network designs, as well as the technical ability of network equipment. In order to translate this work into network design models that are capable of accurately designing actual networks, the implications of modular capacity increments, non-linear equipment costs, and economies of scale would have to be integrated into the presented models. Also requiring consideration in using these methods for designing actual networks are geopolitical constraints on topology, the relevance of dual failure or availability design, and the technical challenge of splitting and combining traffic to travel over multiple paths. These limitations would need to be considered when utilizing the presented models to design actual network.

From this work there are a couple of areas that are open to further study. First, the format of the availability models (DSP-TR-A and DSP-Top-A) lend themselves to possibly be adapted to utilize advanced ILP techniques such as column generation. The column generation method of solving ILPs relies on a block wise structure to the constraints table, which the availability models appear to have. The other area of future work is to utilize the linear availability calculations on other survivability techniques. Other survivability mechanisms, especially path based mechanisms, could be adapted to also take availability into consideration in their ILP based designs.

Dual failure restorability of some sort is required for the next generation of communication networks, and the work presented here explored some of the implications of utilizing DSP for this added survivability versus the original single failure survivability model. By looking at how topology design can also affect dual failure restorability and demand availability, along with adding capacity, networks can be designed to more efficiently to increase their availability.

6.1 Contribution of Thesis Research

6.1.1 Multi-Flow Optimization Model for Design of a Shared Backup Path Protected Network

Todd, B.; Doucette, J., "Multi-Flow Optimization Model for Design of a Shared Backup Path Protected Network," Communications, 2008. ICC '08. IEEE International Conference on , vol., no., pp.131-138, 19-23 May 2008

Designing optimal shared backup path protected networks is a difficult and time-consuming task, and considerable research has been done to develop near optimal heuristics and algorithms that will solve the SBPP model without extensive computing power, but by definition, such methods are sub-optimal. This paper introduces a slight modification to the SBPP problem that allows it to be optimally solved using conventional ILP techniques. By allowing working and backup paths to follow multiple routes, the new SBPP model eliminates the numerous 1/0 variables in the conventional model. The fundamental characteristics of SBPP remain intact, with the problem altering only slightly but it allows ILP solvers to find an optimal solution in a time measured in seconds to minutes compared to the days or longer needed for conventional models.

6.1.2 Use of Network Families in Survivable Network Design and Optimization

Todd, B.; Doucette, J., "Use of Network Families in Survivable Network Design and Optimization," Communications, 2008. ICC '08. IEEE International Conference on , vol., no., pp.151-157, 19-23 May 2008

In modeling communication networks for simulation of survivability schemes, one goal is often to design these networks across varying degrees of nodal connectivity to get unbiased performance results. Abstractions of real networks, simple random networks, and families of networks are the most common categories of these sample networks. This paper looks at how using the network

family concept provides a solid unbiased foundation to compare different network protection models. The network family provides an advantage over random networks by requiring one solution per average nodal degree, as opposed to have to solve many, which could take a significant amount of time. Also, because the network family looks at a protection scheme across a variety of average nodal connectivities, a clearer picture of the scheme's performance is gained compared to just running the simulation on a single network.

6.1.3 Fast and Efficient Design of a Shared Backup Path Protected Network using a Multi-Flow Optimization Model

Todd, B.; Doucette, J., "Fast and Efficient Design of a Shared Backup Path Protected Network using a Multi-Flow Optimization Model," Pending Publication, IEEE Transactions on Reliability

As traffic on core communication networks has increased, along with demands for this traffic capacity to be highly available, network designs have also advanced in order to meet these growing requirements. These network designs are based on network survivability models that use various strategies to provision spare capacity throughout a network in order restore traffic in case of a failure, with a break of a fibre line being the most common of the failures affecting networks. This paper looks at a common protection model called shared backup path protection (SBPP). This is a popular model and there has been a significant amount of work done with it. However, the integer linear program (ILP) based SBPP model has proven difficult to solve using reasonable computing and time resources. While many algorithms and heuristics have been developed to design SBPP based networks, it has been difficult to know well these designs perform compared to ILP optimized networks. This paper presents an SBPP type ILP model that solves in a couple orders of magnitude less time than the traditional model. This new model will allow better benchmarking of SBPP based network

designs, and enhance further study into the performance of SBPP relative to other network survivability models.

6.1.4 Network Design with Availability Considerations

B. Todd, J.Doucette, "DSP Network design with Availability Considerations," DRCN 2008, Washington D.C., (in press)USA, October, 2009

This paper expands on the demand-wise shared protection model in order to capacitate a network that achieves pre-defined minimum availability requirements. By enabling DSP to take into consideration minimum availability, a network can be capacitated to meet user's requirements of the network. This provides a way to customize the level of protection throughout the network that matches the needs of those using the network.

6.1.5 Demand-Wise Shared Protection Network Design with Dual-Failure Restorability

B. Todd, J.Doucette, "Demand-Wise Shared Protection Network Design with Dual-Failure Restorability," RNDM 2009, (in press) St. Petersburg, Russia, October, 2009

The availability requirements placed on core communication networks have been rapidly increasing. As the value of the traffic served by these core networks has increased so has the impact of failure. Demand-wise shared protection (DSP) was developed to provide failure survivability in the network that was more efficient than concurrently routing two paths of traffic (1+1 APS), yet was more straightforward to manage than more complex schemes. The DSP model was adapted to ensure, in addition to 100% single failure survivability, a specified minimum level of dual-failure restorability. The effect of enforcing dual-failure restorability in DSP networks was evaluated in terms of cost and overall increases in availability. Counter intuitively, it was found that in some cases, requiring

some specified dual-failure restorability levels can result in decreases availability. DSP was effectively adapted to ensure dual-failure restorability, however, in order to capitalize on the capacity sharing aspects of the model, networks must be sufficiently well connected.

Bibliography

[1] David Gans, John Kralewski, Terry Hammons, and Bryan Dowd, "Medical Groups' Adoption Of Electronic Health Records And Information Systems" Health Affairs, 24(5): 1323-1333. September/October 2005.

[2] D. Crawford, "Fiber Optic Cable Dig-Ups," *Network Reliability: A Report to the Nation - A Compendium of Technical Papers*, National Engineering Consortium, Chicago, IL, June 1993.

[3] J. Doucette, "Network Restoration and High-Availability Network Design," Presentation at the City of Edmonton IT Conference, June 11-12, 2008.

[4] S. Chatterjee, A. Dutta and V. B. Chandhok, "Introduction—Network Convergence: Issues, Trends and Future", Information Systems Frontiers, vol.6, iss. 3, pp.183-188, Sept. 2004.

[5] CBC, "Backup systems failed in St. John's phone outage," [online] October 21, 2006 [accessed March 28, 2008] http://www.cbc.ca/canada/story/2006/10/21/nfld-outage.html.

[6] CNN**,** Third undersea internet cable cut in Mideast. *CNN*. [Online] February 1, 2008. [Cited: May 28, 2009.] http://www.cnn.com/2008/WORLD/meast/02/01/internet.outage/index.html.

[7] M. To and P. Neusy, "Unavailability analysis of long-haul networks," IEEE Journal on Selected Areas in Communications, vol. 12, no. 1, pp 100-108, Jan. 1994.

[8] M. Clouqueur, W.D. Grover, "Availability analysis of span-restorable mesh networks," *IEEE JSAC Special Issue on Recent Advances in Fundamentals of Network Management*, vol.20, no. 4, pp. 810-821, May 2002.

[9] Doucette, J.; Grover, W.D.; Giese, P.A., "Physical-Layer p-Cycles Adapted for Router-Level Node Protection: A Multi-Layer Design and Operation Strategy," Selected Areas in Communications, IEEE Journal on , vol.25, no.5, pp.963-973, June 2007.

[10] Onguetou, D.P.; Grover, W.D., "A New Insight and Approach to Node Failure Protection with Ordinary p-Cycles," Communications, 2008. ICC '08. IEEE International Conference on , vol., no., pp.5145-5149, 19-23 May 2008.

[11] Y. Liu, D. Tipper, "Successive Survivable Routing for Node Failures," *IEEE Global Telecommunications Conference (GlobeCom 2001)*, San Antonio, TX, pp. 25–29, November 2001.

[12] Shigang Chen, Klara Nahrstedt, “An Overview of Quality-of-Service Routing for the Next Generation High-Speed Networks: Problems and Solutions”, *IEEE Network, Special Issue on Transmission and Distribution of Digital Video*, vol. 12, no. 6, pp. 64-79, November/December 1998.

[13] W. Grover, Mesh-Based Survivable Networks, Upper Saddle River, NJ: Prentice Hall PTR, 2004.

[14] Manchester, J.; Bonenfant, P.; Newton, C., "The evolution of transport network survivability," *IEEE Communications Magazine*, vol.37, no.8, pp.44-51, Aug 1999.

[15] Telecom Paper, “14 Tbps over a Single Optical Fiber: Successful Demonstration of World's Largest Capacity,” [online] September 29, 2006 [accessed May, 28, 2009] http://www.telecompaper.com/news/article.aspx?cid=531873.

[16] Herzberg, M.; Bye, S.J., "An optimal spare-capacity assignment model for survivable networks with hop limits," Global Telecommunications Conference, 1994. GLOBECOM '94. Communications: The Global Bridge., IEEE , vol.3, no., pp.1601-1606 vol.3, 28 Nov- 2 Dec 1994.

[17] A Kister, A Zymolka, M Jager, R Hulsermann, “Demand-wise Shared Protection for Meshed Optical Networks” Design of Reliable Communication Networks (DRCN 2003), Banff, Alberta, Canada, pp.85-92, October, 2003.

[18] M. Herzberg, S. J. Bye, A. Utano, “The Hop-Limit Approach for Spare-Capacity Assignment in Survivable Networks,” IEEE/ACM Transactions on Networking, vol. 3, no. 6, pp. 775-784, December 1995.

[19] Todd, B.; Doucette, J., "Multi-Flow Optimization Model for Design of a Shared Backup Path Protected Network," Communications, 2008. ICC '08. IEEE International Conference on , vol., no., pp.131-138, 19-23 May 2008.

[20] Sheng Huang; Mukherjee, B., "Adaptive Reliable Multi-Path Provisioning in WDM Mesh Networks," Communications, 2008. ICC '08. IEEE International Conference on, pp.5300-5304, 19-23 May 2008

[21] Lei Guo; Jin Cao; Hongfang Yu; Lemin Li, "Path-based routing provisioning with mixed shared protection in WDM mesh networks," Lightwave Technology, Journal of , vol.24, no.3, pp.1129-1141, March 2006.

[22] Ling Zhou; Grover, W.D., "A theory for setting the "safety margin" on availability guarantees in an SLA," Design of Reliable Communication Networks, 2005. (DRCN 2005). Proceedings.5th International Workshop on , vol., no., pp. 7 pp.-, 16-19 Oct. 2005.

[23] J. Lee and R. Ben-Natan, Integrating Service Level Agreements, Wiley Publishing, 2002.

[24] Jacques Marescaux, MD, Joel Leroy, MD, Francesco Rubino, MD, et. al., “Transcontinental Robot-Assisted Remote Telesurgery: Feasibility and

Potential Applications," Annals of Surgery, vol. 235, iss. 4, pp.487-492, April, 2002.

[25] J. Doucette, M. Clouqueur, W.D. Grover, "On the Availability and Capacity Requirements of Shared Backup Path-Protected Mesh Networks," *Optical Networks Magazine*, pp. 29-44, November/December 2003.

[26] M. Clouqueur, W. D. Grover, "Quantitative comparison of end-to-end availability of service paths in ring and mesh-restorable networks," Proceedings of the 19th Annual National Fiber Optics Engineers Conference (NFOEC 2003), Orlando, FL, USA, 7-11 September 2003

[27] Ling Zhou; Held, M.; Sennhauser, U., "Connection Availability Analysis of Shared Backup Path-Protected Mesh Networks," *Lightwave Technology, Journal of* , vol.25, no.5, pp.1111-1119, May 2007

[28] W. D. Grover, J. Doucette, "Topological design of span-restorable mesh transport networks," Annals of Operations Research, Special Issue on Topological Design of Telecommunication Networks, vol. 106, pp. 79-125, September 2001

[29] M. Modarres, M. Kaminsky, V. Krivstov, Reliability Engineering and Risk Analysis, New York, NY: Marcel Dekker, 1999.

[30] W. Winston, Operations Research Applications and Algorithms, Belmont, CA: Brooks/Cole-Thomson Learning, 2004.

[31] Todd, B.; Doucette, J., "Use of Network Families in Survivable Network Design and Optimization," Communications, 2008. ICC '08. IEEE International Conference on , vol., no., pp.151-157, 19-23 May 2008.

[32] R. Hülsermann, M. Jäger, A.M.C.A Koster, S. Orlowski, R. Wessäly, A. Zymolka, "Availability and Cost Based Evaluation of Demand-wise Shared Protection", *ITG workshop on Photonic Networks 2006*, VDE-Verl., pp 161-168, March 2006.

[33] J. Doucette, W. D. Grover, "Capacity Design Studies of Span-Restorable Mesh Networks with Shared-Risk Link Group (SRLG) Effects," *Optical Networking and Communications Conference (OptiComm 2002)*, Boston, MA, USA, pp. 25-38, July-August 2002.

[34] G. Ellinas, T.E. Stern, "Automatic protection switching for link failures in optical networks with bi-directional links", *Global Telecommunications Conference*, 1996. ("Globecom 96"), vol. 1, pp. 152 – 156, Nov. 1996.

[35] R. Bhandari, Survivable Networks: Algorithms for Diverse Routing, Kluwer Academic Publishers, November 1998.

[36] D. Stamatelakis, W.D. Grover, "Theoretical Underpinnings for the Efficiency of Restorable Networks Using Pre-configured Cycles ("p-cycles")," *IEEE Transactions on Communications*, vol.48, no.8, pp. 1262-1265, August 2000.

[37] M. W. Maeda, “Management and Control of Transparent Optical Networks,” *IEEE Journal on Selected Areas in Communications* (JSAC), vol. 16, no. 7, pp. 1005-1023, September 1998.

[38] Y. Xiong, L. G. Mason, “Restoration strategies and spare capacity requirements in self-healing ATM networks,” *IEEE/ACM Transactions on Networking*, vol. 7, no. 1, pp. 98-110, February 1999.

[39] R. Hülsermann, M. Jäger, A.M.C.A Koster, S. Orlowski, R. Wessäly, A. Zymolka, “Availability and Cost Based Evaluation of Demand-wise Shared Protection”, *ITG workshop on Photonic Networks 2006*, VDE-Verl., pp 161-168, March 2006.

[40] J. Doucette, “Advances on Design and Analysis of Mesh-Restorable Networks,” Ph.D. Dissertation, University of Alberta, Edmonton, AB, Canada, 03 December 2004.

[41] Schupke, D.A., “Guaranteeing Service Availability in Optical Network Design,” Invited Talk, *Asia-Pacific Optical Communications Conference* (APOC), Shanghai, China, November 6-10, 2005.

[42] Gruber, C.G.; Koster, A.M.C.A.; Orlowski, S.; Wessaly, R.; Zymolka, A., "A computational study for demand-wise shared protection," *Design of Reliable Communication Networks*, 2005. (DRCN 2005). Proceedings.5th International Workshop on , vol., no., pp. 8 pp.-, 16-19 Oct. 2005.

[43] Frederick, M.T.; Datta, P.; Somani, A.K., "Evaluating dual-failure restorability in mesh-restorable WDM optical networks," *Computer Communications and Networks, 2004. Proceedings. 13th International Conference on* , vol., no., pp.309-314, 11-13 Oct. 2004.

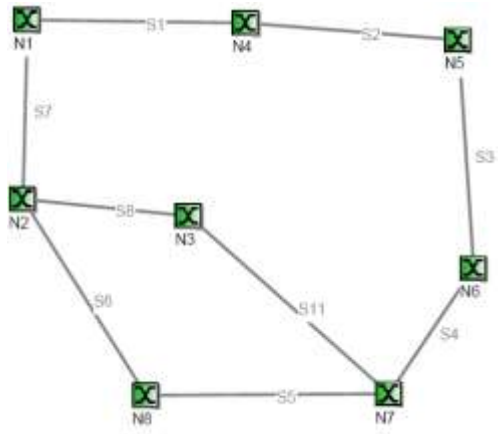
[44] C.-C. Sue and J.-Y. Du, “Capacity-efficient strategy for 100% dual-failure restorability in optical mesh networks utilising reconfigurable p-cycles and a forcer filling concept”, *IET Communications*, Volume 3, Issue 2, p. 198-208, Feb. 2009.

[45] M. Clouqueur, W.D. Grover, “Mesh-restorable networks with enhanced dual-failure restorability properties,” *Photonic Network Communications*, Springer Science, 9:1, pp. 7-18, Jan. 2005.

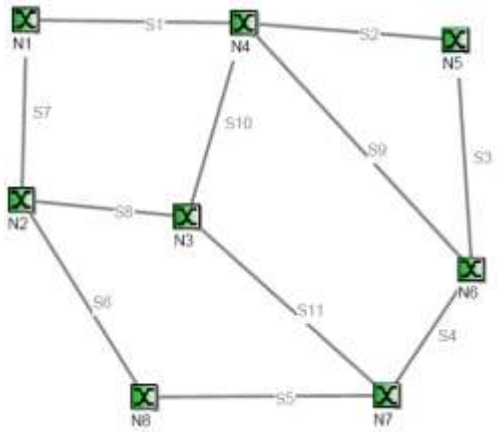
[46] Chen, B.K.; Tobagi, F.A., "Network Topology Design to Optimize Link and Switching Costs," *Communications, 2007. ICC '07. IEEE International Conference on* , vol., no., pp.2450-2456, 24-28 June 2007.

