ABSTRACT

Modern and emerging telecommunication networks employ optical technology in the backbone. Providing high-speed connectivity, high bandwidth and reliable networks, connection failures within backbones may lead to social and economic damages which may be catastrophic. Therefore, survivability is a critical aspect for the success and the delivery of data in achieving end to end connectivity requirements in optical networks. The primary focus of this research is to investigate the effectiveness of two traditional survivable link disjoint routing algorithms, Bhandari's algorithm and iterative Djikstra's algorithm in order to compare them for performance. The performance metrics include connection blocking, connection failure and path length. This research also proposes a novel heuristic link avoidance routing heuristics which is presented as an alternative to the others. Stochastic models resembling the realistic large scale traffic characteristics is used to quantitatively and qualitatively evaluate the survivability approaches under various load conditions. Independent single link failures are injected in order to examine the relative performance benefits in terms of survivability on imperfect networks.

In this research, Department of Energy's Energy Science Network (ESnet) is considered. ESnet contributes scientific researchers from more than 40 institutions and facilitates all DOE application networking needs. ESnet can support up to 100 Gbps nationwide and move datasets equal to 20 billion books every month. The network is managed by a centralized circuit reservation controller On-Demand Secure Circuits and Advance Reservation System (OSCARS) 1.0 prototype. OSCARS invokes appropriate Path Computation Engine (PCE) which computes paths based on traditional and tailored routing algorithms to schedule end to end circuit resources in advance of reservation establishment. This controller

is extensively used in this research to analyze survivable algorithms for connection blocking, connection failures and path length.

ACKNOWLEDGMENTS

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

CHAPTER 1
INTRODUCTION

1.1 Introduction to optical networks

The application in social media, mobile devices, and cloud computing increases the growth of Internet traffic in recent years. In order to serve these high data traffic, the medium through which data travels should handle huge amount of data with high speed. There are two options available that offer the means to transport data- one is copper wire and other is optical fiber. The latter is far superior to the former.

Optical network is a data communication network built with optical fiber technology. It uses fiber optic cable as the medium of transmission for converting and passing data in the form of light pulses between sender and receiver nodes. Since it uses light as a medium they travel faster and supports greater bandwidth than the electrical signals traveling through copper wire. The data loss is also lower and thereby reducing the intermediate signal boosters. Because of these reasons, Optical network is largely employed to carry traffic in high speed backbone networks.

Since optical networks use light and light comes in many different colors, these different colors can be combined on the same fiber. The goal here, is to have signals not interfere with each other. To achieve this, light can be generated at various wavelengths and switched between different wavelength channels. This is called the Wavelength-Division Multiplexing (WDM). In an all-optical network, signals remain in optical domain from sender to receiver node without being converted to electrical, thus eliminating the electro-optic bottleneck [1]. The optical network consists of optical wavelength routing nodes

interconnected by optical fiber links. This allows a light path(s) to be established between any two nodes, thus representing a direct optical connection between any two routing nodes [2]. It is normally required to use the same wavelength along the light path. This limitation is called wavelength continuity constraint. This constraint can also be relaxed with wavelength converters. Although, it is very expansive to have an optical network with network nodes that have the capability of wavelength conversion.

1.2 Routing in optical network

In optical network there are two problems to establish a light path. The first is to determine a path along which the light path can be established. The second is to assign a wavelength to the selected path based on whether the node has capability to convert wavelengths. The existing light paths cannot be re-routed to accommodate the new light path until they release the wavelength. Hence, some light path requests cannot be established if they have no free wavelengths. There are two common approaches to solve this routing and wavelength assignment problem, fixed and adaptive routing.

Fixed routing can further be classified in to fixed and fixed alternate. In fixed, the routing problem is first solved between each pair of nodes before assigning a wavelength. This is the simplest but ineffective approach. This is because a fixed route is predetermined and if it?s in use, the future request for light paths may not be established if there are no free wavelengths although an alternative path exists. A slight modification to fixed path routing is fixed alternate. Instead of one fixed path between nodes, fixed alternate calculates ?n? number of paths between a pair of nodes. This increases the probability of success of setting up the light path as it has more than one path this time. The major drawback of both these approaches is that neither of them considers current state of the network. If the predetermined fixed path(s) are not available, then it will not set up the light path. But an

adaptive routing approach considers the current state of the network. It is an unconstrained routing scheme that considers all the paths between any two nodes and a shortest path can be obtained by dynamic shortest path algorithms based on link costs at the time of light path requests. We will cover more about adaptive routing in the upcoming chapters.

1.3 Advanced and immediate reservation

These are the provisioning mechanisms that are different from each other based on the application. Immediate reservation, the light paths need to be set up immediately whereas advanced reservation does not. In advanced reservation, network operator knows the exact serving time of light path set up. This thesis considers the requests for light path arrives in advance. This means that if a request for light path arrives, then the serving time is considered to be somewhere in future.

1.4 Introduction to Survivability

Since optical networks carry high volume of data, any disruption in light path can cause economic and social damages. Hence, it is extremely important to ensure that communications crossing these links and networks are properly protected. However, the physical layer of optical network is vulnerable to a variety of failures. These failures may be planned or unplanned. Almost 25 percent of failures in networks are planned due to maintenance [Uninett]. But, unplanned failures may lead to light path connection disruption because its unpredictable and beyond network operator?s knowledge. These failures may happen due to disasters, digging works, terrorist attacks etc.,. Chapter 4 explains more on how these accidental failures and its distributions in the network.

Survivability is the ability of the light path to overcome these failures. Extensive researches has already been done on the design of survivable algorithms for different

transport technologies. They can be applied to any network not just optical. We will discuss these algorithms briefly in chapter 2. Survivability in optical networks can be achieved by providing two light paths, primary and backup in advance. These two paths are disjoint, means they do not share any common fiber links or optical components along the path. The idea is to make routing diverse as possible so that failure in primary would not affect backup paths. Once the primary path is failed, the traffic should be switched to corresponding backup path, such that the service can be protected. This approach is called path protection. One of the prime motives of this research is to make a comparative analysis of three path protection survivable algorithms, two existing and one proposed based on performance metrics as specified in chapter 5.

1.5 Centrally Controlled network

In this research, we assume our network is centrally controlled. Current trend towards centrally controlled networks relies on the separation of control and data plane. In a centralized architecture, the controller hosts all the logic in the central location. This way, it is easy to decouple the logic from network nodes, thus provide a greater visibility and control over the network. Relating this centrally controlled approach to our problem, the logically centralized controller (control plane) handles the process of setting up the light path over the optical nodes (data plane). Also, the controller has the central view and hence it is easy for the controller to locate the fault in order to switch the traffic.

CHAPTER 2
BACKGROUND

2.1 The Controller OSCARS

2.1.1 History

Short for On-Demand Secure Circuits and Advance Reservation System (OSCARS) offers multi-domain, high-bandwidth virtual circuits that guarantee end-to-end network data transfer performance. Initially a research concept, OSCARS has developed into a robust production service. Currently, OSCARS virtual circuits carry fifty percent of ESnet's annual 60 petabytes of traffic. As of November 2010, ESnet traffic topped 10 petabytes a month. In 2010, ESnet operated over 30 (up from 26 in October 2009) long-term production OSCARS virtual circuits supporting scientific areas including High Energy Physics [8].