Appendix 1 Network Families

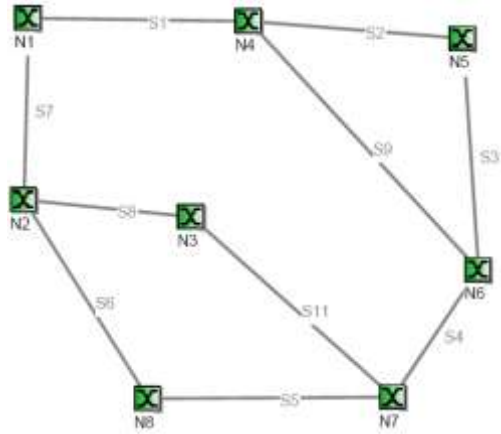
1.1 8 node network



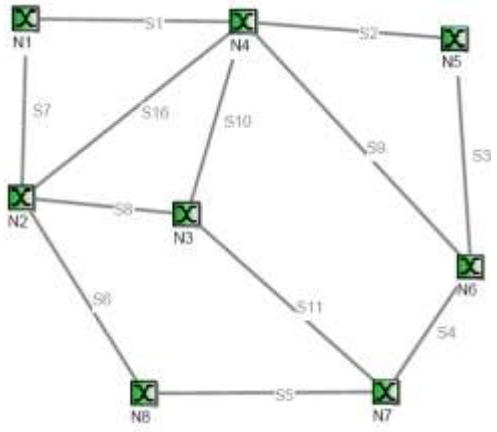
9 s



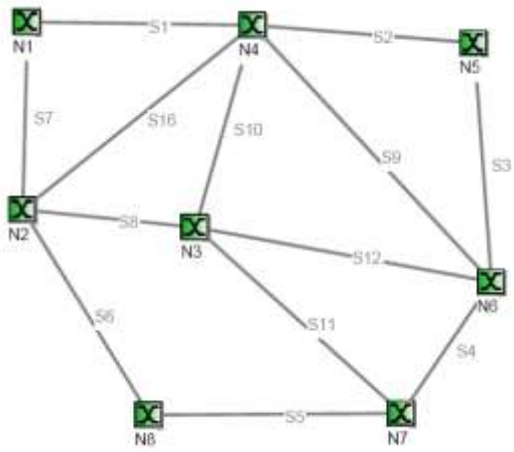
11 s



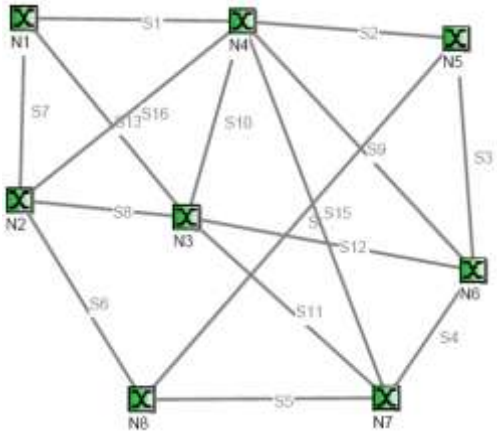
10 s



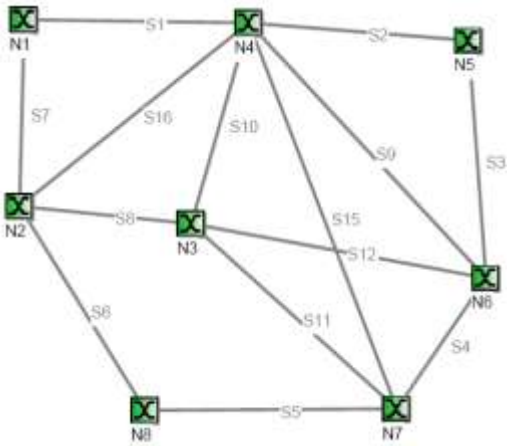
12 s



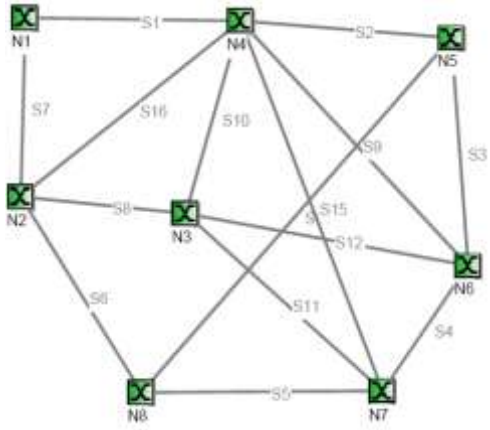
13 s



16 s

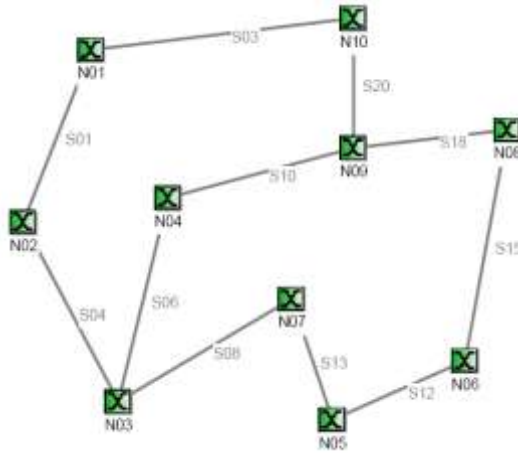


14 s

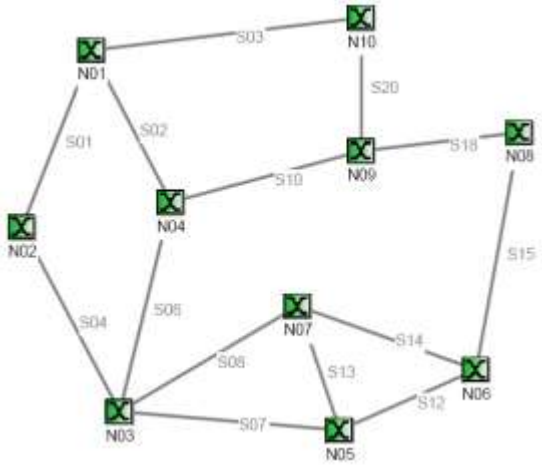


15 s

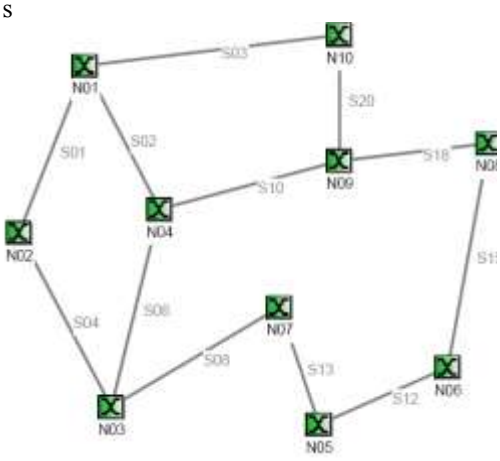
1.2 10 Node Network Family



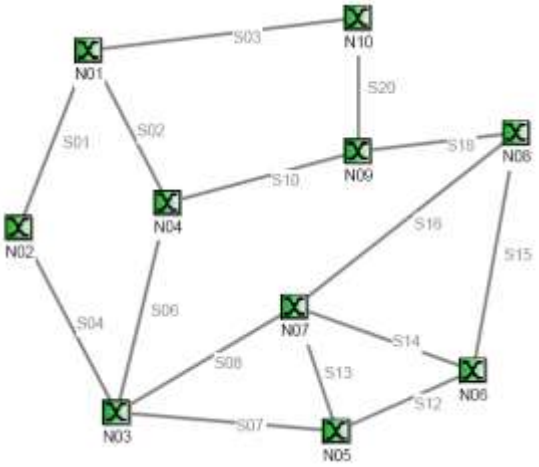
11 s



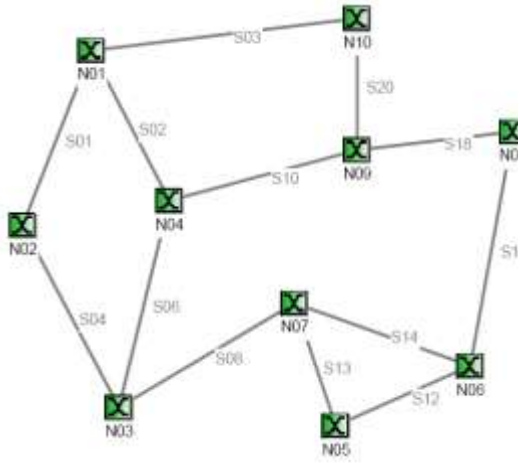
14 s



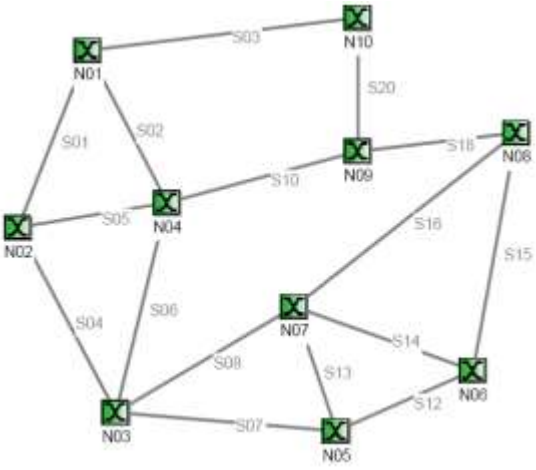
12 s



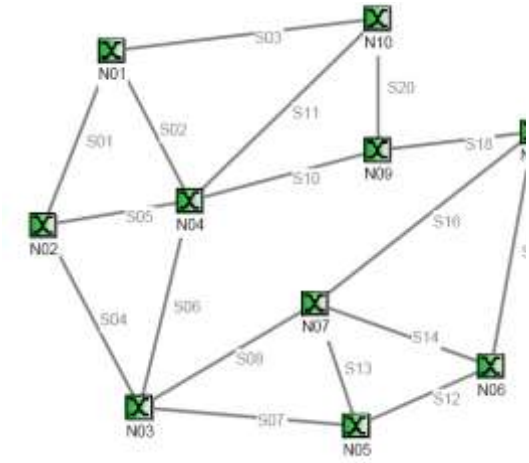
15s



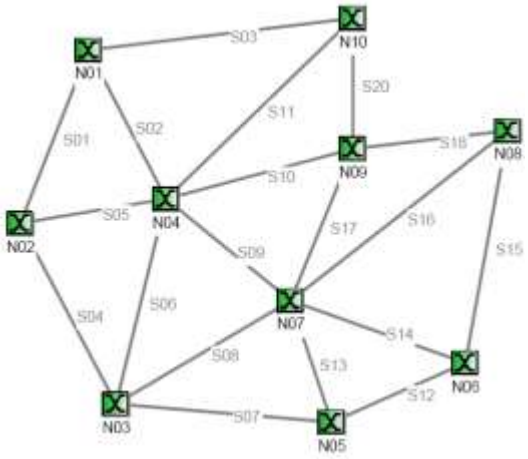
13 s



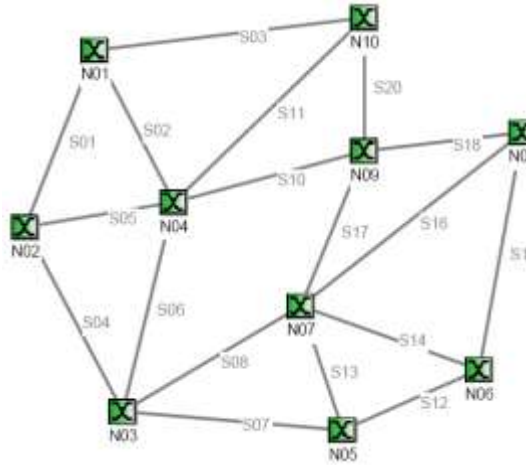
16 s



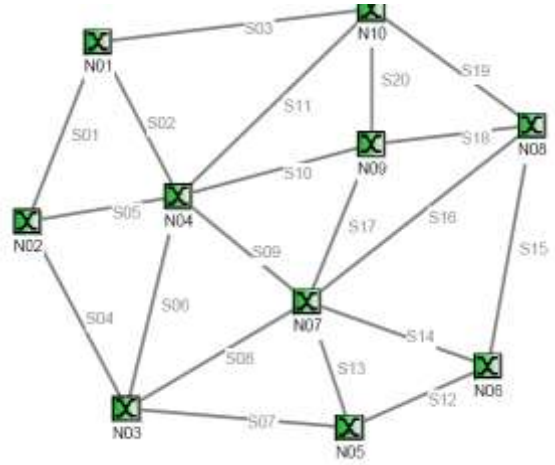
17s



19 s

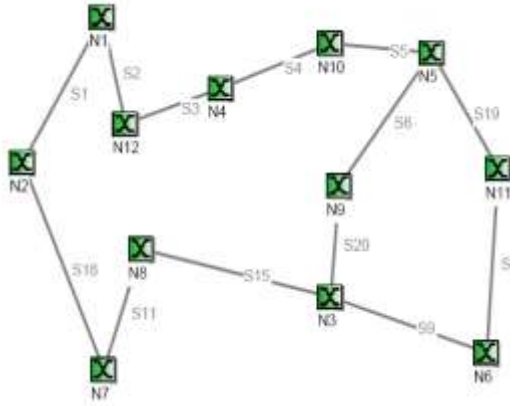


18 s

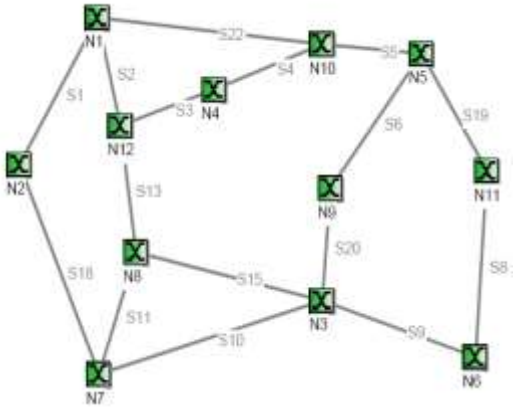


20s

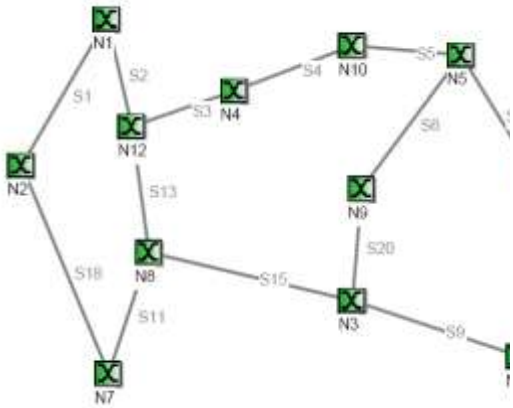
1.3 12 Node Network Family



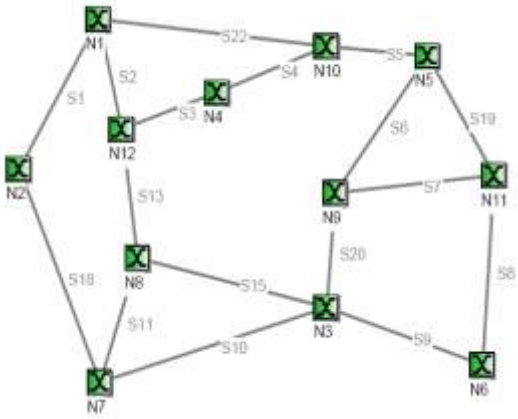
13 s



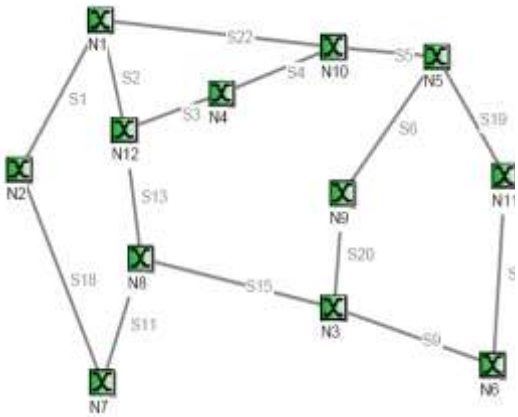
16 s



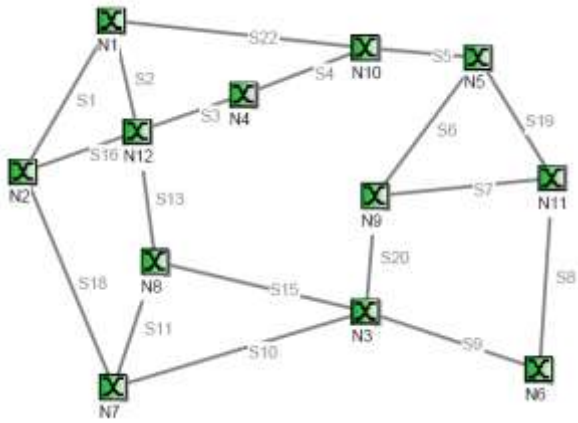
14 s



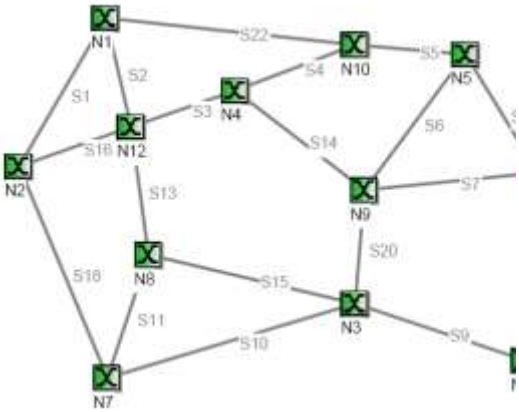
17 s



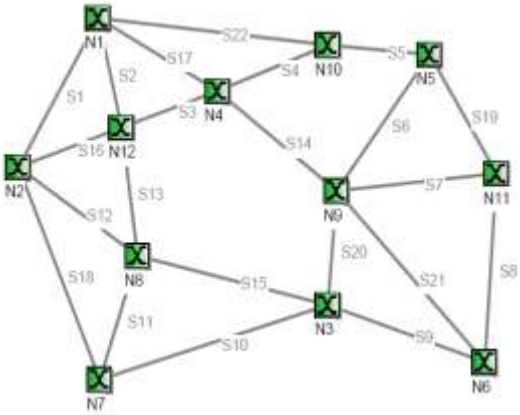
15s



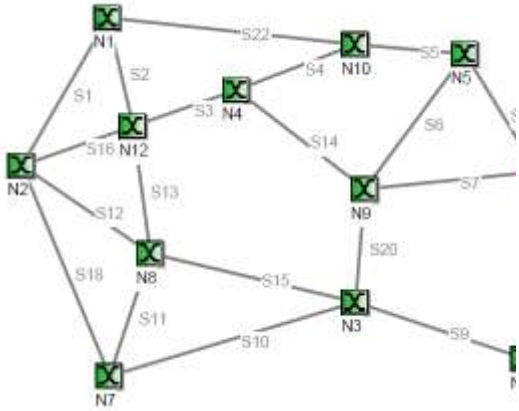
18 s



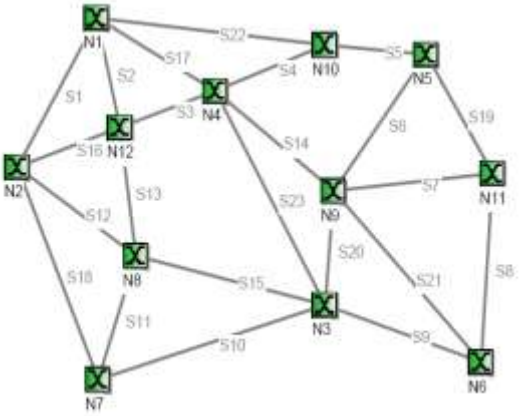
19 s



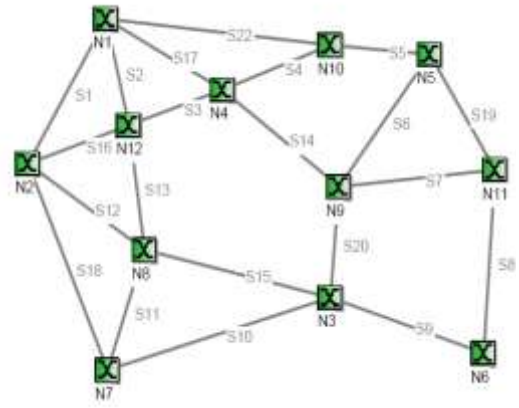
22 s



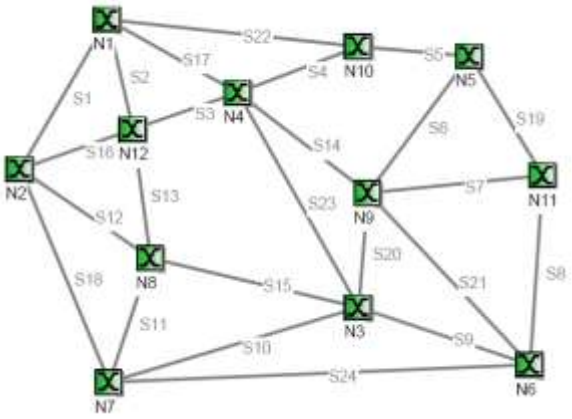
20 s



23 s

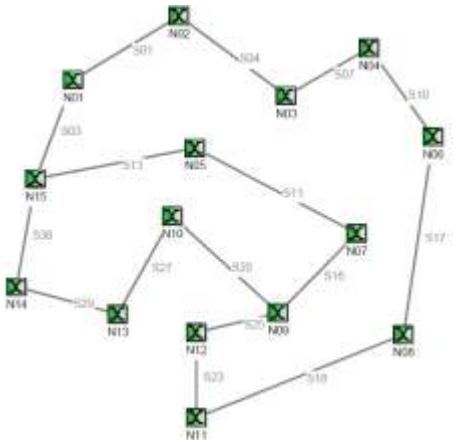


21 s

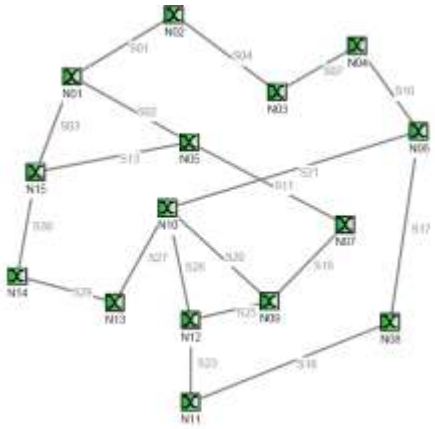


24s

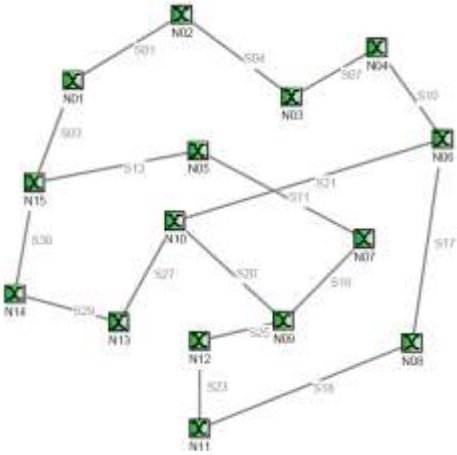
1.4 15 Node Network Family



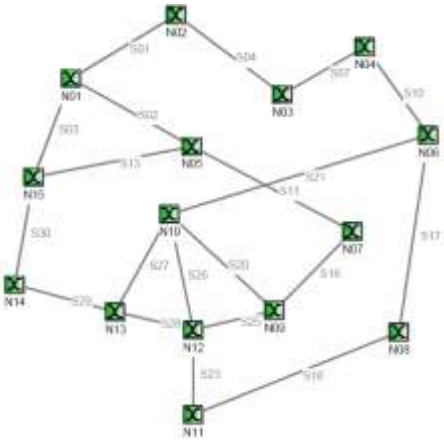
16 s



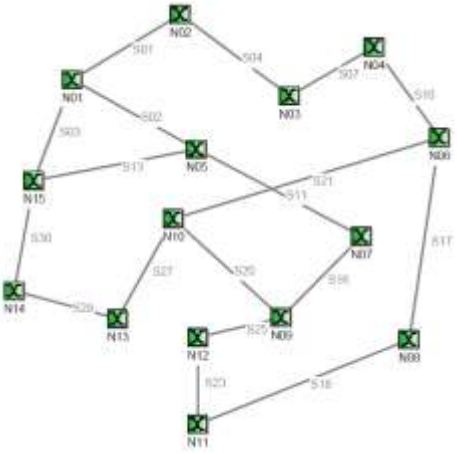
19 s



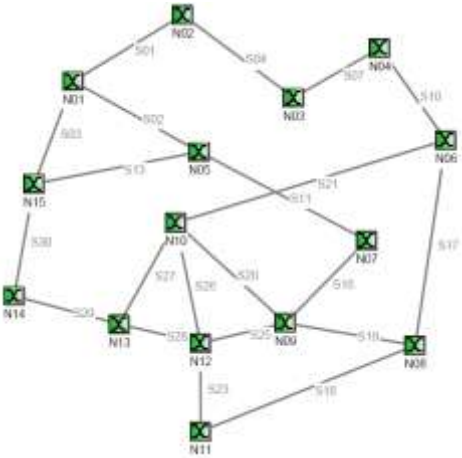
17 s



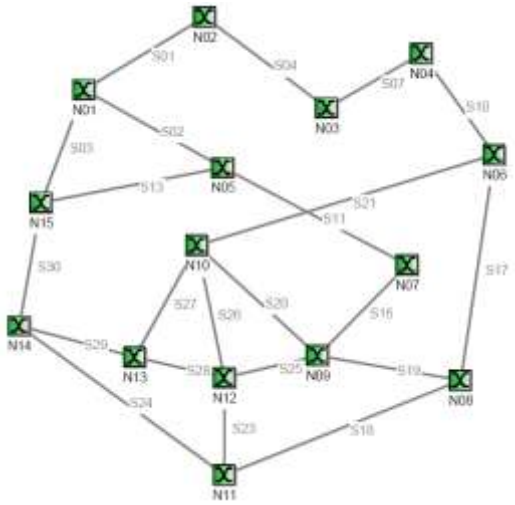
20 s



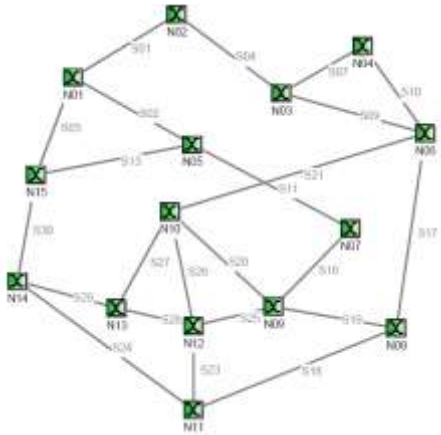
18 s



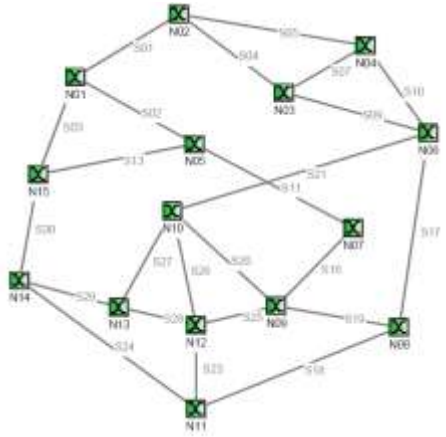
21 s



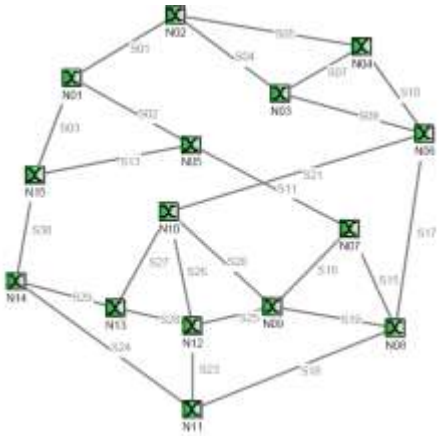
22 s



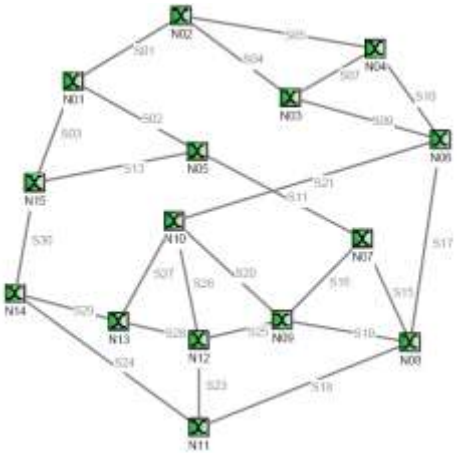
23 s



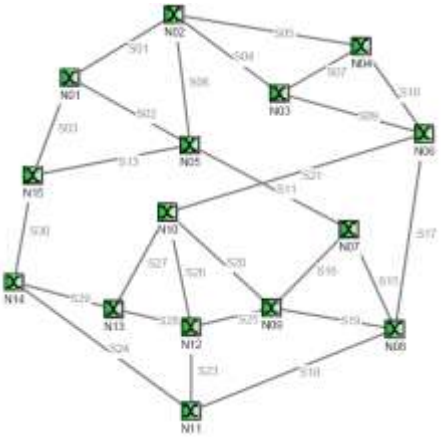
24 s



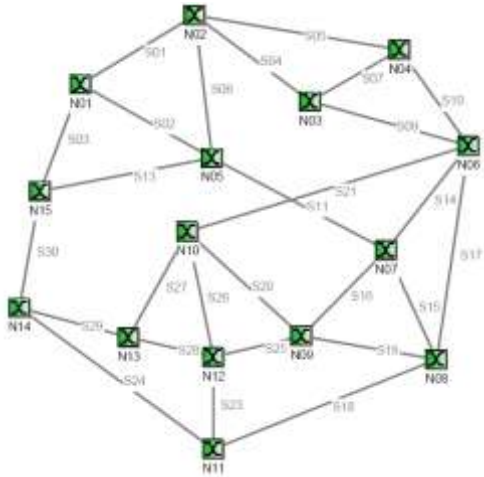
25 s



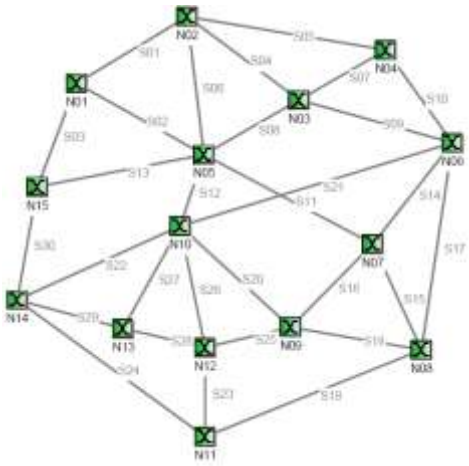
26 s



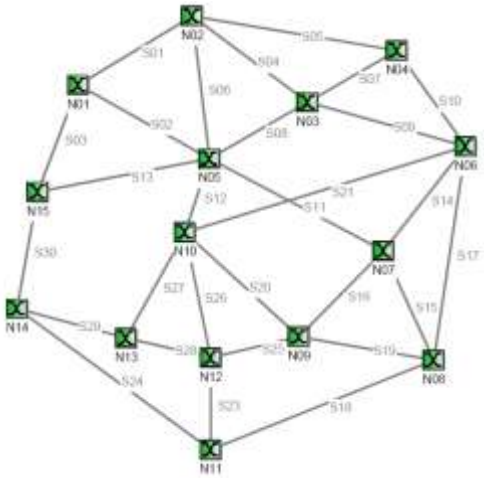
27 s


N01
N02
N03
N04
N05
N06
N07
N08
N09
N10
N11
N12
N13
N14
N15

28 s


N01
N02
N03
N04
N05
N06
N07
N08
N09
N10
N11
N12
N13
N14
N15

30 s


N01
N02
N03
N04
N05
N06
N07
N08
N09
N10
N11
N12
N13
N14
N15

29 s

Appendix 2 Sample DSP-Top Topology results

2.1 8 node network – 20x

```
NAME: 8n16s_Imp_Imp
DATE LAST MODIFIED: Friday, September 19, 2008 1:21:10 PM MDT
MODIFIED BY: MeshBuilder(c)


NODE  X     Y     SIZE
N1    175   58    3
N2    171   249   4
N3    346   266   4
N4    406   61    3
N5    631   79    3
N6    648   322   3
N7    558   456   3
N8    300   457   3

SPAN  O     D     LENGTH      MTBF  MTTR  FIXEDCOST   UNITCOST
S1    N1    N2    191.0418802 0     0     0     191.0418802
S2    N1    N4    231.0194797 0     0     0     231.0194797
S3    N1    N3    269.267525  0     0     0     269.267525
S5    N2    N3    175.8237754 0     0     0     175.8237754
S6    N2    N8    244.7549795 0     0     0     244.7549795
S8    N3    N8    196.4611921 0     0     0     196.4611921
S9    N3    N4    213.6000936 0     0     0     213.6000936
S10   N4    N5    225.7188517 0     0     0     225.7188517
S11   N5    N6    243.5939244 0     0     0     243.5939244
S12   N5    N3    340.8724101 0     0     0     340.8724101
S14   N6    N7    161.4187102 0     0     0     161.4187102
S15   N6    N3    307.1481727 0     0     0     307.1481727
S17   N7    N8    258.001938  0     0     0     258.001938
S18   N7    N3    284.682279  0     0     0     284.682279
```

2.1.1 8 node network – 50x

```
NAME: 8n16s_Imp_Imp
DATE LAST MODIFIED: Friday, September 19, 2008 1:21:10 PM MDT
MODIFIED BY: MeshBuilder(c)


NODE  X     Y     SIZE
N1    175   58    3
N2    171   249   4
N3    346   266   4
N4    406   61    3
N5    631   79    3
N6    648   322   3
N7    558   456   3
```

```
N8    300   457   3

SPAN  O     D     LENGTH      MTBF  MTTR  FIXEDCOST   UNITCOST
S1    N1    N2    191.0418802 0     0     0     191.0418802
S2    N1    N4    231.0194797 0     0     0     231.0194797
S5    N2    N3    175.8237754 0     0     0     175.8237754
S6    N2    N8    244.7549795 0     0     0     244.7549795
S8    N3    N8    196.4611921 0     0     0     196.4611921
S9    N3    N4    213.6000936 0     0     0     213.6000936
S10   N4    N5    225.7188517 0     0     0     225.7188517
S11   N5    N6    243.5939244 0     0     0     243.5939244
S14   N6    N7    161.4187102 0     0     0     161.4187102
S15   N6    N3    307.1481727 0     0     0     307.1481727
S17   N7    N8    258.001938  0     0     0     258.001938
```

2.1.2 8 node network – 100x

```
NAME: 8n16s_Imp_Imp
DATE LAST MODIFIED: Friday, September 19, 2008 1:21:10 PM MDT
MODIFIED BY: MeshBuilder(c)


NODE  X     Y     SIZE
N1    175   58    3
N2    171   249   4
N3    346   266   4
N4    406   61    3
N5    631   79    3
N6    648   322   3
N7    558   456   3
N8    300   457   3

SPAN  O     D     LENGTH      MTBF  MTTR  FIXEDCOST   UNITCOST
S1    N1    N2    191.0418802 0     0     0     191.0418802
S2    N1    N4    231.0194797 0     0     0     231.0194797
S5    N2    N3    175.8237754 0     0     0     175.8237754
S6    N2    N8    244.7549795 0     0     0     244.7549795
S8    N3    N8    196.4611921 0     0     0     196.4611921
S9    N3    N4    213.6000936 0     0     0     213.6000936
S10   N4    N5    225.7188517 0     0     0     225.7188517
S11   N5    N6    243.5939244 0     0     0     243.5939244
S14   N6    N7    161.4187102 0     0     0     161.4187102
S17   N7    N8    258.001938  0     0     0     258.001938
```