OSCARS has been currently active in production environment and deployed in the following areas [9],

1. wide-area backbone networks (ESnet)
2. global research networks
3. regional networks
4. university campus networks
5. local-area networks
6. exchange points
7. testbeds

[1]

[1] Although the controller has been intensively utilized, this section does not talk more on design architecture or goals of controller but rather functionalities of few components that are applied in this research

2.1.2 How OSCAR works?

The path reservation of OSCARS is a two-step process,

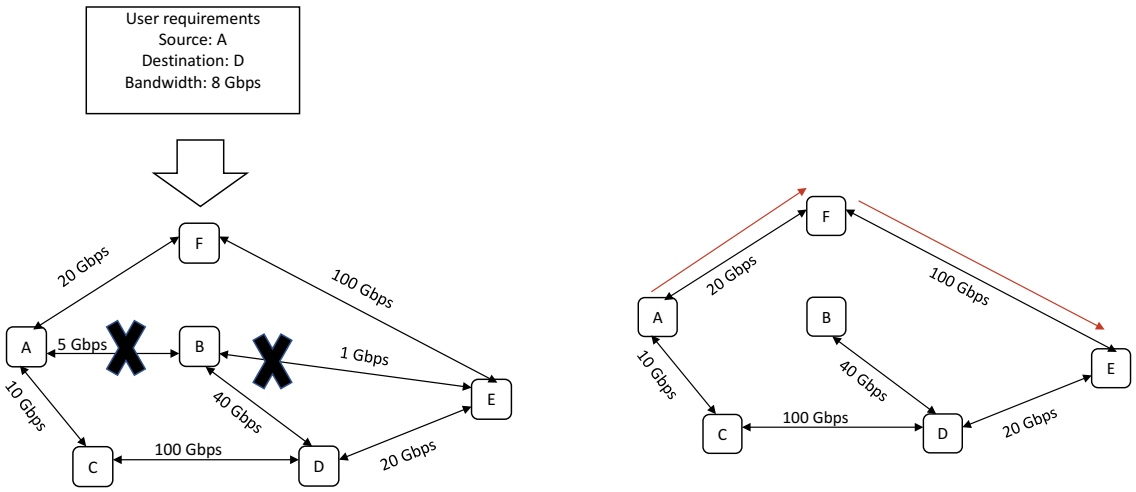
1. Scheduling the resources
2. Path finding based on availability of resources, i.e., multi-constrained path computation

2.1.2.1 Resource Scheduling

The user reservation request at minimum indicates the source, destination, bandwidth required, start and end time. Besides, a user can also submit more complex request making use of advanced API. Upon receiving the request, OSCARS authenticates the identity of the user, and determines if he/she has the appropriate authorization. The OSCARS API is designed to be accessed directly by user software or middleware applications. This greatly increases efficiency by automating the reservation process. Once OSCARS has authenticated and authorized the user requests, it determines if there is a solution path that meets all the constraints of the user. This path computation process is a core function of OSCARS [10]. A user accesses OSCARS software through a web-browser interface, or a user-application via an API as discussed in the section 2.1.3.

2.1.2.2 Path Computation

To make this path-computation process more efficient, the network link topology graph is first simplified based on resource availability. For instance, if the user makes a bandwidth request for 10 gigabits/second, the software removes any edges in the network link topology graph that have less than 10 gigabits/second of available capacity prior to determining a solution path. Then it runs different path computation algorithm in this simplified network. If the path computation successfully finds a solution path, the software updates the resource manager database with the new reservation information [10].



(a) Pruning (b) Path Computation

Figure 2.1. Illustration of Basic PCE Service Execution in OSCARS

The Path-Computation Engine/Element (PCE) in OSCARS is built on a uniquely flexible framework that allows distinct path computation elements to be composed an arbitrary workflow sequence. There are more PCE's added to the OSCARS 1.0 prototype along with their offered service enhancements and algorithms to find the solution path. These path-computation elements can distinctly take into account the basic network properties such as bandwidth (the transmission capacity of the link) of each links, and latency (the time it takes to transmit a unit of data), as well as more esoteric constraints such as energy consumption to promote green networking, which is a growing movement to reduce the energy consumption of data centers [10].

OSCARS maintains its own logical network link topology and link reservation states. All the solution paths are saved locally, synchronizing with the underlying states of the nodes. The resource manager is responsible for updating the link states after incoming requests.

2.1.3 Components of OSCARS 1.0 Prototype

As discussed in the section 2.1.2, the core functionalities of OSCARS is responsible for setting up the circuits end to end, maintaining reservations and management operations. Although this research does not talk much about all the components, it still discusses few components of the OSCARS briefly.

- **Authentication, Authorization, Accounting module:** This component authenticates the user, and authorizes the user request. This component also provides accounting information for billing or auditing [10].
- **Topology Resource Manager:** This keeps track of network link topology information such as bandwidths on each link over time and maintains information about VLAN on each network port over time.
- **Reservation Database:** In essence, it is the database that stores all the key information about the reservations made in the network so that PCE can calculate the path over the existing network.
- **Path Setup Service (PSS):** This is responsible for keep track of information about underlying network devices (routers, switched) such as configuration, supported protocols, model, manufacturer etc.,. This configures the network components for the path resulting from PCE execution as specified in the request.
- **Path Computation Element (PCE):** Runs the routing logic on the network and computes the path if any, based on the specifications provided in the request. A PCE service can be chosen and applied independently for the given request specification.
- **Reservation Workflow Control:** Decides the service capabilities and incorporates during the reservation process based on the plumbing specifications. This submodule determines the order in which the path computation submodules should be accessed.

- **Pruning Service:** Given the request specification and the existing network, this is responsible to temporarily prune the network based on the requested bandwidth. It removes all the ports that cannot support requested bandwidth or VLAN requirements. Blacklisted ports or devices may also be considered for pruning.
- **Plumbing Specification:** OSCARS is so flexible that it can even allow users to specify any combination of circuit set so long as it involves at least two valid source, destination ports and corresponding devices. Users can also request multiple flows (one to any, one to many, one to one with backup) by making use of programmatic API. Once the request being submitted to OSCARS, it is up to OSCARS to calculate the end-to-end path by invoking the PCE. Many of these PCE features are explored in this paper by submitting the right combinations of input and output ports/devices to generate traffic.
- **Web-browser interface, programmatic API:** Interacts directly with the user or user application to get information regarding the connectivity requirements. In fact, this API has been used by the traffic generator to submit multiple connection requests to analyze the network. On receiving the request, OSCARS process them and responds with the successful or unsuccessful reservation objects which will later be processed by traffic generator. Some of the important request/response specifications are discussed here.