2.1.3 8 node network – 200x

```
NAME: 8n16s_Imp_Imp
DATE LAST MODIFIED: Friday, September 19, 2008 1:21:10 PM MDT
MODIFIED BY: MeshBuilder(c)
```

```
NODE  X     Y     SIZE
N1    175   58    3
N2    171   249   4
N3    346   266   4
N4    406   61    3
N5    631   79    3
N6    648   322   3
N7    558   456   3
N8    300   457   3

SPAN  O     D     LENGTH      MTBF  MTTR  FIXEDCOST   UNITCOST
S1    N1    N2    191.0418802 0     0     0     191.0418802
S2    N1    N4    231.0194797 0     0     0     231.0194797
S5    N2    N3    175.8237754 0     0     0     175.8237754
S8    N3    N8    196.4611921 0     0     0     196.4611921
S9    N3    N4    213.6000936 0     0     0     213.6000936
S10   N4    N5    225.7188517 0     0     0     225.7188517
S11   N5    N6    243.5939244 0     0     0     243.5939244
S14   N6    N7    161.4187102 0     0     0     161.4187102
S17   N7    N8    258.001938  0     0     0     258.001938
```

2.1.4 8 node network – 500x

```
NAME: 8n16s_Imp_Imp
DATE  LAST  MODIFIED:   """Friday"  """"  September   """19"      """"  2008
      1:21:10     PM    MDT
MODIFIED    BY:   MeshBuilder(c)



NODE  X     Y     SIZE
N1    175   58    3
N2    171   249   4
N3    346   266   4
N4    406   61    3
N5    631   79    3
N6    648   322   3
N7    558   456   3
N8    300   457   3

SPAN  O     D     LENGTH      MTBF  MTTR  FIXEDCOST   UNITCOST

S1    N1    N2    191.0418802 0     0     0     191.0418802
S2    N1    N4    231.0194797 0     0     0     231.0194797
S5    N2    N3    175.8237754 0     0     0     175.8237754
S8    N3    N8    196.4611921 0     0     0     196.4611921
S10   N4    N5    225.7188517 0     0     0     225.7188517
S11   N5    N6    243.5939244 0     0     0     243.5939244
S14   N6    N7    161.4187102 0     0     0     161.4187102
S17   N7    N8    258.001938  0     0     0     258.001938
```

Appendix 3 AMPL ILP

3.1 DSP-TR

```
# AMPL ILP Model for Demand-wise Shared Protection - Transportation Problem Approach
# August, 2008
# Copyright (C) 2008 TRLabs, Inc. All Rights Reserved.

# ******************************************************

# TRLabs
# 7th Floor
# 9107 116 Street NW
# Edmonton, Alberta, Canada
# T6G 2V4

# +1 780 441-3800
# www.trlabs.ca

# ******************************************************


# This model, including any data and algorithms contained herein, is the
# exclusive property of TRLabs, held on behalf of its sponsors. Except
# as specifically authorized in writing by TRLabs, the recipient of this
# model shall keep it confidential and shall protect it in whole or
# in part from disclosure and dissementation to all third parties.

# If any part of this model, including any data and algorithms contained
# herein, is used in any derivative works or publications, TRLabs shall be
# duly cited as a reference.

# TRlabs makes no representation or warranties about the suitability of
# this model, either express or implied, including but not limited to
# implied warranties of merchantability, fitness for a particular purpose,
# or non-infringement. TRLabs shall not be liable for any damages suffered
# as a result of using, modifying or distributing this model or its
# derivatives.



# This is an AMPL model for Demand-wise Shared Protection
# using the transportation approach.  This model produces incorrect results, and is
# based on a span centric approach

# An AMPL data file to accompany this model can be generated using
# DSP-A-TR-Datprep.exe, software written in C++.

# Explicit route representations for eligible primary and backup routes are given
# as inputs for each demand pair, and demand splitting is allowed. If a span
# failure affects only one of several working paths for a specific demand,
# the surviving working paths remain and only the affected working path is
# restored over its backup route. Stub release is not performed on surviving
```

```
# spans of an effected working path, and backup routes are NOT failure-specific.

# A given span cut is first transformed into the O-D relations affected
# and hence into the overall pattern of O-D pairs affected, with a restoration
# magnitude associated with each.

# From that point on we view the problem as a flow assignment to
# eligible restoration routes on a path replacement basis for each affected
# O-D pair. Eligible restoration routes are excluded if they are not disjoint
# from the working route.



# ************************
# TOPOLOGY DEFINITION
# ************************

set NODES;
# Set of all nodes in the network

set SPANS;
# Set of all spans in the network

set NODE_SPANS{n in NODES} within SPANS;
# Set of all spans attached to a given node

set ADJ_NODES{n in NODES} within NODES;

param Cost{i in SPANS};
# The cost of a unit of working or spare capacity on span j.

# ************************
# DESCRIPTION OF  DEMANDS AND THEIR ROUTING
# ************************

set DEMANDS;
# Set of all demands that exist. Will contains only those non-zero members as
# read in from dat file

set ORIGIN{DEMANDS} within NODES;
# The origin for demand d

set DESTINATION{DEMANDS} within NODES;
# the destination for demand d

param DemUnits{r in DEMANDS} default 0;
# Number of demand units between node pair r.

param MaxFlow := sum {r in DEMANDS} DemUnits[r];
# Used for upper bounds on flow and capacity variables.

# ************************
# VARIABLES
# ************************

var span_flow{r in DEMANDS, i in SPANS} >=0, <=MaxFlow integer;
```

```
# The amount of lightpaths assigned to span that starts at i and finishes at j.

var node_flow{r in DEMANDS, i in NODES, j in ADJ_NODES[i]} >=0, <=MaxFlow;# integer;
# Whether or not span i,j has any traffic routed on it

var node_direction {r in DEMANDS, i in NODES, j in ADJ_NODES[i]} >=0, <=1 integer;
# ensures traffic is not assigned in both directions on a span

var totalwork >=0, <= (( sum{j in SPANS} Cost[j] ) * MaxFlow);

# ************************
# OBJECTIVE FUNCTION
# ************************


minimize TotalCost: totalwork;
#minimize cost_of_mesh: sum{j in SPANS} span_flow[j]* Cost[j];



# ************************
# CONSTRAINTS
# ************************

subject to calculate_totalwork:
        totalwork = sum{r in DEMANDS, i in SPANS} (span_flow[r,i] * Cost[i]);#  + span_used[r,i] * Cost[i]);
        # total cost of spans.

subject to single_failure_restorability {r in DEMANDS, n in (ORIGIN[r] union DESTINATION[r]), j in
NODE_SPANS[n]}:
        sum{k in NODE_SPANS[n]: k<>j}span_flow[r,k] >= DemUnits[r];
        # If a single path fails (starting with span i,j), the rest of the paths must have enough capacity to route
traffic on it.

subject to translate_node_direction {r in DEMANDS, n1 in NODES, n2 in ADJ_NODES[n1]}:
    MaxFlow * node_direction[r, n1, n2] >= node_flow[r,n1, n2];

subject to node_flow_not_both_ways {r in DEMANDS, n1 in NODES, n2 in ADJ_NODES[n1]}:
    node_direction[r, n1, n2] + node_direction[r, n2, n1] <= 1;

subject to disjoint_paths_1 {r in DEMANDS, n1 in ((NODES diff ORIGIN[r]) diff DESTINATION[r])}:
    sum{n2 in ADJ_NODES[n1]} node_direction[r,n1,n2] <= 1;

subject to disjoint_paths_2 {r in DEMANDS, n1 in ((NODES diff ORIGIN[r]) diff DESTINATION[r])}:
    sum{n2 in ADJ_NODES[n1]} node_direction[r,n2,n1] <= 1;

subject to transit_flow {r in DEMANDS, n1 in ((NODES diff ORIGIN[r]) diff DESTINATION[r])}:
        sum{n2 in ADJ_NODES[n1]} node_flow[r,n1,n2] = sum{n2 in ADJ_NODES[n1]} node_flow[r,n2,n1];
        # the incoming flow and the outgoing flow must be equal

subject to origin_flow{r in DEMANDS, n1 in ORIGIN[r], n2 in ADJ_NODES[n1], i in (NODE_SPANS[n1] inter
NODE_SPANS[n2])}:
        node_flow[r,n1,n2] = span_flow[r,i];
        # If a span is connected to the origin, then the flow of traffic must be away from the origin node
```

```
subject to destination_flow{r in DEMANDS, n1 in DESTINATION[r], n2 in ADJ_NODES[n1], i in
(NODE_SPANS[n1] inter NODE_SPANS[n2])}:
        node_flow[r,n2,n1] = span_flow[r,i];
        # if a span is connected to the destination then the traffic must be toward the destination

subject to one_way_traffic{r in DEMANDS, n1 in NODES, n2 in ADJ_NODES[n1], i in (NODE_SPANS[n1] inter
NODE_SPANS[n2])}:
        node_flow[r,n1,n2] + node_flow[r,n2,n1] = span_flow[r,i];
        # traffic must only flow in one direction.  This is only figurative, since once a link is established
        # from origin to destination, it is assumed that a link in the opposite direction is included.

subject to limit_span_flow{r in DEMANDS, i in SPANS}:
    span_flow[r,i] <= DemUnits[r];
```

3.2 DSP-Top

```
# AMPL ILP Model for Demand-wise Shared Protection With Availability - Hybrid Version
# August, 2008
# Copyright (C) 2008 TRLabs, Inc. All Rights Reserved.

# ******************************************************

# TRLabs
# 7th Floor
# 9107 116 Street NW
# Edmonton, Alberta, Canada
# T6G 2V4

# +1 780 441-3800
# www.trlabs.ca

# ******************************************************

# Changes on August 20, 2008 - Took out constraints for the if then statement
# that control flow conservation this is done in the node_flow parameter
# constraints

# Changes on August 24, 2008 - The constraints for the if then statement were added
# back.  The span_used parameter for enforcing node disjointness was not adequate

# This model, including any data and algorithms contained herein, is the
# exclusive property of TRLabs, held on behalf of its sponsors. Except
# as specifically authorized in writing by TRLabs, the recipient of this
# model shall keep it confidential and shall protect it in whole or
# in part from disclosure and dissementation to all third parties.

# If any part of this model, including any data and algorithms contained
# herein, is used in any derivative works or publications, TRLabs shall be
# duly cited as a reference.

# TRlabs makes no representation or warranties about the suitability of
# this model, either express or implied, including but not limited to
# implied warranties of merchantability, fitness for a particular purpose,
# or non-infringement. TRLabs shall not be liable for any damages suffered
# as a result of using, modifying or distributing this model or its
# derivatives.



# This is an AMPL model for Demand-wise Shared Protection
# using the transportation approach.  This model produces incorrect results, and is
# based on a span centric approach

# An AMPL data file to accompany this model can be generated using
# DSP-A-TR-Datprep.exe, software written in C++.

# Explicit route representations for eligible primary and backup routes are given
# as inputs for each demand pair, and demand splitting is allowed. If a span
# failure affects only one of several working paths for a specific demand,
# the surviving working paths remain and only the affected working path is
```

```
# restored over its backup route. Stub release is not performed on surviving
# spans of an effected working path, and backup routes are NOT failure-specific.

# A given span cut is first transformed into the O-D relations affected
# and hence into the overall pattern of O-D pairs affected, with a restoration
# magnitude associated with each.

# From that point on we view the problem as a flow assignment to
# eligible restoration routes on a path replacement basis for each affected
# O-D pair. Eligible restoration routes are excluded if they are not disjoint
# from the working route.



# ************************
# TOPOLOGY DEFINITION
# ************************

set NODES;
# Set of all nodes in the network

set SPANS;
# Set of all spans in the network

set NODE_SPANS{n in NODES} within SPANS;
# Set of all spans attached to a given node

set ADJ_NODES{n in NODES} within NODES;

param Cost{i in SPANS};
# The cost of a unit of working or spare capacity on span j.

param Implement_Cost{i in SPANS};
# The cost of using at least one span in the network

# ************************
# DESCRIPTION OF  DEMANDS AND THEIR ROUTING
# ************************

set DEMANDS;
# Set of all demands that exist. Will contains only those non-zero members as
# read in from dat file

set ORIGIN{DEMANDS} within NODES;
# The origin for demand d

set DESTINATION{DEMANDS} within NODES;
# the destination for demand d

param DemUnits{r in DEMANDS} default 0;
# Number of demand units between node pair r.

param MaxFlow := sum {r in DEMANDS} DemUnits[r];
# Used for upper bounds on flow and capacity variables.
```

```
# ************************
# VARIABLES
# ************************

var span_flow{r in DEMANDS, i in SPANS} >=0, <=MaxFlow;# integer;
# The amount of lightpaths assigned to span that starts at i and finishes at j.

var span_used{i in SPANS} >=0, <=1 integer;
# The amount of lightpaths assigned to span that starts at i and finishes at j.

var node_flow{r in DEMANDS, i in NODES, j in ADJ_NODES[i]} >=0, <=MaxFlow;# integer;
# Whether or not span i,j has any traffic routed on it

var node_direction {r in DEMANDS, i in NODES, j in ADJ_NODES[i]} >=0, <=1 integer;
# ensures traffic is not assigned in both directions on a span


var totalwork >=0;

# ************************
# OBJECTIVE FUNCTION
# ************************

minimize TotalCost: totalwork;
#minimize cost_of_mesh: sum{j in SPANS} span_flow[j]* Cost[j];



# ************************
# CONSTRAINTS
# ************************

subject to calculate_totalwork:
        totalwork = (sum{r in DEMANDS, i in SPANS}span_flow[r,i] * Cost[i]) + (sum{i in SPANS}span_used[i]
* Implement_Cost[i]);
        # total cost of spans.

subject to translate_span_used {i in SPANS}:
    span_used[i] * MaxFlow * 10 >= sum{r in DEMANDS} span_flow[r,i];

subject to single_failure_restorability {r in DEMANDS, n in (ORIGIN[r] union DESTINATION[r]), j in
NODE_SPANS[n]}:
        sum{k in NODE_SPANS[n]: k<>j}span_flow[r,k] >= DemUnits[r];
        # If a single path fails (starting with span i,j), the rest of the paths must have enough capacity to route
traffic on it.

subject to translate_node_direction {r in DEMANDS, n1 in NODES, n2 in ADJ_NODES[n1]}:
    MaxFlow * node_direction[r, n1, n2] >= node_flow[r,n1, n2];

subject to node_flow_not_both_ways {r in DEMANDS, n1 in NODES, n2 in ADJ_NODES[n1]}:
    node_direction[r, n1, n2] + node_direction[r, n2, n1] <= 1;
# each span cannot have traffic in both directions

subject to disjoint_paths_1 {r in DEMANDS, n1 in ((NODES diff ORIGIN[r]) diff DESTINATION[r])}:
    sum{n2 in ADJ_NODES[n1]} node_direction[r,n1,n2] <= 1;
```

```
subject to disjoint_paths_2 {r in DEMANDS, n1 in ((NODES diff ORIGIN[r]) diff DESTINATION[r])}:
      sum{n2 in ADJ_NODES[n1]} node_direction[r,n2,n1] <= 1;

subject to direction_conservation {r in DEMANDS, n1 in ((NODES diff ORIGIN[r]) diff DESTINATION[r])}:
      sum{n2 in ADJ_NODES[n1]} node_direction[r,n1,n2] - sum{n2 in ADJ_NODES[n1]}
node_direction[r,n2,n1] = 0;

subject to transit_flow {r in DEMANDS, n1 in ((NODES diff ORIGIN[r]) diff DESTINATION[r])}:
            sum{n2 in ADJ_NODES[n1]} (node_flow[r,n1,n2] - node_flow[r,n2,n1]) = 0;
            # the incoming flow and the outgoing flow must be equal

subject to origin_flow{r in DEMANDS, n1 in ORIGIN[r], n2 in ADJ_NODES[n1], i in (NODE_SPANS[n1] inter
NODE_SPANS[n2])}:
            node_flow[r,n1,n2] = span_flow[r,i];
            # If a span is connected to the origin, then the flow of traffic must be away from the origin node

subject to destination_flow{r in DEMANDS, n1 in DESTINATION[r], n2 in ADJ_NODES[n1], i in
(NODE_SPANS[n1] inter NODE_SPANS[n2])}:
            node_flow[r,n2,n1] = span_flow[r,i];
            # if a span is connected to the destination then the traffic must be toward the destination

subject to one_way_traffic{r in DEMANDS, n1 in NODES, n2 in ADJ_NODES[n1], i in (NODE_SPANS[n1] inter
NODE_SPANS[n2])}:
            node_flow[r,n1,n2] + node_flow[r,n2,n1] = span_flow[r,i];
            # traffic must only flow in one direction.  This is only figurative, since once a link is established
            # from origin to destination, it is assumed that a link in the opposite direction is included.

subject to limit_span_flow{r in DEMANDS, i in SPANS}:
      span_flow[r,i] <= DemUnits[r];

# helper constraints
subject to no_origin_in_flow{r in DEMANDS, n1 in ORIGIN[r], n2 in ADJ_NODES[n1]}:
            node_flow[r,n2,n1] + node_direction[r,n2,n1] = 0;

subject to no_destination_out_flow{r in DEMANDS, n1 in DESTINATION[r], n2 in ADJ_NODES[n1]}:
            node_flow[r,n1,n2] + node_direction[r,n1,n2] = 0;

subject to no_path_splitting{r in DEMANDS, n1 in ORIGIN[r], n3 in DESTINATION[r]}:
            sum{n2 in ADJ_NODES[n1]} node_direction[r,n1,n2] - sum{n4 in ADJ_NODES[n3]}
node_direction[r,n4,n3] = 0;
```

3.3 DSP-TR-A

```
# AMPL ILP Model for Demand-wise Shared Protection With Availability - Hybrid Version
# August, 2008
# Copyright (C) 2008 TRLabs, Inc. All Rights Reserved.

# ******************************************************

# TRLabs
# 7th Floor
# 9107 116 Street NW
# Edmonton, Alberta, Canada
# T6G 2V4

# +1 780 441-3800
# www.trlabs.ca

# ******************************************************

# Changes on August 20, 2008 - Took out constraints for the if then statement
# that control flow conservation this is done in the node_flow parameter
# constraints

# Changes on August 24, 2008 - The constraints for the if then statement were added
# back.  The span_used parameter for enforcing node disjointness was not adequate

# This model, including any data and algorithms contained herein, is the
# exclusive property of TRLabs, held on behalf of its sponsors. Except
# as specifically authorized in writing by TRLabs, the recipient of this
# model shall keep it confidential and shall protect it in whole or
# in part from disclosure and dissementation to all third parties.

# If any part of this model, including any data and algorithms contained
# herein, is used in any derivative works or publications, TRLabs shall be
# duly cited as a reference.

# TRlabs makes no representation or warranties about the suitability of
# this model, either express or implied, including but not limited to
# implied warranties of merchantability, fitness for a particular purpose,
# or non-infringement. TRLabs shall not be liable for any damages suffered
# as a result of using, modifying or distributing this model or its
# derivatives.



# This is an AMPL model for Demand-wise Shared Protection
# using the transportation approach.  This model produces incorrect results, and is
# based on a span centric approach

# An AMPL data file to accompany this model can be generated using
# DSP-A-TR-Datprep.exe, software written in C++.

# Explicit route representations for eligible primary and backup routes are given
# as inputs for each demand pair, and demand splitting is allowed. If a span
# failure affects only one of several working paths for a specific demand,
# the surviving working paths remain and only the affected working path is
```

```
# restored over its backup route. Stub release is not performed on surviving
# spans of an effected working path, and backup routes are NOT failure-specific.

# A given span cut is first transformed into the O-D relations affected
# and hence into the overall pattern of O-D pairs affected, with a restoration
# magnitude associated with each.

# From that point on we view the problem as a flow assignment to
# eligible restoration routes on a path replacement basis for each affected
# O-D pair. Eligible restoration routes are excluded if they are not disjoint
# from the working route.



# ************************
# TOPOLOGY DEFINITION
# ************************

set NODES;
# Set of all nodes in the network

set SPANS;
# Set of all spans in the network

set NODE_SPANS{n in NODES} within SPANS;
# Set of all spans attached to a given node

set ADJ_NODES{n in NODES} within NODES;

param Cost{i in SPANS};
# The cost of a unit of working or spare capacity on span j.


# ************************
# DESCRIPTION OF  DEMANDS AND THEIR ROUTING
# ************************

set DEMANDS;
# Set of all demands that exist. Will contains only those non-zero members as
# read in from dat file

set ORIGIN{DEMANDS} within NODES;
# The origin for demand d

set DESTINATION{DEMANDS} within NODES;
# the destination for demand d

param DemUnits{r in DEMANDS} default 0;
# Number of demand units between node pair r.

param MaxFlow := sum {r in DEMANDS} DemUnits[r];
# Used for upper bounds on flow and capacity variables.


# ************************
# AVAILABILITY DEFINITIONS
```

```
# ************************

param MIN_AVAILABILITY{D in DEMANDS} default 0;
# This builds a set of minimum availabilities for each demand

param REQ_AVAILABILITY{D in DEMANDS} default 0;
# This builds a set of minimum availabilities for each demand

param MTTF{i in SPANS};
# The mean time to failure of a path used in calculating probabilities of failure, in hours

param MTTR;
# The mean time to repair is set to be 12 hours


# ************************
# VARIABLES
# ************************

var span_flow{r in DEMANDS, i in SPANS} >=0, <=MaxFlow integer;
# The amount of lightpaths assigned to span that starts at i and finishes at j.

var node_flow{r in DEMANDS, i in NODES, j in ADJ_NODES[i]} >=0, <=MaxFlow;# integer;
# Whether or not span i,j has any traffic routed on it

var node_direction {r in DEMANDS, i in NODES, j in ADJ_NODES[i]} >=0, <=1 integer;
# ensures traffic is not assigned in both directions on a span

var affected_dual_failure {r in DEMANDS, i in SPANS, j in SPANS} >= 0, <=MaxFlow;
# Counts the number of unroutable units of demand

#var affected_single_failure{r in DEMANDS, i in SPANS} >=0, <=MaxFlow integer;
# Number of unassignable demands when a single path fails

#var survivable_flow {r in DEMANDS, i in SPANS, j in SPANS} >= 0;# integer;
# the amount of lightpaths that can still be routed if spans i and j fail.

var dual_failure_span_flow {r in DEMANDS, i in SPANS, j in SPANS, k in SPANS} >= 0, <=MaxFlow;# integer;
# the amount of traffic that can be routed on span i if spans j and k fail.

var unavailability {d in DEMANDS} >=0, <=1;
# The actual unavailability for the demand.

var totalwork >=0, <= (( sum{j in SPANS} Cost[j] ) * MaxFlow);

# ************************
# OBJECTIVE FUNCTION
# ************************

minimize Demand_unavailability: sum{r in DEMANDS} 1000000*unavailability[r];

minimize TotalCost: totalwork;
#minimize cost_of_mesh: sum{j in SPANS} span_flow[j]* Cost[j];
```

```
# ************************
# CONSTRAINTS
# ************************

subject to calculate_totalwork:
	totalwork = sum{r in DEMANDS, i in SPANS} (span_flow[r,i] * Cost[i]);#  + span_used[r,i] * Cost[i]);
	# total cost of spans.

subject to single_failure_restorability {r in DEMANDS, n in (ORIGIN[r] union DESTINATION[r]), j in
NODE_SPANS[n]}:
	sum{k in NODE_SPANS[n]: k<>j}span_flow[r,k] >= DemUnits[r];
	# If a single path fails (starting with span i,j), the rest of the paths must have enough capacity to route
traffic on it.

subject to translate_node_direction {r in DEMANDS, n1 in NODES, n2 in ADJ_NODES[n1]}:
	MaxFlow * node_direction[r, n1, n2] >= node_flow[r,n1, n2];

subject to node_flow_not_both_ways {r in DEMANDS, n1 in NODES, n2 in ADJ_NODES[n1]}:
	node_direction[r, n1, n2] + node_direction[r, n2, n1] <= 1;
# each span cannot have traffic in both directions

subject to disjoint_paths_1 {r in DEMANDS, n1 in ((NODES diff ORIGIN[r]) diff DESTINATION[r])}:
	sum{n2 in ADJ_NODES[n1]} node_direction[r,n1,n2] <= 1;

subject to disjoint_paths_2 {r in DEMANDS, n1 in ((NODES diff ORIGIN[r]) diff DESTINATION[r])}:
	sum{n2 in ADJ_NODES[n1]} node_direction[r,n2,n1] <= 1;

#subject to direction_conservation {r in DEMANDS, n1 in ((NODES diff ORIGIN[r]) diff DESTINATION[r])}:
#	sum{n2 in ADJ_NODES[n1]} node_direction[r,n1,n2] - sum{n2 in ADJ_NODES[n1]}
node_direction[r,n2,n1] = 0;

subject to transit_flow {r in DEMANDS, n1 in ((NODES diff ORIGIN[r]) diff DESTINATION[r])}:
	sum{n2 in ADJ_NODES[n1]} node_flow[r,n1,n2] = sum{n2 in ADJ_NODES[n1]} node_flow[r,n2,n1];
	# the incoming flow and the outgoing flow must be equal

subject to origin_flow{r in DEMANDS, n1 in ORIGIN[r], n2 in ADJ_NODES[n1], i in (NODE_SPANS[n1] inter
NODE_SPANS[n2])}:
	node_flow[r,n1,n2] = span_flow[r,i];
	# If a span is connected to the origin, then the flow of traffic must be away from the origin node

subject to destination_flow{r in DEMANDS, n1 in DESTINATION[r], n2 in ADJ_NODES[n1], i in
(NODE_SPANS[n1] inter NODE_SPANS[n2])}:
	node_flow[r,n2,n1] = span_flow[r,i];
	# if a span is connected to the destination then the traffic must be toward the destination

subject to one_way_traffic{r in DEMANDS, n1 in NODES, n2 in ADJ_NODES[n1], i in (NODE_SPANS[n1] inter
NODE_SPANS[n2])}:
	node_flow[r,n1,n2] + node_flow[r,n2,n1] = span_flow[r,i];
	# traffic must only flow in one direction.  This is only figurative, since once a link is established
	# from origin to destination, it is assumed that a link in the opposite direction is included.

subject to limit_span_flow{r in DEMANDS, i in SPANS}:
	span_flow[r,i] <= DemUnits[r];

# Availability Calculations
```

```
subject to calculate_unavailability{r in DEMANDS}:
	#(sum{i in SPANS} (MTTR / (MTTF[i] + MTTR) * affected_single_failure[r,i]) +
		sum{s1 in SPANS, s2 in SPANS}( MTTR / (MTTF[s1] + MTTR) * (1- exp(-MTTR/MTTF[s2]))
			* affected_dual_failure[r,s1,s2]) / DemUnits[r]
		<= unavailability[r];
	#calculates the unavailability of a demand

subject to availability{r in DEMANDS}:
	100000*(1-unavailability[r]) >= 100000*(REQ_AVAILABILITY[r]);
	#ensures each demand meets the given availability requirements

subject to calculate_dual_affected{r in DEMANDS, n in ORIGIN[r], s1 in SPANS, s2 in SPANS}:
      affected_dual_failure[r,s1,s2] >= DemUnits[r] - sum{i in NODE_SPANS[n]}
dual_failure_span_flow[r,i,s1,s2];

subject to calculate_dual_failure_span_flow{r in DEMANDS, i in SPANS, s1 in SPANS, s2 in SPANS}:
	dual_failure_span_flow[r,i,s1,s2] <= span_flow[r,i];

subject to no_flow_on_failed_spans_1{r in DEMANDS, s1 in SPANS, s2 in SPANS}:
	dual_failure_span_flow[r,s1,s1,s2] + dual_failure_span_flow[r,s2,s1,s2] = 0;

subject to no_flow_on_failed_spans_2{r in DEMANDS, s1 in SPANS, s2 in SPANS}:
	dual_failure_span_flow[r,s2,s1,s2] = 0;

subject to dual_failure_flow_conservation {r in DEMANDS, n in ((NODES diff ORIGIN[r]) diff
DESTINATION[r]), i in NODE_SPANS[n], s1 in SPANS, s2 in SPANS}:
	dual_failure_span_flow[r,i,s1,s2] - sum{j in NODE_SPANS[n]:j<>i}dual_failure_span_flow[r,j,s1,s2] <=
0;
```

3.4 DSP-Top-A

```
# AMPL ILP Model for Demand-wise Shared Protection With Availability - Hybrid Version
# August, 2008
# Copyright (C) 2008 TRLabs, Inc. All Rights Reserved.

# ******************************************************

# TRLabs
# 7th Floor
# 9107 116 Street NW
# Edmonton, Alberta, Canada
# T6G 2V4

# +1 780 441-3800
# www.trlabs.ca

# ******************************************************

# Changes on August 20, 2008 - Took out constraints for the if then statement
# that control flow conservation this is done in the node_flow parameter
# constraints

# Changes on August 24, 2008 - The constraints for the if then statement were added
# back.  The span_used parameter for enforcing node disjointness was not adequate

# This model, including any data and algorithms contained herein, is the
# exclusive property of TRLabs, held on behalf of its sponsors. Except
# as specifically authorized in writing by TRLabs, the recipient of this
# model shall keep it confidential and shall protect it in whole or
# in part from disclosure and dissementation to all third parties.

# If any part of this model, including any data and algorithms contained
# herein, is used in any derivative works or publications, TRLabs shall be
# duly cited as a reference.

# TRlabs makes no representation or warranties about the suitability of
# this model, either express or implied, including but not limited to
# implied warranties of merchantability, fitness for a particular purpose,
# or non-infringement. TRLabs shall not be liable for any damages suffered
# as a result of using, modifying or distributing this model or its
# derivatives.



# This is an AMPL model for Demand-wise Shared Protection
# using the transportation approach.  This model produces incorrect results, and is
# based on a span centric approach

# An AMPL data file to accompany this model can be generated using
# DSP-A-TR-Datprep.exe, software written in C++.

# Explicit route representations for eligible primary and backup routes are given
# as inputs for each demand pair, and demand splitting is allowed. If a span
# failure affects only one of several working paths for a specific demand,
# the surviving working paths remain and only the affected working path is
```

```
# restored over its backup route. Stub release is not performed on surviving
# spans of an effected working path, and backup routes are NOT failure-specific.

# A given span cut is first transformed into the O-D relations affected
# and hence into the overall pattern of O-D pairs affected, with a restoration
# magnitude associated with each.

# From that point on we view the problem as a flow assignment to
# eligible restoration routes on a path replacement basis for each affected
# O-D pair. Eligible restoration routes are excluded if they are not disjoint
# from the working route.



# ************************
# TOPOLOGY DEFINITION
# ************************

set NODES;
# Set of all nodes in the network

set SPANS;
# Set of all spans in the network

set NODE_SPANS{n in NODES} within SPANS;
# Set of all spans attached to a given node

set ADJ_NODES{n in NODES} within NODES;

param Cost{i in SPANS};
# The cost of a unit of working or spare capacity on span j.

param Implement_Cost{i in SPANS};
# The cost of using at least one span in the network

# ************************
# DESCRIPTION OF  DEMANDS AND THEIR ROUTING
# ************************

set DEMANDS;
# Set of all demands that exist. Will contains only those non-zero members as
# read in from dat file

set ORIGIN{DEMANDS} within NODES;
# The origin for demand d

set DESTINATION{DEMANDS} within NODES;
# the destination for demand d

param DemUnits{r in DEMANDS} default 0;
# Number of demand units between node pair r.

param MaxFlow := sum {r in DEMANDS} DemUnits[r];
# Used for upper bounds on flow and capacity variables.
```

```
# ************************
# AVAILABILITY DEFINITIONS
# ************************

param MIN_AVAILABILITY{D in DEMANDS} default 0;
# This builds a set of minimum availabilities for each demand

param REQ_AVAILABILITY{D in DEMANDS} default 0;
# This builds a set of minimum availabilities for each demand

param MTTF{i in SPANS};
# The mean time to failure of a path used in calculating probabilities of failure, in hours

param MTTR;
# The mean time to repair is set to be 12 hours


# ************************
# VARIABLES
# ************************

var span_flow{r in DEMANDS, i in SPANS} >=0, <=MaxFlow integer;
# The amount of lightpaths assigned to span that starts at i and finishes at j.

var span_used{i in SPANS} >=0, <=1 integer;
# The amount of lightpaths assigned to span that starts at i and finishes at j.

var node_flow{r in DEMANDS, i in NODES, j in ADJ_NODES[i]} >=0, <=MaxFlow;# integer;
# Whether or not span i,j has any traffic routed on it

var node_direction {r in DEMANDS, i in NODES, j in ADJ_NODES[i]} >=0, <=1 integer;
# ensures traffic is not assigned in both directions on a span

var affected_dual_failure {r in DEMANDS, i in SPANS, j in SPANS} >= 0, <=MaxFlow;
# Counts the number of unroutable units of demand

#var affected_single_failure{r in DEMANDS, i in SPANS} >=0, <=MaxFlow integer;
# Number of unassignable demands when a single path fails

var survivable_flow {r in DEMANDS, i in SPANS, j in SPANS} >= 0;# integer;
# the amount of lightpaths that can still be routed if spans i and j fail.

var dual_failure_span_flow {r in DEMANDS, i in SPANS, j in SPANS, k in SPANS} >= 0, <=MaxFlow;# integer;
# the amount of traffic that can be routed on span i if spans j and k fail.

var unavailability {d in DEMANDS} >=0, <=1;
# The actual unavailability for the demand.

var totalwork >=0, <= (( sum{j in SPANS} Cost[j] ) * MaxFlow);

#var y{r in DEMANDS, n in NODES, i in NODE_SPANS[n]} >=0, <=1, integer;


# ************************
# OBJECTIVE FUNCTION
# ************************
```

```
minimize Demand_unavailability: sum{r in DEMANDS} 1000000*unavailability[r];

minimize TotalCost: totalwork;
#minimize cost_of_mesh: sum{j in SPANS} span_flow[j]* Cost[j];

minimize paths_in_dual_failure_restorable: sum{r in DEMANDS, n1 in ORIGIN[r], n2 in ADJ_NODES[n1]}
node_flow[r,n1, n2];



# ************************
# CONSTRAINTS
# ************************

subject to dual_failure_restorability {r in DEMANDS, n1 in (ORIGIN[r])}:
      sum{n2 in ADJ_NODES[n1]}node_flow[r,n1,n2] >= 3;
      #forces at least 3 paths per connection

subject to calculate_totalwork:
          totalwork = (sum{r in DEMANDS, i in SPANS}span_flow[r,i] * Cost[i]) + (sum{i in SPANS}span_used[i]
* Implement_Cost[i]);
          # total cost of spans.

subject to translate_span_used {i in SPANS}:
      span_used[i] * MaxFlow * 10 >= sum{r in DEMANDS} span_flow[r,i];

subject to single_failure_restorability {r in DEMANDS, n in (ORIGIN[r] union DESTINATION[r]), j in
NODE_SPANS[n]}:
          sum{k in NODE_SPANS[n]: k<>j}span_flow[r,k] >= DemUnits[r];
          # If a single path fails (starting with span i,j), the rest of the paths must have enough capacity to route
traffic on it.

subject to translate_node_direction {r in DEMANDS, n1 in NODES, n2 in ADJ_NODES[n1]}:
      MaxFlow * node_direction[r, n1, n2] >= node_flow[r,n1, n2];

subject to node_flow_not_both_ways {r in DEMANDS, n1 in NODES, n2 in ADJ_NODES[n1]}:
      node_direction[r, n1, n2] + node_direction[r, n2, n1] <= 1;
# each span cannot have traffic in both directions

subject to disjoint_paths_1 {r in DEMANDS, n1 in ((NODES diff ORIGIN[r]) diff DESTINATION[r])}:
      sum{n2 in ADJ_NODES[n1]} node_direction[r,n1,n2] <= 1;

subject to disjoint_paths_2 {r in DEMANDS, n1 in ((NODES diff ORIGIN[r]) diff DESTINATION[r])}:
      sum{n2 in ADJ_NODES[n1]} node_direction[r,n2,n1] <= 1;

subject to direction_conservation {r in DEMANDS, n1 in ((NODES diff ORIGIN[r]) diff DESTINATION[r])}:
      sum{n2 in ADJ_NODES[n1]} node_direction[r,n1,n2] - sum{n2 in ADJ_NODES[n1]}
node_direction[r,n2,n1] = 0;

subject to transit_flow {r in DEMANDS, n1 in ((NODES diff ORIGIN[r]) diff DESTINATION[r])}:
          sum{n2 in ADJ_NODES[n1]} node_flow[r,n1,n2] = sum{n2 in ADJ_NODES[n1]} node_flow[r,n2,n1];
          # the incoming flow and the outgoing flow must be equal

subject to origin_flow{r in DEMANDS, n1 in ORIGIN[r], n2 in ADJ_NODES[n1], i in (NODE_SPANS[n1] inter
NODE_SPANS[n2])}:
```

```
        node_flow[r,n1,n2] = span_flow[r,i];
        # If a span is connected to the origin, then the flow of traffic must be away from the origin node

subject to destination_flow{r in DEMANDS, n1 in DESTINATION[r], n2 in ADJ_NODES[n1], i in
(NODE_SPANS[n1] inter NODE_SPANS[n2])}:
        node_flow[r,n2,n1] = span_flow[r,i];
        # if a span is connected to the destination then the traffic must be toward the destination

subject to one_way_traffic{r in DEMANDS, n1 in NODES, n2 in ADJ_NODES[n1], i in (NODE_SPANS[n1] inter
NODE_SPANS[n2])}:
        node_flow[r,n1,n2] + node_flow[r,n2,n1] = span_flow[r,i];
        # traffic must only flow in one direction.  This is only figurative, since once a link is established
        # from origin to destination, it is assumed that a link in the opposite direction is included.

subject to limit_span_flow{r in DEMANDS, i in SPANS}:
    span_flow[r,i] <= DemUnits[r];


# Availability Calculations

subject to calculate_unavailability{r in DEMANDS}:
        #(sum{i in SPANS} (MTTR / (MTTF[i] + MTTR) * affected_single_failure[r,i]) +
                sum{s1 in SPANS, s2 in SPANS}( MTTR / (MTTF[s1] + MTTR) * (1- exp(-MTTR/MTTF[s2]))
                        * affected_dual_failure[r,s1,s2]) / DemUnits[r]
                <= unavailability[r];
        #calculates the unavailability of a demand

subject to availability{r in DEMANDS}:
        100000*(1-unavailability[r]) >= 100000*(REQ_AVAILABILITY[r]);
        #ensures each demand meets the given availability requirements

subject to calculate_single_affected{r in DEMANDS, i in SPANS}:
        DemUnits[r] - survivable_flow[r,i,i] <= affected_single_failure[r,i];
        #calculates how many demand units won't be routed if p1 fails

subject to calculate_dual_affected{r in DEMANDS, s1 in SPANS, s2 in SPANS}:# : s1 <> s2}:
    DemUnits[r] - survivable_flow[r,s1,s2] <= affected_dual_failure[r,s1,s2];
        #calculates how many demand units won't be routed if p1 and p2 fail

subject to calculate_survivable_flow{r in DEMANDS, n in ORIGIN[r], s1 in SPANS, s2 in SPANS}:
    survivable_flow[r,s1,s2] = sum{i in NODE_SPANS[n]} dual_failure_span_flow[r,i,s1,s2];

subject to calculate_dual_failure_span_flow{r in DEMANDS, i in SPANS, s1 in SPANS, s2 in SPANS}:
        dual_failure_span_flow[r,i,s1,s2] <= span_flow[r,i];

subject to no_flow_on_failed_spans_1{r in DEMANDS, s1 in SPANS, s2 in SPANS}:
        dual_failure_span_flow[r,s1,s1,s2] + dual_failure_span_flow[r,s2,s1,s2] = 0;

subject to no_flow_on_failed_spans_2{r in DEMANDS, s1 in SPANS, s2 in SPANS}:
        dual_failure_span_flow[r,s2,s1,s2] = 0;

subject to dual_failure_flow_conservation {r in DEMANDS, n in ((NODES diff ORIGIN[r]) diff
DESTINATION[r]), i in NODE_SPANS[n], s1 in SPANS, s2 in SPANS}:
        dual_failure_span_flow[r,i,s1,s2] - sum{j in NODE_SPANS[n]:j<>i}dual_failure_span_flow[r,j,s1,s2] <=
0;
```