Table 2.1. Input Request Parameters

Parameter	**Description**
Connection ID	Unique ID for a connection request
Start time	Starting time of a connection request
End time	This is the time that reserved circuits for this connection gets released
Minimum number of flows	Minimum number of flows that should be reserved
Maximum number of flows	Maximum number of flows to be reserved
Flows	Submits multiple circuit sets(source/destination Bandwidth, VLAN, Path, Survivability status)

Table 2.2. Request Flow Parameters

Parameter	**Description**
Source Device	Source device for this connection
Destination Device	Destination device for this connection
Source Port	Source port in a selected source device
Destination Port	Destination port in a selected destination device
AZ Bandwidth	A->Z Bandwidth for the end to end connection request
ZA Bandwidth	Z->A Bandwidth for the end to end connection request
Palindromic	Decides if A->Z and Z->A path are same based on options
Blacklist	Devices/ Ports to be pruned before finding path
Survivability	Determines the level of survivability (disjoint paths) based on preset values
Number of Paths	The number of disjoint paths for any connection
AZ Route	Explicitly route the packets through a route from A->Z
ZA Route	Explicitly route the packets through a route from Z->A

Table 2.3. Response Parameters

Parameter	**Description**
Connection ID	Identification of a connection request
Status	Status of connection request (Committed or Aborted)
Path	A->Z and Z->A path for the connection

2.2 Traffic Generator

The traffic generator is merely a client to the controller (OSCARS SDN) which uses the controller's API to submit the request. Since controller does most of the tasks from running a simple algorithm to installing flows in the networking devices in the topology and other business related operations, the traffic generator only has to submit the request to controller which then reserves circuits end-to-end as per the request parameters. Upon receiving the

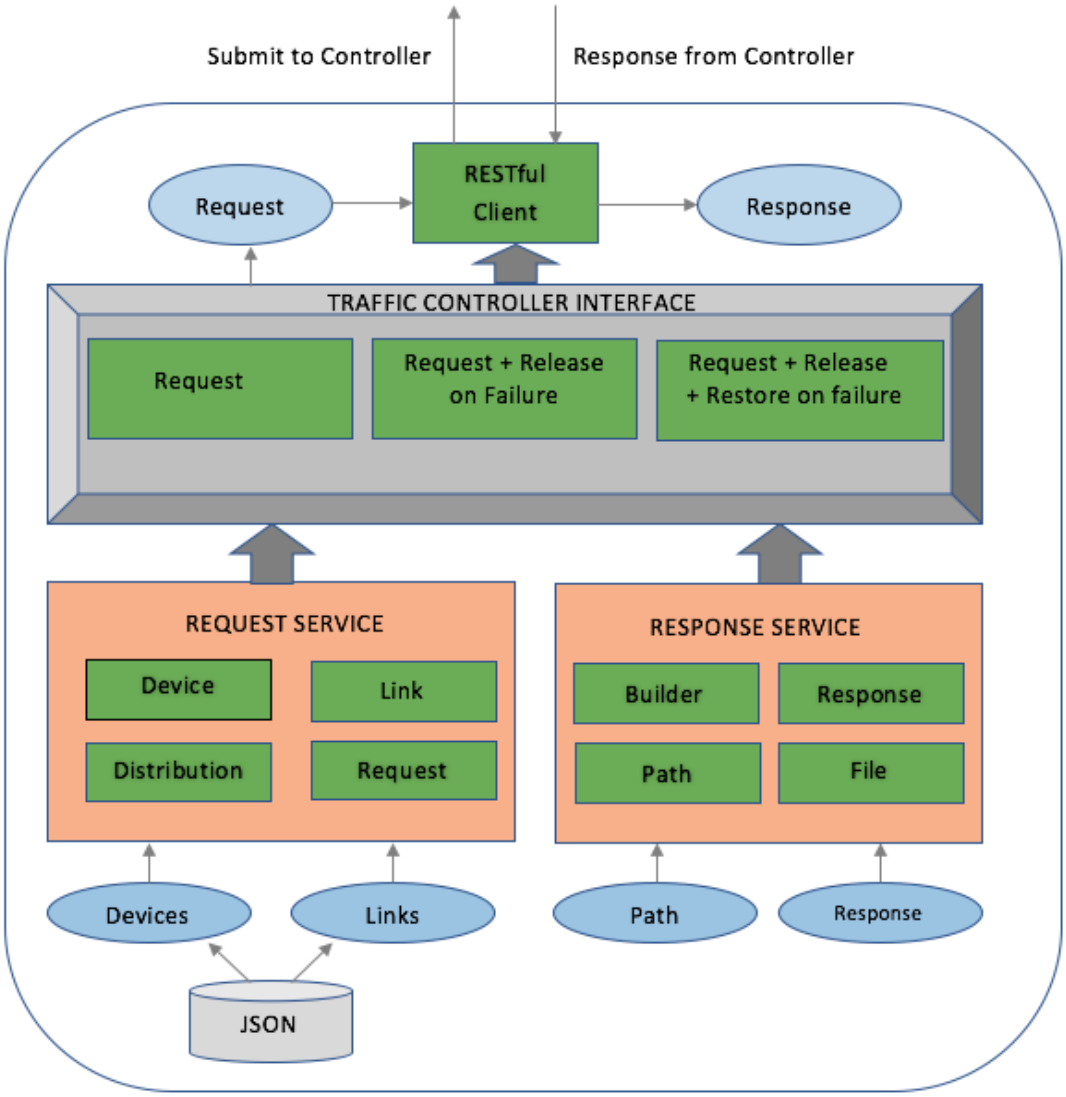


Figure 2.2. Traffic Generator Architecture

response from the controller, the traffic generator parses, processes them locally and carry out operations related to the research requirements.

Traffic Generator mainly comprises of two major components, traffic controller interface and service components that deals with all the business-related functionalities.

2.2.1 The Controller Interface

Each controller maintains its own service flows in service classes and based on which future actions are decided. The future actions include,

1. New connection request
2. Aborting previously held connection request
3. Aborting previously held but failed request
4. Converting the connection request from one type to another (Example: Multiple Unicasts to Unicast Survivability)
5. Restore previously failed connection

All these future actions are controlled based on what type of traffic one would like to generate. You can invoke the controller by its name. One can also include/write more controllers if the actions of the current controllers cannot satisfy the demands of research requirements.

[2]

The traffic generator can run controller for three different scenarios. They are,

1. Run traffic without any failures
2. Releasing the resources of failed connection if necessary. (optional)
3. Restoring the connections based on the requirements. (optional)

These specifications are decided before the start of traffic generation. Although controller OSCARS hosts most of the reservation capabilities and routing algorithms, it can still offer enough flexibility for researchers to modify these algorithms with the help of pruning and plumbing services. Hence, it is important for traffic generator to consider the possibilities to explore the controller OSCARS for new algorithms . This workaround can

[2]Read the complete document at [13] on adding more traffic controllers and how to configure/register them with the spring container

be possible until most of the end-to-end path computation algorithms runs on top of Djikstra. That said, traffic generator relies entirely on the core algorithms of controller OSCARS except that it could only force the input parameters of OSCARS to pursue and obtain the status of the network it wants to have. Current workarounds in traffic generators are,

1. All release and restoration actions for protection
2. Iterative paths
3. Parallel paths
4. Single/Multi-link disjoint path

Other PCE components in OSCARS can be found at [12]

2.2.1.1 Service Components

Service classes helps traffic controller decide the type of request it has to submit to the controller OSCARS. It mainly has two components, request and response. Each of these components maintain its own models.

2.2.1.2 Request Components

Request components have the total control over input parameters. These includes devices and links for requests (also makes sure one each of them are unique), start time, end time and bandwidths of connection based on respective distributions and finally blacklisting of links if required. They also maintain the models for devices and links.

2.2.1.3 Response Components

The response components vary based on the type of traffic controller chosen. The traffic controller interface commits to a service for a given set of input parameters to the traffic generator. They can be,

1. General response service (Unicast, Anycast, Manycast)

2. Protection service (Release and Restore)
3. Iterative path service (Release and Restore)

Similar to controllers, more services can be added. The idea behind this way of implementation is to make the tool maintainable and easily modifiable.

2.2.1.4 Other Service Components and Utilities

Besides Request and response there are other service components such as file, builder services and utility classes.

2.2.2 Rest API

Once the request is ready to submit to OSCARS, the traffic controller pushes it to rest controller. Upon receiving the request, the rest controller converts the Json to response objects.

[3]

2.2.3 Configuration files

These files contain devices, links and the length of links. Hence, if the properties of a link needs to be changed, it is enough to change the respective configuration file. This also allows reshaping of topology and easy replacement of entire network topology.