3.5 DSP-TR-R2

```
# AMPL ILP Model for Demand-wise Shared Protection With Dual Failure Restorability
# August, 2008
# Copyright (C) 2008 TRLabs, Inc. All Rights Reserved.

# ******************************************************

# TRLabs
# 7th Floor
# 9107 116 Street NW
# Edmonton, Alberta, Canada
# T6G 2V4

# +1 780 441-3800
# www.trlabs.ca

# ******************************************************


# This model, including any data and algorithms contained herein, is the
# exclusive property of TRLabs, held on behalf of its sponsors. Except
# as specifically authorized in writing by TRLabs, the recipient of this
# model shall keep it confidential and shall protect it in whole or
# in part from disclosure and dissementation to all third parties.

# If any part of this model, including any data and algorithms contained
# herein, is used in any derivative works or publications, TRLabs shall be
# duly cited as a reference.

# TRlabs makes no representation or warranties about the suitability of
# this model, either express or implied, including but not limited to
# implied warranties of merchantability, fitness for a particular purpose,
# or non-infringement. TRLabs shall not be liable for any damages suffered
# as a result of using, modifying or distributing this model or its
# derivatives.



# This is an AMPL model for Demand-wise Shared Protection
# using the transportation approach.

# An AMPL data file to accompany this model can be generated using
# DSP-A-TR-Datprep.exe, software written in C++.

# Explicit route representations for eligible primary and backup routes are given
# as inputs for each demand pair, and demand splitting is allowed. If a span
# failure affects only one of several working paths for a specific demand,
# the surviving working paths remain and only the affected working path is
# restored over its backup route. Stub release is not performed on surviving
# spans of an effected working path, and backup routes are NOT failure-specific.

# A given span cut is first transformed into the O-D relations affected
# and hence into the overall pattern of O-D pairs affected, with a restoration
# magnitude associated with each.
```

```
# From that point on we view the problem as a flow assignment to
# eligible restoration routes on a path replacement basis for each affected
# O-D pair. Eligible restoration routes are excluded if they are not disjoint
# from the working route.



# ************************
# TOPOLOGY DEFINITION
# ************************

set NODES;
# Set of all nodes in the network

set SPANS;
# Set of all spans in the network

set NODE_SPANS{n in NODES} within SPANS;
# Set of all spans attached to a given node

set ADJ_NODES{n in NODES} within NODES;

param Cost{i in SPANS};
# The cost of a unit of working or spare capacity on span j.

# ************************
# DESCRIPTION OF  DEMANDS AND THEIR ROUTING
# ************************

set DEMANDS;
# Set of all demands that exist. Will contains only those non-zero members as
# read in from dat file

set ORIGIN{DEMANDS} within NODES;
# The origin for demand d

set DESTINATION{DEMANDS} within NODES;
# the destination for demand d

param DemUnits{r in DEMANDS} default 0;
# Number of demand units between node pair r.

param MaxFlow := sum {r in DEMANDS} DemUnits[r];
# Used for upper bounds on flow and capacity variables.

# ************************
# RESTORABILITY DEFINITIONS
# ************************

param Req_R2_Restorability{r in DEMANDS} default 0;
# This builds a set of minimum availabilities for each demand


# ************************
# VARIABLES
# ************************
```

```
var span_flow{r in DEMANDS, i in SPANS} >= 0;

var node_flow{r in DEMANDS, i in NODES, j in ADJ_NODES[i]} >=0, <=MaxFlow integer;
# Whether or not span i,j has any traffic routed on it

var node_direction {r in DEMANDS, i in NODES, j in ADJ_NODES[i]} >=0, <=1 integer;
# ensures traffic is not assigned in both directions on a span

var R2 {r in DEMANDS} >=0, <=1;


var totalwork >=0, <= (( sum{j in SPANS} Cost[j] ) * MaxFlow);

# ************************
# OBJECTIVE FUNCTION
# ************************

minimize TotalCost: totalwork;


#minimize cost_of_mesh: sum{j in SPANS} span_flow[j]* Cost[j];



# ************************
# CONSTRAINTS
# ************************



subject to calculate_totalwork:
        totalwork = sum{r in DEMANDS, n1 in NODES, n2 in ADJ_NODES[n1], i in (NODE_SPANS[n1] inter
NODE_SPANS[n2])} (node_flow[r,n1,n2] * Cost[i]);#  + span_used[r,i] * Cost[i]);
        # total cost of spans.

subject to calculate_span_flow {r in DEMANDS, n1 in NODES, n2 in ADJ_NODES[n1], i in (NODE_SPANS[n1]
inter NODE_SPANS[n2])}:
    span_flow[r,i] >= node_flow[r,n1,n2] + node_flow[r,n2,n1];

subject to single_failure_restorability {r in DEMANDS, n1 in (ORIGIN[r]), n2 in ADJ_NODES[n1]}:
        sum{n3 in ADJ_NODES[n1]: n3<>n2}node_flow[r,n1,n3] >= DemUnits[r];
        # If a single path fails (starting with span i,j), the rest of the paths must have enough capacity to route
traffic on it.

subject to calculate_dual_failure_restorability {r in DEMANDS, n1 in ORIGIN[r], n2 in ADJ_NODES[n1], n3 in
ADJ_NODES[n1]:n3<>n2}:
    sum{n4 in ADJ_NODES[n1]:n4<>n2 and n4<>n3}node_flow[r,n1,n4] / DemUnits[r] >= R2[r];

subject to enforce_restorability {r in DEMANDS}:
        R2[r] >= Req_R2_Restorability[r];

subject to translate_node_direction {r in DEMANDS, n1 in NODES, n2 in ADJ_NODES[n1]}:
    MaxFlow * 10 * node_direction[r, n1, n2] >= node_flow[r,n1, n2];

subject to node_flow_not_both_ways {r in DEMANDS, n1 in NODES, n2 in ADJ_NODES[n1]}:
    node_direction[r, n1, n2] + node_direction[r, n2, n1] <= 1;
```

```
subject to disjoint_paths_1 {r in DEMANDS, n1 in ((NODES diff ORIGIN[r]) diff DESTINATION[r])}:
      sum{n2 in ADJ_NODES[n1]} node_direction[r,n1,n2] <= 1;

subject to disjoint_paths_2 {r in DEMANDS, n1 in ((NODES diff ORIGIN[r]) diff DESTINATION[r])}:
      sum{n2 in ADJ_NODES[n1]} node_direction[r,n2,n1] <= 1;

subject to transit_flow {r in DEMANDS, n1 in ((NODES diff ORIGIN[r]) diff DESTINATION[r])}:
            sum{n2 in ADJ_NODES[n1]} node_flow[r,n1,n2] = sum{n2 in ADJ_NODES[n1]} node_flow[r,n2,n1];
            # the incoming flow and the outgoing flow must be equal

subject to limit_span_flow{r in DEMANDS, n1 in NODES, n2 in ADJ_NODES[n1]}:
      node_flow[r,n1,n2] <= DemUnits[r];

# helper constraints
subject to no_origin_in_flow{r in DEMANDS, n1 in ORIGIN[r], n2 in ADJ_NODES[n1]}:
            node_flow[r,n2,n1] + node_direction[r,n2,n1] = 0;

subject to no_destination_out_flow{r in DEMANDS, n1 in DESTINATION[r], n2 in ADJ_NODES[n1]}:
            node_flow[r,n1,n2] + node_direction[r,n1,n2] = 0;

subject to no_path_splitting{r in DEMANDS, n1 in ORIGIN[r], n3 in DESTINATION[r]}:
            sum{n2 in ADJ_NODES[n1]} node_direction[r,n1,n2] - sum{n4 in ADJ_NODES[n3]}
node_direction[r,n4,n3] = 0;
```

3.6 DSP-Top-R2

```
# AMPL ILP Model for Demand-wise Shared Protection With Topology Design and Dual failure restorability
# August, 2008
# Copyright (C) 2008 TRLabs, Inc. All Rights Reserved.

# ******************************************************

# TRLabs
# 7th Floor
# 9107 116 Street NW
# Edmonton, Alberta, Canada
# T6G 2V4

# +1 780 441-3800
# www.trlabs.ca

# ******************************************************


# This model, including any data and algorithms contained herein, is the
# exclusive property of TRLabs, held on behalf of its sponsors. Except
# as specifically authorized in writing by TRLabs, the recipient of this
# model shall keep it confidential and shall protect it in whole or
# in part from disclosure and dissementation to all third parties.

# If any part of this model, including any data and algorithms contained
# herein, is used in any derivative works or publications, TRLabs shall be
# duly cited as a reference.

# TRlabs makes no representation or warranties about the suitability of
# this model, either express or implied, including but not limited to
# implied warranties of merchantability, fitness for a particular purpose,
# or non-infringement. TRLabs shall not be liable for any damages suffered
# as a result of using, modifying or distributing this model or its
# derivatives.



# This is an AMPL model for Demand-wise Shared Protection
# using the transportation approach.

# An AMPL data file to accompany this model can be generated using
# DSP-A-TR-Datprep.exe, software written in C++.

# Explicit route representations for eligible primary and backup routes are given
# as inputs for each demand pair, and demand splitting is allowed. If a span
# failure affects only one of several working paths for a specific demand,
# the surviving working paths remain and only the affected working path is
# restored over its backup route. Stub release is not performed on surviving
# spans of an effected working path, and backup routes are NOT failure-specific.

# A given span cut is first transformed into the O-D relations affected
# and hence into the overall pattern of O-D pairs affected, with a restoration
# magnitude associated with each.
```

```
# From that point on we view the problem as a flow assignment to
# eligible restoration routes on a path replacement basis for each affected
# O-D pair. Eligible restoration routes are excluded if they are not disjoint
# from the working route.



# ************************
# TOPOLOGY DEFINITION
# ************************

set NODES;
# Set of all nodes in the network

set SPANS;
# Set of all spans in the network

set NODE_SPANS{n in NODES} within SPANS;
# Set of all spans attached to a given node

set ADJ_NODES{n in NODES} within NODES;

param Cost{i in SPANS};
# The cost of a unit of working or spare capacity on span j.

param Implement_Cost{i in SPANS};
# The cost of using at least one span in the network

# ************************
# DESCRIPTION OF  DEMANDS AND THEIR ROUTING
# ************************

set DEMANDS;
# Set of all demands that exist. Will contains only those non-zero members as
# read in from dat file

set ORIGIN{DEMANDS} within NODES;
# The origin for demand d

set DESTINATION{DEMANDS} within NODES;
# the destination for demand d

param DemUnits{r in DEMANDS} default 0;
# Number of demand units between node pair r.

param MaxFlow := sum {r in DEMANDS} DemUnits[r];
# Used for upper bounds on flow and capacity variables.

# ************************
# RESTORABILITY DEFINITIONS
# ************************

param Req_R2_Restorability{r in DEMANDS} default 0;
# This builds a set of minimum availabilities for each demand

# ************************
```

```
# VARIABLES
# ************************

var span_flow{r in DEMANDS, i in SPANS} >=0, <=MaxFlow;# integer;
# The amount of lightpaths assigned to span that starts at i and finishes at j.

var span_used{i in SPANS} >=0, <=1 integer;
# The amount of lightpaths assigned to span that starts at i and finishes at j.

var node_flow{r in DEMANDS, i in NODES, j in ADJ_NODES[i]} >=0, <=MaxFlow;# integer;
# Whether or not span i,j has any traffic routed on it

var node_direction {r in DEMANDS, i in NODES, j in ADJ_NODES[i]} >=0, <=1 integer;
# ensures traffic is not assigned in both directions on a span

var R2 {r in DEMANDS} >=0, <=DemUnits[r];
# The actual unavailability for the demand.

var totalwork >=0, <= (( sum{j in SPANS} Cost[j] ) * MaxFlow);

# ************************
# OBJECTIVE FUNCTION
# ************************
minimize TotalCost: totalwork;
maximize Total_R2: sum{r in DEMANDS}R2[r];

#minimize cost_of_mesh: sum{j in SPANS} span_flow[j]* Cost[j];



# ************************
# CONSTRAINTS
# ************************

subject to calculate_dual_failure_restorability {r in DEMANDS, n in ORIGIN[r], s1 in NODE_SPANS[n], s2 in
NODE_SPANS[n]:s2<>s1}:
      sum{s3 in NODE_SPANS[n]: s3<>s1 and s3<>s2}span_flow[r,s3] >= R2[r];

subject to enforce_restorability {r in DEMANDS}:
            R2[r] >= Req_R2_Restorability[r];

subject to calculate_totalwork:
            totalwork = (sum{r in DEMANDS, i in SPANS}span_flow[r,i] * Cost[i]) + (sum{i in SPANS}span_used[i]
* Implement_Cost[i]);
            # total cost of spans.

subject to translate_span_used {i in SPANS}:
      span_used[i] * MaxFlow * 10 >= sum{r in DEMANDS} span_flow[r,i];

subject to single_failure_restorability {r in DEMANDS, n in (ORIGIN[r] union DESTINATION[r]), j in
NODE_SPANS[n]}:
            sum{k in NODE_SPANS[n]: k<>j}span_flow[r,k] >= DemUnits[r];
            # If a single path fails (starting with span i,j), the rest of the paths must have enough capacity to route
traffic on it.

subject to translate_node_direction {r in DEMANDS, n1 in NODES, n2 in ADJ_NODES[n1]}:
```

```
	MaxFlow * node_direction[r, n1, n2] >= node_flow[r,n1, n2];

subject to node_flow_not_both_ways {r in DEMANDS, n1 in NODES, n2 in ADJ_NODES[n1]}:
	node_direction[r, n1, n2] + node_direction[r, n2, n1] <= 1;
# each span cannot have traffic in both directions

subject to disjoint_paths_1 {r in DEMANDS, n1 in ((NODES diff ORIGIN[r]) diff DESTINATION[r])}:
	sum{n2 in ADJ_NODES[n1]} node_direction[r,n1,n2] <= 1;

subject to disjoint_paths_2 {r in DEMANDS, n1 in ((NODES diff ORIGIN[r]) diff DESTINATION[r])}:
	sum{n2 in ADJ_NODES[n1]} node_direction[r,n2,n1] <= 1;

#subject to direction_conservation {r in DEMANDS, n1 in ((NODES diff ORIGIN[r]) diff DESTINATION[r])}:
#	sum{n2 in ADJ_NODES[n1]} node_direction[r,n1,n2] - sum{n2 in ADJ_NODES[n1]}
node_direction[r,n2,n1] = 0;

subject to transit_flow {r in DEMANDS, n1 in ((NODES diff ORIGIN[r]) diff DESTINATION[r])}:
		sum{n2 in ADJ_NODES[n1]} (node_flow[r,n1,n2] - node_flow[r,n2,n1]) = 0;
		# the incoming flow and the outgoing flow must be equal

subject to origin_flow{r in DEMANDS, n1 in ORIGIN[r], n2 in ADJ_NODES[n1], i in (NODE_SPANS[n1] inter
NODE_SPANS[n2])}:
		node_flow[r,n1,n2] = span_flow[r,i];
		# If a span is connected to the origin, then the flow of traffic must be away from the origin node

subject to destination_flow{r in DEMANDS, n1 in DESTINATION[r], n2 in ADJ_NODES[n1], i in
(NODE_SPANS[n1] inter NODE_SPANS[n2])}:
		node_flow[r,n2,n1] = span_flow[r,i];
		# if a span is connected to the destination then the traffic must be toward the destination

subject to one_way_traffic{r in DEMANDS, n1 in NODES, n2 in ADJ_NODES[n1], i in (NODE_SPANS[n1] inter
NODE_SPANS[n2])}:
		node_flow[r,n1,n2] + node_flow[r,n2,n1] = span_flow[r,i];
		# traffic must only flow in one direction.  This is only figurative, since once a link is established
		# from origin to destination, it is assumed that a link in the opposite direction is included.

subject to limit_span_flow{r in DEMANDS, i in SPANS}:
	span_flow[r,i] <= DemUnits[r];

# helper constraints
subject to no_origin_in_flow{r in DEMANDS, n1 in ORIGIN[r], n2 in ADJ_NODES[n1]}:
		node_flow[r,n2,n1] + node_direction[r,n2,n1] = 0;

subject to no_destination_out_flow{r in DEMANDS, n1 in DESTINATION[r], n2 in ADJ_NODES[n1]}:
		node_flow[r,n1,n2] + node_direction[r,n1,n2] = 0;

subject to no_path_splitting{r in DEMANDS, n1 in ORIGIN[r], n3 in DESTINATION[r]}:
		sum{n2 in ADJ_NODES[n1]} node_direction[r,n1,n2] - sum{n4 in ADJ_NODES[n3]}
node_direction[r,n4,n3] = 0;
```