2.3 Network Survivability

2.3.1 Fault tolerance issues in infrastructure layer

Fault tolerance issues are present in each layer which might arise from the interaction between layers. For example, this research considers only possible sources of failures impact physical layer. In SDN, the failure of the links can be classified into two,

[3]Current implementation of rest controller handles few HTTP exceptions and status messages. However, there may be some unexplored exceptions which needs to be handled in a proper way to get the accurate results

1. Link failures in the infrastructure layer
2. Failure of links connecting infrastructure layer and controller

This distinction is mainly because of their varying impact in the network. This research investigates failures only in the infrastructure layer. However, failures in the links connecting nodes and controller needs separate study that may or may not affect control or application layer

Fault tolerance issues of the infrastructure layer are mostly related to issues already present in traditional networks, namely link and node failure. However, centralized management and programmability shed a different light on those issues, enabling novel strategies and bringing new challenges [14]. Let's look how it handles some of them.

2.3.1.1 Failure detection and Location

In traditional networks, fault detection and location must be performed in a distributed manner. SDN can ease this task by keeping a centralized network view that all network devices must update upon link failure. For example, a controller might identify which switches are affected by a specific failure and notify only those switches[14]. Many complex techniques are being developed to detect failures in the path/underlying nodes and to notify the failures to the controller in a reasonable time to reduce the overhead of path recovery.

2.3.1.2 Failure Recovery

Link and node failure is a problem almost as old as communications. In computer networks, there are two general failure recovery approaches, *protection andrestoration.*

(a) *Protection:* In protection switching, the protection spare capacity is pre-planned, and pre-allocated. In the case of a failure, protection switching to re-route the affected link (or connections using that link) is pre-planned, including the protection capacity. This is planned during network design and/or when the connection is established. The

downside of this approach is that protection capacity is usually not utilized during normal network operation. This reduces network resource utilization [15]. However, the same approach can be modified to improve protection capacity utilization.

In 1+1 protection, data is transmitted in working link and protection link simultaneously, and receiver selects the signal with higher quality. This approach benefits from fast recovery in case of a failure, and much simplified protection switching equipment.

1:1 protection allocates a dedicated protection link for the working link. However, data is switched to protection link only after a failure is detected. 1:1 is slower than 1+1 protection, but has the advantage that low priority traffic can use protection links during normal operation of the network. Low priority traffic using the backup links will be disrupted when a failure occurs. 1:N protection is similar to 1:1 protection, except in this case, one protection channel is reserved for N working channels. In other words, it uses shared protection capacity as opposed to dedicated (in 1:1).

(b) *Restoration:* In restoration survivability approach, the recovery (and the routing associated with it) is dynamic. In this case, network is designed with more capacity than what is needed for working traffic. The main advantage of this approach is network scalability and re-configurability. When a new traffic demand arrives, network operator's job for protecting the new connection is much easier compared to other methods. They do not need to implement specific protection rings for the new links, etc. The resource utilization could also be higher. It also likely can survive more simultaneous failures, compared to 'protection' methods. The down sides of this approach are the following: the recovery is slower than protection method, and more importantly, restoration is not guaranteed [15].

2.3.2 Algorithms for Network Survivability

The reliability of a connection can be realized through physical diversity. This can be achieved by setting up a connection with two or more disjoint paths. That said, disjoint itself can take two forms as to whether its node or just link disjoint. Fortunately, there are algorithms that can run in polynomial time to find node/link disjoint paths (K>=1). The two most common algorithms are, Bhandari's edge disjoint and Suurballe's node disjoint path algorithms. Note that Suurballe's requirement is comparatively stricter than Bhandari's as it does not allow paths to share a node. That said, Bhandari has some implementation benefits because its simpler and faster. However, if you consider yourself a 'user' of these algorithms, it is likely that the two algorithms have little variations that suits the needs but it's up to network operator to decide as to whether he wants link/node disjoint. Note that the controller OSCARS 1.0 prototype runs link disjoint algorithms and it allocates a primary and backup path(s) in advance.

2.3.2.1 Bhandari's link disjoint

The algorithms for finding disjoint path runs on top of single source shortest path algorithms Djikstra (works only with positive edge weights) and Bellmann ford (variation of Djikstra that works with negative edge weights). This section does not explore these two algorithms in detail as it is a common approach in day to day network routing problems and it is very well documented everywhere. Figure 2.3 explains how Bhandari finds two shortest edge-disjoints paths between A and F,

After obtaining the two disjoint paths, one would like to consider the shortest path as the primary path. The advantage of Bhandari is that it tries to find the total optimal cost and distribute amongst the paths. In other words, it's not greedy as Djikstra and hence, much efficient in resource allocation. Referring to the section 2.3.1.2, Bhandari being a

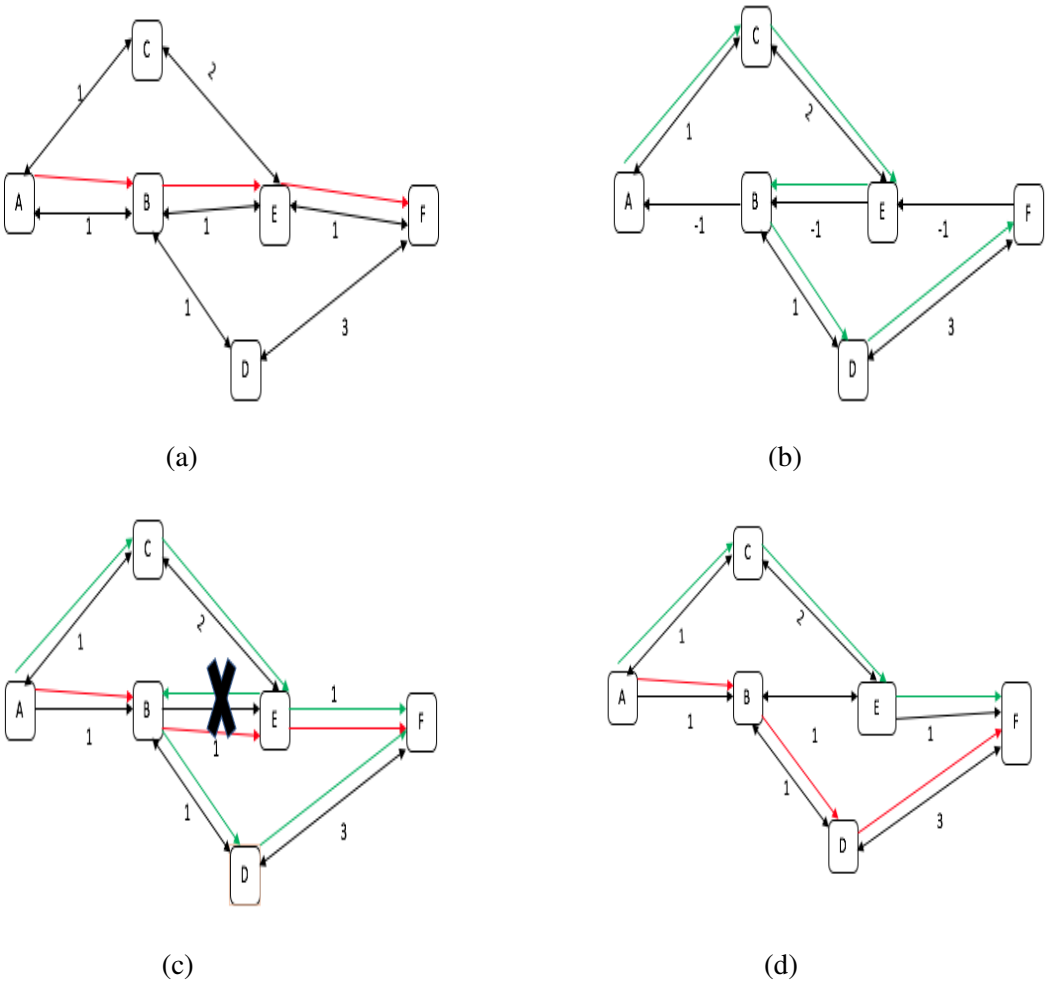
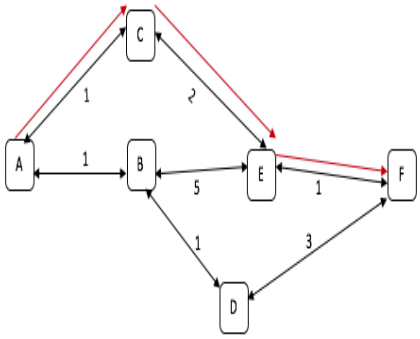


Figure 2.3. Bhandari's Disjoint Path Pair Algorithm. (a) Finds the shortest path using Djikstra (b) Reverses the edges of that shortest path with negative costs and once again runs the shortest path algorithm but this time bellman ford due to the introduction of negative edge costs (c)Removes the inverse edges obtained from running the above two (d) Puts all paths back on and hence the two link disjoint paths

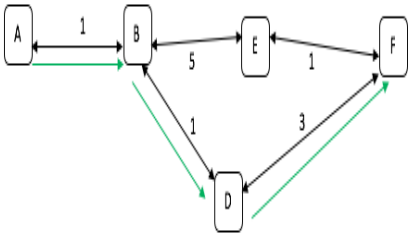
more appropriate solution for protection scenario since the connections are set in advance. Moreover, both the connections are available for the entire duration and allows the operator to have the options of parallel transfers if necessary.

2.3.2.2 Iterative Shortest Path

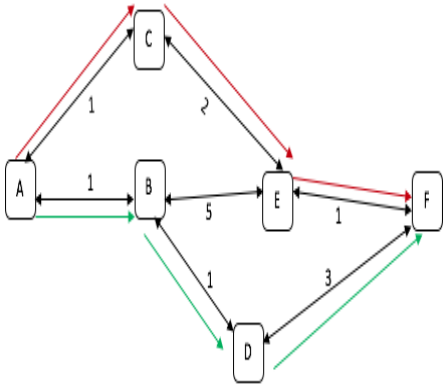
One more straightforward algorithm for link disjoint would be to iteratively run Djikstra for (K>=1). The pruning facility in OSCARS allows user or operator to prune the links based on customer requirements. These ascents the intent of operators/researchers to try out different survivability algorithms, one of them being iterative Djikstra. Let's see how it can be implemented.



(a)



(b)



(c)

Figure 2.4. Iterative Disjoint Shortest path Algorithm (a) Find the shortest path with Djikstra (b) Remove the links in first shortest path from the network and run Djikstra again (c) Add all the paths

Note that the costs are modified for the sake of this example so that iterative Djikstra can find two disjoint paths. If you run the same in our previous network used for Bhandari, you end up with only one. Djikstra being so greedy tries to allocate minimum resources as possible to primary and hence end up blocking more than Bhandari because of lack of resources to set up the backup. See the example below,

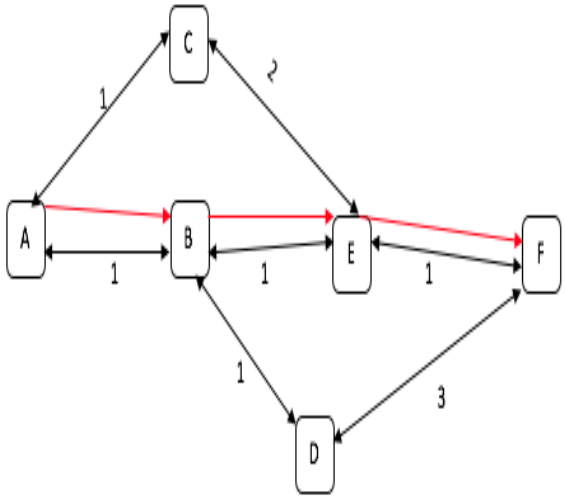


Figure 2.5. Problem with iterative SP approach

In figure 2.5, there is no space for backup path. An intelligent reader would consider the performance of this algorithm for the application 'restore on failure'. Because sometimes you need to restore unicast connections only if a failure occurs. In that case, you run Djikstra again to find the shortest path. From the above, the possibility to restore the connection is low but it still allows new connections to come in as it does not double the allocation in advance. But this research still sticks with protection approach for both the algorithms. But it is interesting to explore successful connection from protection and restoration under dynamic traffic.

CHAPTER 3
TOPOLOGY AND TRAFFIC MODELLING

3.1 Augmented ESNet Topology

ESnet is the department of Energy's dedicated science network, helping researchers meet their goals. scientific data flows are like enormous elephants. ESnet is optimized to help those elephants move quickly and efficiently around the world, so that scientists can pursue discoveries.

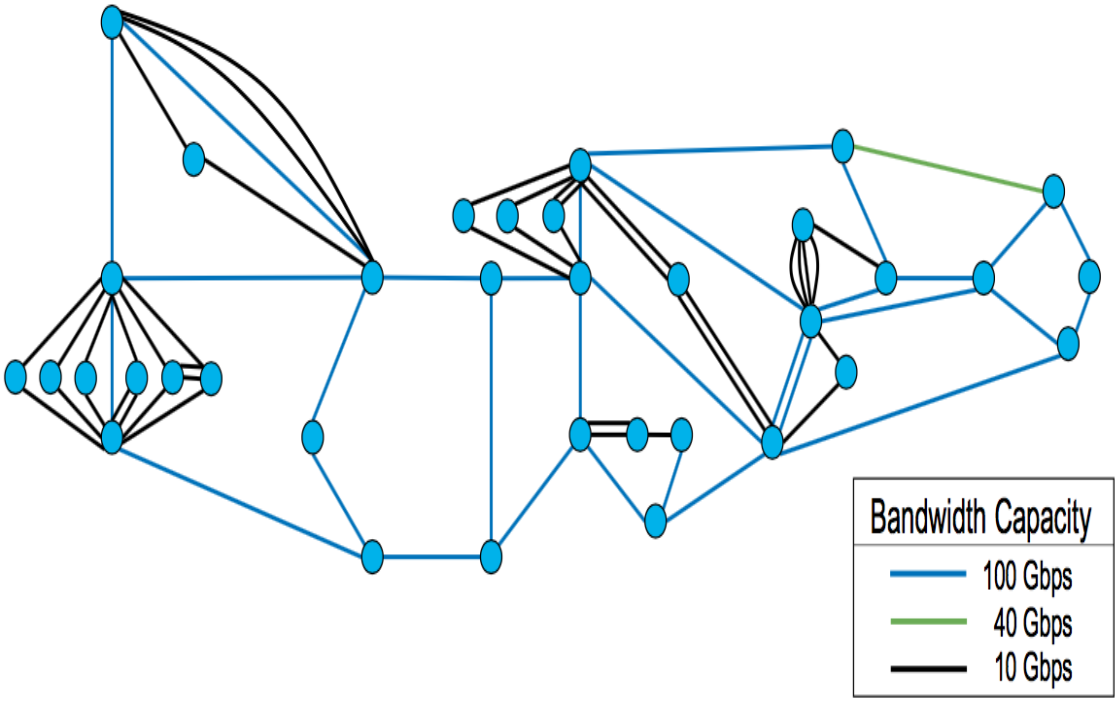


Figure 3.1. Augmented ESnet Topology

ESNet can move data at the speed of 100 gigabits per second. More than 40 percent of the links are 100 gigabits per second (precisely 34 in the above topology out of 73) and

rest 10 Gbps and one of them being 40 Gbps. The following assumptions and modifications were made about the network in this research,

1. Almost all the blocking happens inside the network due to the insufficient bandwidth. It is also assumed that there are enough vlan tags available for all the generated requests in the entire period of simulation.
2. Links that could not support giant flows were taken out of the topology because most of them were used for management purposes.
3. The Bandwidth of access ports of switches (the entry point to the network) were all set to 100 Gbps to avoid blocking at the fixtures. This makes sure that all the blocking only happens inside the domain.
4. The topology is assumed to have a bidirectional links.