Appendix 4 Sample Dat Files

4.1 DSP-TR – 8 Node 9 Span Network Data File

```
# AMPL ILP Model for Demand-wise Shared
Protection - Version 2.0.


# Contact btodd@trlabs.ca for more information.
# Copyright TRLabs 2008, All Rights Reserved.

# Command Line Used:
# DSP-TR-dat-prep.exe
#   ../../../../Networks/8n16s1-Family/8n16s1-9s.top
../../../../Networks/8n16s1-Family/8n16s1.dem
8n16s1-9s.dat


set NODES := N1 N2 N3 N4 N5 N6 N7 N8;
set SPANS := S1 S2 S3 S4 S5 S6 S7 S8 S11;
set NODE_SPANS[N1] := S1 S7;
set NODE_SPANS[N2] := S6 S7 S8;
set NODE_SPANS[N3] := S8 S11;
set NODE_SPANS[N4] := S1 S2;
set NODE_SPANS[N5] := S2 S3;
set NODE_SPANS[N6] := S3 S4;
set NODE_SPANS[N7] := S4 S5 S11;
set NODE_SPANS[N8] := S5 S6;;
set ADJ_NODES[N1] := N4 N2;
set ADJ_NODES[N2] := N8 N1 N3;
set ADJ_NODES[N3] := N2 N7;
set ADJ_NODES[N4] := N1 N5;
set ADJ_NODES[N5] := N4 N6;
set ADJ_NODES[N6] := N5 N7;
set ADJ_NODES[N7] := N6 N8 N3;
set ADJ_NODES[N8] := N7 N2;;

param Cost :=
S1      231.019
S2      225.719
S3      243.594
S4      161.419
S5      258.002
S6      244.755
S7      191.042
S8      175.824
S11     284.682;

set DEMANDS := D1 D2 D3 D4 D5 D6 D7 D8 D9
D10 D11 D12 D13 D14 D15 D16 D17 D18 D19 D20
D21 D22 D23 D24 D25 D26 D27 D28;

param DemUnits :=
D1      3.000000
D2      9.000000
D3      5.000000
D4      10.000000
D5      6.000000
D6      1.000000
D7      8.000000
D8      3.000000
D9      5.000000
D10     4.000000
D11     9.000000
D12     4.000000
D13     10.000000
D14     6.000000
D15     4.000000
D16     2.000000
D17     9.000000
D18     10.000000
D19     8.000000
D20     9.000000
D21     4.000000
D22     9.000000
D23     6.000000
D24     1.000000
D25     2.000000
D26     3.000000
D27     7.000000
D28     9.000000;

set ORIGIN[D1] := N1 ;
set ORIGIN[D2] := N1 ;
set ORIGIN[D3] := N1 ;
set ORIGIN[D4] := N1 ;
set ORIGIN[D5] := N1 ;
set ORIGIN[D6] := N1 ;
set ORIGIN[D7] := N1 ;
set ORIGIN[D8] := N2 ;
set ORIGIN[D9] := N2 ;
set ORIGIN[D10] := N2;
set ORIGIN[D11] := N2 ;
```

```
set ORIGIN[D12] := N2 ;
set ORIGIN[D13] := N2 ;
set ORIGIN[D14] := N3 ;
set ORIGIN[D15] := N3 ;
set ORIGIN[D16] := N3 ;
set ORIGIN[D17] := N3 ;
set ORIGIN[D18] := N3 ;
set ORIGIN[D19] := N4 ;
set ORIGIN[D20] := N4 ;
set ORIGIN[D21] := N4 ;
set ORIGIN[D22] := N4 ;
set ORIGIN[D23] := N5 ;
set ORIGIN[D24] := N5 ;
set ORIGIN[D25] := N5 ;
set ORIGIN[D26] := N6 ;
set ORIGIN[D27] := N6 ;
set ORIGIN[D28] := N7 ;
set DESTINATION[D1] := N2 ;
set DESTINATION[D2] := N3 ;
set DESTINATION[D3] := N4 ;
set DESTINATION[D4] := N5 ;
set DESTINATION[D5] := N6 ;
set DESTINATION[D6] := N7 ;
set DESTINATION[D7] := N8 ;
set DESTINATION[D8] := N3 ;
set DESTINATION[D9] := N4 ;
set DESTINATION[D10] := N5 ;
set DESTINATION[D11] := N6 ;
set DESTINATION[D12] := N7 ;
set DESTINATION[D13] := N8 ;
set DESTINATION[D14] := N4 ;
set DESTINATION[D15] := N5 ;
set DESTINATION[D16] := N6 ;
set DESTINATION[D17] := N7 ;
set DESTINATION[D18] := N8 ;
set DESTINATION[D19] := N5 ;
set DESTINATION[D20] := N6 ;
set DESTINATION[D21] := N7 ;
set DESTINATION[D22] := N8 ;
set DESTINATION[D23] := N6 ;
set DESTINATION[D24] := N7 ;
set DESTINATION[D25] := N8 ;
set DESTINATION[D26] := N7 ;
set DESTINATION[D27] := N8 ;
set DESTINATION[D28] := N8 ;
```

4.2 DSP-Top8 Node 9 Span Network -20x Implementation FactorData File

```
# AMPL ILP Model for Demand-wise Shared Protection With Topology Design.

# Created on 10/14/08 for use with DSP-TR-Top.mod AMPL model.
# Generated by Dat-prep-DSP-TR-Top.exe, written by Brody Todd, June 2008.
# Contact btodd@trlabs.ca for more information.
# Copyright TRLabs 2008, All Rights Reserved.

# Command Line Used:
# DSP-TR-Top-dat-prep.exe
#   HCon-8n16s1-19s.top ../../../../Networks/8n16s1-Family/8n16s1.dem 8n16s1-Top-20.dat 20


set NODES := N1 N2 N3 N4 N5 N6 N7 N8;
set SPANS := S1 S2 S3 S4 S5 S6 S7 S8 S9 S10 S11 S12 S13 S14 S15 S16 S17 S18 S19;
set NODE_SPANS[N1] := S1 S2 S3 S4;
set NODE_SPANS[N2] := S1 S5 S6 S7;
set NODE_SPANS[N3] := S3 S5 S8 S9 S12 S15 S18;
set NODE_SPANS[N4] := S2 S7 S9 S10 S16;
set NODE_SPANS[N5] := S10 S11 S12 S13;
set NODE_SPANS[N6] := S11 S14 S15 S16 S19;
set NODE_SPANS[N7] := S13 S14 S17 S18;
set NODE_SPANS[N8] := S4 S6 S8 S17 S19;;
set ADJ_NODES[N1] := N2 N4 N3 N8;
set ADJ_NODES[N2] := N1 N3 N8 N4;
set ADJ_NODES[N3] := N1 N2 N8 N4 N5 N6 N7;
set ADJ_NODES[N4] := N1 N2 N3 N5 N6;
set ADJ_NODES[N5] := N4 N6 N3 N7;
set ADJ_NODES[N6] := N5 N7 N3 N4 N8;
set ADJ_NODES[N7] := N5 N6 N8 N3;
set ADJ_NODES[N8] := N1 N2 N3 N7 N6;;

param Cost :=
S1      191.042
S2      231.019
S3      269.268
S4      418.122
S5      175.824
S6      244.755
S7      300.947
S8      196.461
S9      213.600
S10     225.719
S11     243.594
S12     340.872
S13     384.003
S14     161.419
S15     307.148
S16     355.928
S17     258.002
S18     284.682
S19     373.268;

param Implement_Cost :=
S1      3820.838
S2      4620.390
S3      5385.350
S4      8362.440
S5      3516.476
S6      4895.100
S7      6018.937
S8      3929.224
S9      4272.002
S10     4514.377
S11     4871.878
S12     6817.448
S13     7680.052
S14     3228.374
S15     6142.963
S16     7118.567
S17     5160.039
S18     5693.646
S19     7465.360;

set DEMANDS := D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12 D13 D14 D15 D16 D17 D18 D19 D20 D21 D22 D23 D24 D25 D26 D27 D28;

param DemUnits :=
D1      3.000000
D2      9.000000
D3      5.000000
D4      10.000000
D5      6.000000
D6      1.000000
D7      8.000000
D8      3.000000
D9      5.000000
D10     4.000000
D11     9.000000
D12     4.000000
D13     10.000000
D14     6.000000
D15     4.000000
D16     2.000000
D17     9.000000
D18     10.000000
D19     8.000000
D20     9.000000
```

```
D21     4.000000
D22     9.000000
D23     6.000000
D24     1.000000
D25     2.000000
D26     3.000000
D27     7.000000
D28     9.000000;

set ORIGIN[D1] := N1 ;
set ORIGIN[D2] := N1 ;
set ORIGIN[D3] := N1 ;
set ORIGIN[D4] := N1 ;
set ORIGIN[D5] := N1 ;
set ORIGIN[D6] := N1 ;
set ORIGIN[D7] := N1 ;
set ORIGIN[D8] := N2 ;
set ORIGIN[D9] := N2 ;
set ORIGIN[D10] := N2 ;
set ORIGIN[D11] := N2 ;
set ORIGIN[D12] := N2 ;
set ORIGIN[D13] := N2 ;
set ORIGIN[D14] := N3 ;
set ORIGIN[D15] := N3 ;
set ORIGIN[D16] := N3 ;
set ORIGIN[D17] := N3 ;
set ORIGIN[D18] := N3 ;
set ORIGIN[D19] := N4 ;
set ORIGIN[D20] := N4 ;
set ORIGIN[D21] := N4 ;
set ORIGIN[D22] := N4 ;
set ORIGIN[D23] := N5 ;
set ORIGIN[D24] := N5 ;
set ORIGIN[D25] := N5 ;
set ORIGIN[D26] := N6 ;
set ORIGIN[D27] := N6 ;
set ORIGIN[D28] := N7 ;
set DESTINATION[D1] := N2 ;
set DESTINATION[D2] := N3 ;
set DESTINATION[D3] := N4 ;
set DESTINATION[D4] := N5 ;
set DESTINATION[D5] := N6 ;
set DESTINATION[D6] := N7 ;
set DESTINATION[D7] := N8 ;
set DESTINATION[D8] := N3 ;
set DESTINATION[D9] := N4 ;
set DESTINATION[D10] := N5 ;
set DESTINATION[D11] := N6 ;
set DESTINATION[D12] := N7 ;
set DESTINATION[D13] := N8 ;
set DESTINATION[D14] := N4 ;
set DESTINATION[D15] := N5 ;
set DESTINATION[D16] := N6 ;
set DESTINATION[D17] := N7 ;
set DESTINATION[D18] := N8 ;
set DESTINATION[D19] := N5 ;
set DESTINATION[D20] := N6 ;
set DESTINATION[D21] := N7 ;
set DESTINATION[D22] := N8 ;
set DESTINATION[D23] := N6 ;
set DESTINATION[D24] := N7 ;
set DESTINATION[D25] := N8 ;
set DESTINATION[D26] := N7 ;
set DESTINATION[D27] := N8 ;
set DESTINATION[D28] := N8 ;
```

4.3 DSP-TR-A 15 Node 18 Span Network Data File for Demand 1

```
# AMPL ILP Model for Demand-wise Shared Protection With Availability - Version 2.0.

# Created on 10/06/08 for use with DSP-TR-A.mod AMPL model.
# Generated by Dat-prep-DSP-A-pathsets.exe, written by Brody Todd, June 2008.
# Contact btodd@trlabs.ca for more information.
# Copyright TRLabs 2008, All Rights Reserved.

# Command Line Used:
# dsp-a-tr-dat-prep.exe
#   ../../../../../Networks/15n30s1-Family/15n30s1-18s.top ../../../../../Networks/15n30s1-Family/0.99999/15n30s1-1.dem 15n30s1-18s-1.dat 12 2920000

# The baseline per unit MTTF is 2920000.


set NODES := N01 N02 N03 N04 N05 N06 N07 N08 N09 N10 N11 N12 N13 N14 N15;

set SPANS := S01 S02 S03 S04 S07 S10 S11 S13 S16 S17 S18 S20 S21 S23 S25 S27 S29 S30;

set NODE_SPANS[N01] := S01 S02 S03;
set NODE_SPANS[N02] := S01 S04;
set NODE_SPANS[N03] := S04 S07;
set NODE_SPANS[N04] := S07 S10;
set NODE_SPANS[N05] := S02 S11 S13;
set NODE_SPANS[N06] := S10 S17 S21;
set NODE_SPANS[N07] := S11 S16;
set NODE_SPANS[N08] := S17 S18;
set NODE_SPANS[N09] := S16 S20 S25;
set NODE_SPANS[N10] := S20 S21 S27;
set NODE_SPANS[N11] := S18 S23;
set NODE_SPANS[N12] := S23 S25;
set NODE_SPANS[N13] := S27 S29;
set NODE_SPANS[N14] := S29 S30;
set NODE_SPANS[N15] := S03 S13 S30;;
set ADJ_NODES[N01] := N02 N05 N15;
set ADJ_NODES[N02] := N01 N03;
set ADJ_NODES[N03] := N02 N04;
set ADJ_NODES[N04] := N03 N06;
set ADJ_NODES[N05] := N01 N07 N15;
set ADJ_NODES[N06] := N04 N08 N10;
set ADJ_NODES[N07] := N05 N09;
set ADJ_NODES[N08] := N06 N11;
set ADJ_NODES[N09] := N07 N10 N12;
set ADJ_NODES[N10] := N09 N06 N13;
set ADJ_NODES[N11] := N08 N12;
set ADJ_NODES[N12] := N11 N09;
set ADJ_NODES[N13] := N10 N14;
set ADJ_NODES[N14] := N13 N15;
set ADJ_NODES[N15] := N01 N05 N14;;

param Cost :=
S01     166.355
S02     189.404
S03     143.024
S04     180.205
S07     129.495
S10     149.030
S11     246.941
S13     218.874
S16     150.615
S17     269.980
S18     302.870
S20     193.933
S21     368.860
S23     115.004
S25     113.265
S27     151.717
S29     141.908
S30     149.282;

set DEMANDS := D1;

param DemUnits :=
D1      8.000000;

set ORIGIN[D1] := N01 ;

set DESTINATION[D1] := N02 ;

param MIN_AVAILABILITY :=
D1      0.999990;

param REQ_AVAILABILITY :=
D1      0.000000;

param MTTF :=
S01 17590
S02 15449
S03 20419
S04 16222
S07 22635
S10 19597
S11 11869
S13 13394
S16 19466
```

S17 10855
S18 9668
S20 15129
S21 7934
S23 25391
S25 25840
S27 19337
S29 20709
S30 19597;

param MTTR := 12;