[1]

3.2 Traffic Generation

Source traffic modeling and traffic generation is a very important aspect of research about network communication. An accurate estimation to the network traffic is the basis to handle/control the traffic. A good estimation to network traffic (PDF, CDF, mean, variance, etc..,) helps to solve the questions in the design of routing algorithm. Also, it provides a powerful and flexible tool to test all kinds of service designed for network communication.

Many works have been done in this area and the approaches cover every aspect of internet traffic. Although it is hard to dig deep into most popular questions about traffic modeling and generation. That said, this research still does its best to discuss some characteristics of internet traffic particularly in optical networks, some important distributions used to describe different aspects of traffic [1].

[1] OSCARS 1.0 is being deployed in Energy Science network

3.2.1 Traffic Modelling

Traffic generation and traffic modeling are two sides of a coin. A good traffic model is the goal to develop a detailed understanding of the traffic characteristics of the network. Analysis of the traffic provides information like the blocking probability for various load, the bandwidth requirements, traffic flow from a source to destination for a type of traffic (Unicast, Anycast, Manycast) and numerous other details. Also, traffic models enable network designers to make assumptions about the networks being designed based on experience and enable prediction of reservation performance for future requirements [2].

A good model demands a closest approximation to the input traffic parameters such as arrival and holding time, bandwidth, source and destination for a request and their corresponding distributions. Let's inspect these parameters one at a time.

3.2.1.1 Arrival time distribution

The Poisson process is an extremely useful process for modeling purposes in many practical applications, such as, for example, to model arrival processes for queueing models. It is empirically found that in many circumstances the arising stochastic processes can be well approximated by a Poisson process. Hence under these conditions, the arrival processes can be modeled as Poisson,

1. Each request has to be independent to each other.
2. At any time, only one request can arrive (ie. Only one can be served at a time while the other is waiting in the queue).
3. For any given time frame, the probability of arrivals of certain time interval is the same for all time intervals of equal length

From the standpoint of the network users, the requests are typically triggered by a particular events. But from the standpoint of the network, these request initiations are

somewhat arbitrary and unpredictable. Therefore, the sequence of times at which requests arrive is a random process. Moreover, these requests are triggered by different users that has no relation to each other. To make this assumption further stronger, each request has to be treated one by one based on first come first serve and this seems to acknowledge the orderliness property. Also, the requests arriving to OSCARS will be processed one by one. Since our problem in hand uphold the properties of Poisson, it has been considered in our work [2].

(a) *Poisson Arrival Process*

Now that we have established scenarios where we can assume an arrival process to be Poisson. Let's look at the probability density function of exponential arrival .

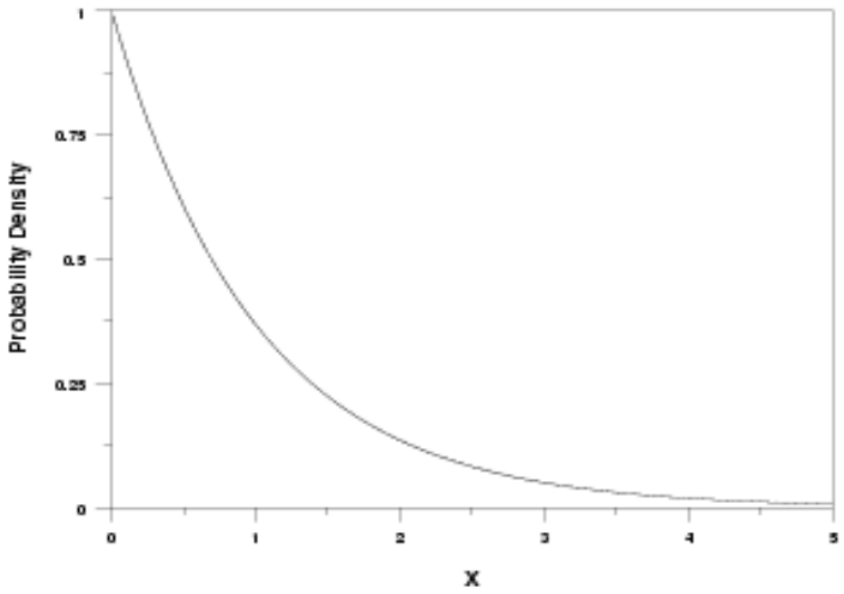


Figure 3.2. Exponential PDF

An interesting observation in Poisson models is that as the mean increases (lambda is more than 30), the properties of the Poisson distribution approach those of the normal distribution.

It is also observed that the time between any two arrivals follow exponential distribution. When events occur according to an exponential distribution, they are said to occur completely at random. Thus, the arrival time of the next event is not affected by the time elapsed since the previous event.

(b) *Single Controller criteria*

With M/M/1 queueing system, we have a single server for the queue [3]. Hence, M/M/1 can be applied to our system since it meets our criteria of single controller for requests from one network domain.

3.2.1.2 Exponential Service Times

In an M/M/1 queueing system we assume that service times for customers are also exponentially distributed (i.e. generated by a Poisson process). Unfortunately, this assumption is not as general as the arrival time distribution. But it could still be a reasonable assumption when no other data is available about service times which in our case its true.

(a) Calculation of Load

Load or Traffic Intensity, is defined as the average arrival rate (lambda) divided by the average service rate (mu).

Load = Lambda / mu; (Load $>=$ 0)

while Lambda- Average Arrival rate mu- Average Service rate

[2]

3.2.1.3 Bandwidth Distribution

From [5], it is true that the commercial network has diversified request bandwidths. The more appropriate distribution in this case follows uniform distribution considering the range

[2]Choice of load value is very critical for the analysis of traffic. From the above equation if arrival increases, the load on the network also increases (there should be more active requests in the network).

of usage of network in question. However, this generalized assumption cannot be valid for ESnet. As explained in the previous sections, ESnet is a high-speed network engineered and optimized to support large-scale transfer [8]. It is also understood that it is carrying data that is twice the rate of the commercial Internet; today it carries approximately 20 petabytes of data each month. Hence, uniform distribution of bandwidth may lead to inaccurate results as most of the links are 100 Gbps and the range of usage is not clearly known.

Any alternatives here? It is also inferred from [4] that ESnet differs from a traditional provider of network services because massive science data flows require different handling than small flows generated on the global Internet. This makes it certain that most of the requests for ESnet may demand high bandwidth as they arrive. This offers an ascend to consider normal distribution for the choice of bandwidth for each request. Meanwhile, this does not provide any assurance for realistic traffics except that it is more appropriate than uniform distribution in our case. That said, the realistic bandwidth distribution may not fit to any general distribution because it requires more intensive research based on the traffic monitoring.

The following is the plot of the standard normal probability density function

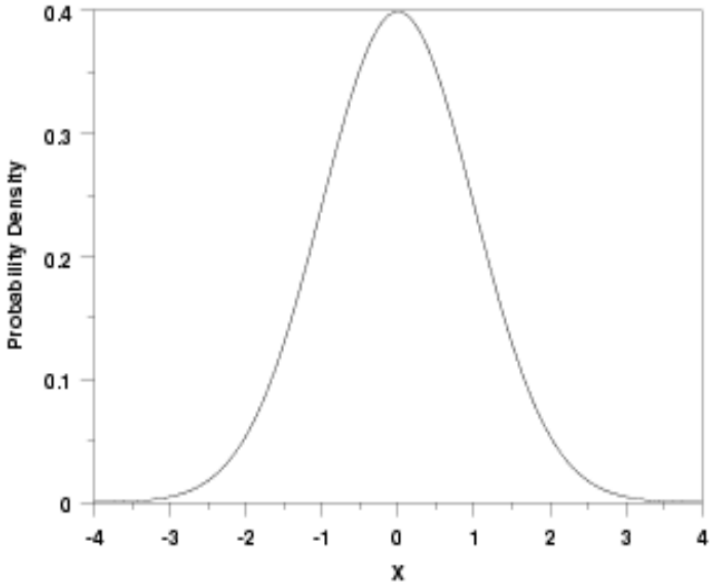


Figure 3.3. PDF of Normal Distribution

3.2.1.4 Device Distribution

Unlike bandwidth, vendor, unfortunately did not provide further statistics on how these traffic flows are distributed in the network because they form business critical information. Due to this reason, it is safe to consider uniform distribution as load may be equally distributed across all fixtures when number of requests approach higher values. The probability density function of the uniform distribution is given by,

$f(x) = 1/b - a$; for $0 < x < 1$

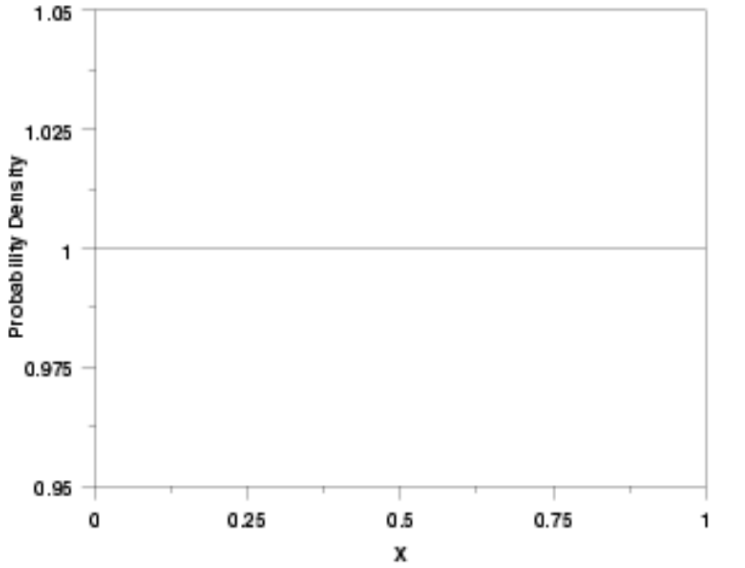


Figure 3.4. PDF of Uniform Distribution

3.2.2 Input parameters for traffic generation

To determine a realistic performance evaluation, the input load is carefully chosen to keep the blocking probability closer to the working range as much as possible. It is highly unlikely that a network of such capacity be used for high blocking for dynamic traffic generation. Moreover, the topology is frequently being updated by ESNet to support the high volume of network as the demand increases. Because of these reasons, strenuous testing for very high loads of traffic is avoided.

Repeated trials and accurate examinations have been made to fix a proper load range to achieve the above. Also, the traffic generation parameters are carefully chosen to match the realistic traffic generated to controller OSCARS handling ESNet. For example, OSCARS is responsible for approximately 5,000 virtual circuit reservations created for demos, transient experiments, and projects [7]. This detail on number of requests over the simulation period is considered for this research and it puts optimum efforts to adjust the load accordingly. It

is also crucial that simulator needs to run for more number of small, realistic traffic sets to realize the optimal results. Also, the request bandwidths vary based on the type of traffic flows and follows normal distribution as discussed in the section 3.2.1.3.

Table 3.1. Traffic Distribution Summary

Traffic Parameter	**Value**	**Distribution**
Bandwidth	1 Gbps (variant of 400 Mbps)	Normal
Device	(S,D) => $(0, 34)$	Uniform
Ports in the device	Varies based on devices	Uniform
Start Time	(-ln U/lambda)	Discrete Poisson
End Time	Start time + (-ln U/mu)	Exponential

All results shown in this chapter represent the average of 10 unique sets of reservation requests, and we have included the 95% confidence intervals, which are quite narrow.

CHAPTER 4
LINK FAILURE DISTRIBUTION AND ITS EFFECTS

4.1 Introduction

Analysis of survivability begins from introducing failures in the network and test its performance against realistic failure scenarios. On the contrary, backbone networks are usually well-engineered and adequately provisioned, leading to very low packet losses and negligible queueing delays. This robust network design is one of the reasons why the occurrence and impact of failures in these networks have received little attention. However, such failures occur almost everyday and an in-depth understanding of their properties and impact is extremely valuable to Internet Service Providers (ISPs) [16]. Unfortunately, the lack of failure data from operational networks has further limited the investigation of failures in backbone networks. No wonder in this research, the network ESNet did not reveal any such failure data about the network like most of the vendors and this forces it to consider the most common scenarios of link failures. Almost 25% of the failures in the network is due to the scheduled maintenance activities and can be ignored. This research considered only unplanned link failures due to fiber cuts, malfunctioning of optical equipment, and amplifier failures which may lead to connection interruption. Also, the availability of devices is considered as 1 (it does not fail the entire duration) and therefore minimal overlaps of link failures in the network. The following section describes the distribution, commonly applied failure and repair rates to the backbone networks.

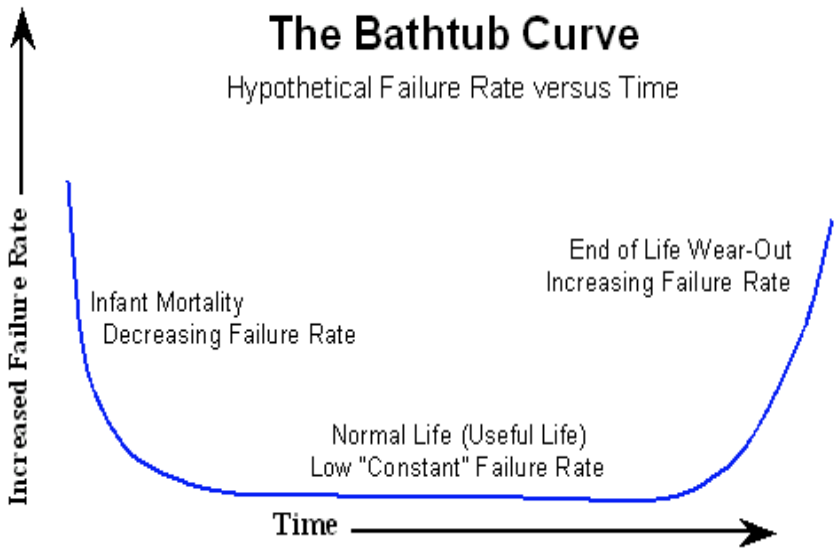


Figure 4.1. Bathtub Curve

4.2 Failure characteristic and distribution

It is common to assume Weibull distribution for modelling failures for practical applications. This is because of the adaptability of this distribution to the above bathtub curve. The curve is divided in to three regions and each of these regions states that,