4.4 DSP-Top-A

```
# AMPL ILP Model for Demand-wise Shared
Protection With Availability - Version 2.0.

# Created on 10/06/08 for use with DSP-TR-A.mod
AMPL model.
# Generated by Dat-prep-DSP-A-pathsets.exe,
written by Brody Todd, June 2008.
# Contact btodd@trlabs.ca for more information.
# Copyright TRLabs 2008, All Rights Reserved.

# Command Line Used:
# dsp-a-tr-dat-prep.exe
#   ../../../../../Networks/15n30s1-Family/15n30s1-
18s.top ../../../../../Networks/15n30s1-
Family/0.99999/15n30s1-1.dem 15n30s1-18s-1.dat
12 2920000

# The baseline per unit MTTF is 2920000.



set NODES := N01 N02 N03 N04 N05 N06 N07 N08
N09 N10 N11 N12 N13 N14 N15;

set SPANS := S01 S02 S03 S04 S07 S10 S11 S13
S16 S17 S18 S20 S21 S23 S25 S27 S29 S30;

set NODE_SPANS[N01] := S01 S02 S03;
set NODE_SPANS[N02] := S01 S04;
set NODE_SPANS[N03] := S04 S07;
set NODE_SPANS[N04] := S07 S10;
set NODE_SPANS[N05] := S02 S11 S13;
set NODE_SPANS[N06] := S10 S17 S21;
set NODE_SPANS[N07] := S11 S16;
set NODE_SPANS[N08] := S17 S18;
set NODE_SPANS[N09] := S16 S20 S25;
set NODE_SPANS[N10] := S20 S21 S27;
set NODE_SPANS[N11] := S18 S23;
set NODE_SPANS[N12] := S23 S25;
set NODE_SPANS[N13] := S27 S29;
set NODE_SPANS[N14] := S29 S30;
set NODE_SPANS[N15] := S03 S13 S30;;
set ADJ_NODES[N01] := N02 N05 N15;
set ADJ_NODES[N02] := N01 N03;
set ADJ_NODES[N03] := N02 N04;
set ADJ_NODES[N04] := N03 N06;
set ADJ_NODES[N05] := N01 N07 N15;
set ADJ_NODES[N06] := N04 N08 N10;
set ADJ_NODES[N07] := N05 N09;
set ADJ_NODES[N08] := N06 N11;
set ADJ_NODES[N09] := N07 N10 N12;
set ADJ_NODES[N10] := N09 N06 N13;
set ADJ_NODES[N11] := N08 N12;
set ADJ_NODES[N12] := N11 N09;
set ADJ_NODES[N13] := N10 N14;
set ADJ_NODES[N14] := N13 N15;
set ADJ_NODES[N15] := N01 N05 N14;;

param Cost :=
S1      143.024
S2      166.355
S3      189.404
S4      228.432
S5      288.839
S6      291.096
S7      323.798
S8      381.494
S9      179.354
S10     180.205
S11     260.409
S12     270.150
S13     291.908
S14     379.521
S15     380.664
S16     129.495
S17     142.021
S18     206.729
S19     209.747
S20     224.243
S21     294.206
S22     149.030
S23     252.573
S24     271.516
S25     350.343
S26     379.486
S27     390.836
S28     97.082
S29     218.874
S30     246.941
S31     247.873
S32     249.546
S33     167.263
S34     269.980
S35     317.402
S36     322.349
S37     368.860
S38     150.306
S39     150.615
S40     250.154
S41     254.189
S42     172.456
S43     280.007
S44     302.870
S45     351.524
S46     360.014
S47     378.190
S48     113.265
S49     180.236
S50     193.933
```

```
S51     216.009
S52     151.717
S53     161.409
S54     190.937
S55     230.903
S56     115.004
S57     175.000
S58     274.869
S59     299.822
S60     330.288
S61     365.001
S62     108.908
S63     250.008
S64     250.785
S65     141.908
S66     214.888
S67     149.282
S68     305.655
S69     354.731
S70     301.281
S71     356.073;

param Implement_Cost :=
S1      14302.447
S2      16635.504
S3      18940.433
S4      22843.161
S5      28883.906
S6      29109.620
S7      32379.778
S8      38149.443
S9      17935.440
S10     18020.544
S11     26040.929
S12     27014.996
S13     29190.752
S14     37952.075
S15     38066.389
S16     12949.517
S17     14202.113
S18     20672.929
S19     20974.747
S20     22424.317
S21     29420.571
S22     14903.020
S23     25257.276
S24     27151.611
S25     35034.269
S26     37948.650
S27     39083.628
S28     9708.244
S29     21887.439
S30     24694.129
S31     24787.295
S32     24954.559
S33     16726.327
S34     26997.963
S35     31740.195
S36     32234.919
S37     36886.041
S38     15030.635
S39     15061.540
S40     25015.395
S41     25418.891
S42     17245.579
S43     28000.714
S44     30286.961
S45     35152.383
S46     36001.389
S47     37819.043
S48     11326.518
S49     18023.596
S50     19393.298
S51     21600.926
S52     15171.684
S53     16140.942
S54     19093.716
S55     23090.258
S56     11500.435
S57     17500.000
S58     27486.906
S59     29982.161
S60     33028.775
S61     36500.137
S62     10890.822
S63     25000.800
S64     25078.477
S65     14190.842
S66     21488.834
S67     14928.161
S68     30565.503
S69     35473.088
S70     30128.060
S71     35607.303;
set DEMANDS := D1;

param DemUnits :=
D1      8.000000;

set ORIGIN[D1] := N01 ;

set DESTINATION[D1] := N02 ;

param MIN_AVAILABILITY :=
D1      0.999990;

param REQ_AVAILABILITY :=
D1      0.000000;

param MTTF :=
S1 20419
S2 17590
```

S3 15449
S4 12807
S5 10138
S6 10034
S7 9040
S8 7664
S9 16312
S10 16222
S11 11230
S12 10814
S13 10034
S14 7704
S15 7684
S16 22635
S17 20563
S18 14174
S19 13971
S20 13035
S21 9931
S22 19597
S23 11587
S24 10774
S25 8342
S26 7704
S27 7487
S28 30103
S29 13394
S30 11869
S31 11821
S32 11726
S33 17485
S34 10855
S35 9211
S36 9068
S37 7934
S38 19466
S39 19466
S40 11680
S41 11496
S42 16976
S43 10428
S44 9668
S45 8319
S46 8111
S47 7724
S48 25840
S49 16222
S50 15129
S51 13518
S52 19337
S53 18136
S54 15368
S55 12695
S56 25391
S57 16685
S58 10656
S59 9765
S60 8848
S61 8000
S62 27037
S63 11680
S64 11680
S65 20709
S66 13644
S67 19597
S68 9573
S69 8248
S70 9700
S71 8202;
param MTTR := 12;

4.5 DSP-TR-R2 8 node 9 span data file for an R2 value of 0.5

```
# AMPL ILP Model for Demand-wise Shared
Protection With R2 - Version 1.0.

# Created on 11/17/08 for use with DSP-R2-TR.mod
AMPL model.
# Generated by Dat-prep-DSP-R2-TR.exe, written by
Brody Todd, June 2008.
# Contact btodd@trlabs.ca for more information.
# Copyright TRLabs 2008, All Rights Reserved.

# Command Line Used:
# DSP-TR-R2-dat-prep.exe
#   ../../../../../Networks/8n16s1-Family/8n16s1-9s.top
../../../../../Networks/8n16s1-Family/8n16s1.dem
8n16s1-9s.dat 0.5

# The baseline R2 is 0.5.


set NODES := N1 N2 N3 N4 N5 N6 N7 N8;
set SPANS := S1 S2 S3 S4 S5 S6 S7 S8 S11;
set NODE_SPANS[N1] := S1 S7;
set NODE_SPANS[N2] := S6 S7 S8;
set NODE_SPANS[N3] := S8 S11;
set NODE_SPANS[N4] := S1 S2;
set NODE_SPANS[N5] := S2 S3;
set NODE_SPANS[N6] := S3 S4;
set NODE_SPANS[N7] := S4 S5 S11;
set NODE_SPANS[N8] := S5 S6;;
set ADJ_NODES[N1] := N4 N2;
set ADJ_NODES[N2] := N8 N1 N3;
set ADJ_NODES[N3] := N2 N7;
set ADJ_NODES[N4] := N1 N5;
set ADJ_NODES[N5] := N4 N6;
set ADJ_NODES[N6] := N5 N7;
set ADJ_NODES[N7] := N6 N8 N3;
set ADJ_NODES[N8] := N7 N2;;

param Cost :=
S1      231.019
S2      225.719
S3      243.594
S4      161.419
S5      258.002
S6      244.755
S7      191.042
S8      175.824
S11     284.682;

set DEMANDS := D1 D2 D3 D4 D5 D6 D7 D8 D9
D10 D11 D12 D13 D14 D15 D16 D17 D18 D19 D20
D21 D22 D23 D24 D25 D26 D27 D28;

param DemUnits :=
D1      3.000000
D2      9.000000
D3      5.000000
D4      10.000000
D5      6.000000
D6      1.000000
D7      8.000000
D8      3.000000
D9      5.000000
D10     4.000000
D11     9.000000
D12     4.000000
D13     10.000000
D14     6.000000
D15     4.000000
D16     2.000000
D17     9.000000
D18     10.000000
D19     8.000000
D20     9.000000
D21     4.000000
D22     9.000000
D23     6.000000
D24     1.000000
D25     2.000000
D26     3.000000
D27     7.000000
D28     9.000000;

set ORIGIN[D1] := N1 ;

set ORIGIN[D2] := N1 ;
set ORIGIN[D3] := N1 ;
set ORIGIN[D4] := N1 ;
set ORIGIN[D5] := N1 ;
set ORIGIN[D6] := N1 ;
set ORIGIN[D7] := N1 ;
set ORIGIN[D8] := N2 ;
set ORIGIN[D9] := N2 ;
set ORIGIN[D10] := N2 ;
set ORIGIN[D11] := N2 ;
set ORIGIN[D12] := N2 ;
set ORIGIN[D13] := N2 ;
set ORIGIN[D14] := N3 ;
set ORIGIN[D15] := N3 ;
set ORIGIN[D16] := N3 ;
```

```
set ORIGIN[D17] := N3 ;
set ORIGIN[D18] := N3 ;
set ORIGIN[D19] := N4 ;
set ORIGIN[D20] := N4 ;
set ORIGIN[D21] := N4 ;
set ORIGIN[D22] := N4 ;
set ORIGIN[D23] := N5 ;
set ORIGIN[D24] := N5 ;
set ORIGIN[D25] := N5 ;
set ORIGIN[D26] := N6 ;
set ORIGIN[D27] := N6 ;
set ORIGIN[D28] := N7 ;
set DESTINATION[D1] := N2 ;
set DESTINATION[D2] := N3 ;
set DESTINATION[D3] := N4 ;
set DESTINATION[D4] := N5 ;
set DESTINATION[D5] := N6 ;
set DESTINATION[D6] := N7 ;
set DESTINATION[D7] := N8 ;
set DESTINATION[D8] := N3 ;
set DESTINATION[D9] := N4 ;
set DESTINATION[D10] := N5 ;
set DESTINATION[D11] := N6 ;
set DESTINATION[D12] := N7 ;
set DESTINATION[D13] := N8 ;
set DESTINATION[D14] := N4 ;
set DESTINATION[D15] := N5 ;
set DESTINATION[D16] := N6 ;
set DESTINATION[D17] := N7 ;
set DESTINATION[D18] := N8 ;
set DESTINATION[D19] := N5 ;
set DESTINATION[D20] := N6 ;
set DESTINATION[D21] := N7 ;
set DESTINATION[D22] := N8 ;
set DESTINATION[D23] := N6 ;
set DESTINATION[D24] := N7 ;
set DESTINATION[D25] := N8 ;
set DESTINATION[D26] := N7 ;
set DESTINATION[D27] := N8 ;
set DESTINATION[D28] := N8 ;

param Req_R2_Restorability :=
D1      0
D2      0
D3      0
D4      0
D5      0
D6      0
D7      0
D8      0
D9      0
D10     0
D11     0
D12     0.50
D13     0
D14     0
D15     0
D16     0
D17     0
D18     0
D19     0
D20     0
D21     0
D22     0
D23     0
D24     0
D25     0
D26     0
D27     0
D28     0;
```

4.6 DSP-Top-R2 8 node 9 span data file for an R2 value of 0.5 and an implementation factor of 20x

```
# AMPL ILP Model for Demand-wise Shared Protection With R2 - Version 1.0.

# Created on 10/22/08 for use with DSP-R2-TR.mod AMPL model.
# Generated by Dat-prep-DSP-R2-TR.exe, written by Brody Todd, June 2008.
# Contact btodd@trlabs.ca for more information.
# Copyright TRLabs 2008, All Rights Reserved.

# Command Line Used:
# DSP-R2-Top-dat-prep.exe
#   HCon-8n16s1-19s.top ../../../../../Networks/8n16s1-Family/8n16s1.dem 8n16s1-Top-20.dat 0.5

# The baseline R2 is 0.5.


set NODES := N1 N2 N3 N4 N5 N6 N7 N8;
set SPANS := S1 S2 S3 S4 S5 S6 S7 S8 S9 S10 S11 S12 S13 S14 S15 S16 S17 S18 S19;
set NODE_SPANS[N1] := S1 S2 S3 S4;
set NODE_SPANS[N2] := S1 S5 S6 S7;
set NODE_SPANS[N3] := S3 S5 S8 S9 S12 S15 S18;
set NODE_SPANS[N4] := S2 S7 S9 S10 S16;
set NODE_SPANS[N5] := S10 S11 S12 S13;
set NODE_SPANS[N6] := S11 S14 S15 S16 S19;
set NODE_SPANS[N7] := S13 S14 S17 S18;
set NODE_SPANS[N8] := S4 S6 S8 S17 S19;;
set ADJ_NODES[N1] := N2 N4 N3 N8;
set ADJ_NODES[N2] := N1 N3 N8 N4;
set ADJ_NODES[N3] := N1 N2 N8 N4 N5 N6 N7;
set ADJ_NODES[N4] := N1 N2 N3 N5 N6;
set ADJ_NODES[N5] := N4 N6 N3 N7;
set ADJ_NODES[N6] := N5 N7 N3 N4 N8;
set ADJ_NODES[N7] := N5 N6 N8 N3;
set ADJ_NODES[N8] := N1 N2 N3 N7 N6;;

param Cost :=
S1      191.042
S2      231.019
S3      269.268
S4      418.122
S5      175.824
S6      244.755
S7      300.947
S8      196.461
S9      213.600
S10     225.719
S11     243.594
S12     340.872
S13     384.003
S14     161.419
S15     307.148
S16     355.928
S17     258.002
S18     284.682
S19     373.268;

param Implement_Cost :=
S1      3820.838
S2      4620.390
S3      5385.350
S4      8362.440
S5      3516.476
S6      4895.100
S7      6018.937
S8      3929.224
S9      4272.002
S10     4514.377
S11     4871.878
S12     6817.448
S13     7680.052
S14     3228.374
S15     6142.963
S16     7118.567
S17     5160.039
S18     5693.646
S19     7465.360;

set DEMANDS := D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12 D13 D14 D15 D16 D17 D18 D19 D20 D21 D22 D23 D24 D25 D26 D27 D28;

param DemUnits :=
D1      3.000000
D2      9.000000
D3      5.000000
D4      10.000000
D5      6.000000
D6      1.000000
D7      8.000000
D8      3.000000
D9      5.000000
D10     4.000000
D11     9.000000
```

```
D12     4.000000
D13     10.000000
D14     6.000000
D15     4.000000
D16     2.000000
D17     9.000000
D18     10.000000
D19     8.000000
D20     9.000000
D21     4.000000
D22     9.000000
D23     6.000000
D24     1.000000
D25     2.000000
D26     3.000000
D27     7.000000
D28     9.000000;

set ORIGIN[D1] := N1 ;
set ORIGIN[D2] := N1 ;
set ORIGIN[D3] := N1 ;
set ORIGIN[D4] := N1 ;
set ORIGIN[D5] := N1 ;
set ORIGIN[D6] := N1 ;
set ORIGIN[D7] := N1 ;
set ORIGIN[D8] := N2 ;
set ORIGIN[D9] := N2 ;
set ORIGIN[D10] := N2 ;
set ORIGIN[D11] := N2 ;
set ORIGIN[D12] := N2 ;
set ORIGIN[D13] := N2 ;
set ORIGIN[D14] := N3 ;
set ORIGIN[D15] := N3 ;
set ORIGIN[D16] := N3 ;
set ORIGIN[D17] := N3 ;
set ORIGIN[D18] := N3 ;
set ORIGIN[D19] := N4 ;
set ORIGIN[D20] := N4 ;
set ORIGIN[D21] := N4 ;
set ORIGIN[D22] := N4 ;
set ORIGIN[D23] := N5 ;
set ORIGIN[D24] := N5 ;
set ORIGIN[D25] := N5 ;
set ORIGIN[D26] := N6 ;
set ORIGIN[D27] := N6 ;
set ORIGIN[D28] := N7 ;
set DESTINATION[D1] := N2 ;
set DESTINATION[D2] := N3 ;
set DESTINATION[D3] := N4 ;
set DESTINATION[D4] := N5 ;
set DESTINATION[D5] := N6 ;
set DESTINATION[D6] := N7 ;
set DESTINATION[D7] := N8 ;
set DESTINATION[D8] := N3 ;
set DESTINATION[D9] := N4 ;
set DESTINATION[D10] := N5 ;
set DESTINATION[D11] := N6 ;
set DESTINATION[D12] := N7 ;
set DESTINATION[D13] := N8 ;
set DESTINATION[D14] := N4 ;
set DESTINATION[D15] := N5 ;
set DESTINATION[D16] := N6 ;
set DESTINATION[D17] := N7 ;
set DESTINATION[D18] := N8 ;
set DESTINATION[D19] := N5 ;
set DESTINATION[D20] := N6 ;
set DESTINATION[D21] := N7 ;
set DESTINATION[D22] := N8 ;
set DESTINATION[D23] := N6 ;
set DESTINATION[D24] := N7 ;
set DESTINATION[D25] := N8 ;
set DESTINATION[D26] := N7 ;
set DESTINATION[D27] := N8 ;
set DESTINATION[D28] := N8 ;

param Req_R2_Restorability :=
D1      0.50
D2      0.50
D3      0.50
D4      0.50
D5      0.50
D6      0.50
D7      0.50
D8      0.50
D9      0.50
D10     0.50
D11     0.50
D12     0.50
D13     0.50
D14     0.50
D15     0.50
D16     0.50
D17     0.50
D18     0.50
D19     0.50
D20     0.50
D21     0.50
D22     0.50
D23     0.50
D24     0.50
D25     0.50
D26     0.50
D27     0.50
D28     0.50
```