1. Any equipment in its early period has decreasing or zero failure rates because new stuffs are less likely to fail often. (shape parameter$<$1)
2. Later, failure rate remains constant in normal useful life. At this period, failure is completely random in time and even the considered sample. Here the failure rates are constant. (shape parameter=1)
3. The third and the final region is 'wear-out' region. Here, the failure rates increase to infinity and the components need to be replaced. (shape parameter$>$1)

All these three regions can be easily modeled with Weibull. In fact, it decays to exponential distribution when the shape parameter is 1. This is the region where the failures

are random and can be used to model accidental failure scenarios. That said, it is not appropriate to consider other regions unless more data on components can be known. Given the links that is assumed to have external influence, it safe to consider the middle region of bathtub curve. This research does not consider deliberate failures or failure of components due to its aging. Having said all these, the vendors can model their own failure rate curve like 'Bathtub Curve' and can set threshold on replacements based on their demands and requirements of the customer.

[17] explores the application of Weibull distribution and fits it to the network-wide time between failures. [18] investigated UNINETT IP backbone on a per-link basis and fits the up time of these links to well-known distributions. It is important to consider that in [18], the links are characterized based on the distance. For example, the short, medium distance links were fitted to Weibull distribution with shape parameter less than 1 whereas the long-distance links could be modelled with Gamma distribution. To derive these failure data, the vendors should outsource the information and an accurate modelling can be possible by fitting each link to its corresponding distribution. But in order to do this, the location and number of failures of link should be observed for a reasonable period of time. The authors of [17], [18] also warned to not to blindly apply these parameters to any other network as it may lead to a deception in the failure analysis. Hence, in this research, arrival is modelled with Poisson distribution and repair times follow exponential [21].

Many researchers from [19] [20] [21] have found that the failure of link is directly proportional to the distances and they are independent to each other. In fact, [19] has given the failure rates and repair rates of the links based on its installation type which will be discussed in the following section.

4.3 Failure and Repair Rates

In reliability engineering, one should be very familiar with few terms.

(a) *MTTR or Downtime*

The average time that it takes to repair a failed component,

(b) *MTBF or Uptime*

The average time that it takes for a link to fail

(c) *Availability = (MTBF-MTTR)/MTBF*

(d) *FIT (failure-in-time)*

The term FIT is defined as a failure rate of 1 per billion hours.

To model the failures, one needs to do enough research on these values. This research tries its best to fit these values to link failures from reviewing [22], [23], [24], [25], [26] [27] and judged based on what majority of these authors concluded. Since not all ESNet links are buried (some of the them goes overseas), the FIT's for buried links were picked up from [22], [26], overseas from [22] and amplifiers from [23], [27]. The concluded final values are tabulated as follows.

Table 4.1. Summary of failure rate assumption w.r.t link type

Installation Type of Link	**FIT/km**	**Amplifier FIT/**	**MTTR (in hrs)**
Buried	114	2850	20
Overseas	21.55	2850	450

It is commonly assumed to have one amplifier for every 100 km. Since amplifiers are also the function of links, it can be converted to corresponding distance using the simple formula. Each amplifier contributes to failure of link that is 25 km long (= 2850/114). For instance, if a link is 1500 km long, then it's considered to have 15 (=1500/100) amplifiers and therefore, it is equivalent to the link distance 375 (=15*25). Now, the new link distance

becomes 1875 (1500+375) that includes the rate of amplifier failures. Hence the MTBF of the link is calculated as follows,

MTBF = 1000000000/ FIT * Total link length

Total link length (includes amplifier failures) = ((Amplifier FIT/Link FIT) * (Link distance/Number of amplifiers)) + link distance.

Using this formula, the MTBF can be calculated for all the links as shown in the topology.

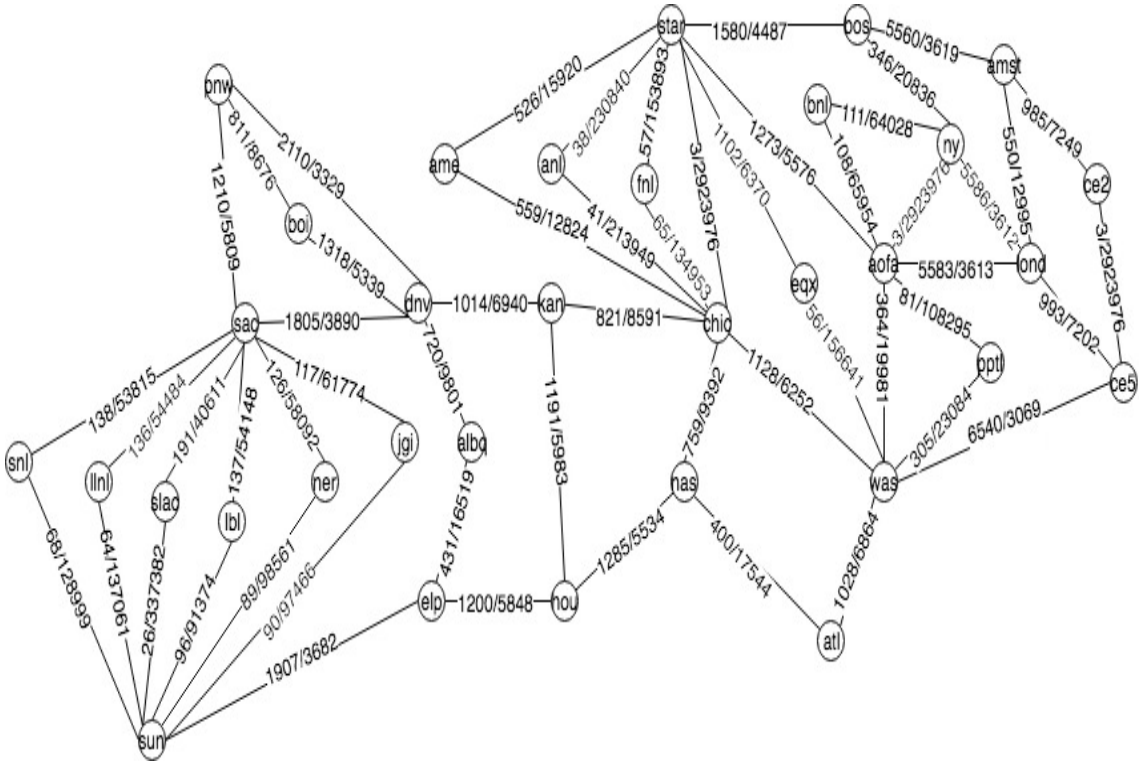


Figure 4.2. Augmented ESnet Topology with MTTF and MTTR

Table 4.2. Number of links based on its installation type

Type)	Number of links
Buried	66
Overseas	6

4.4 Effects of Failure Assumptions

All these values discussed in the previous sections, has various impact on failures analysis. Some of them are discussed in this section.

1. The simulation is set to run for a 1 year period (8760 hours) and total failures in this duration happens to be 63. This is due to the constant number of failures in every sub-interval.

Table 4.3. Periodic failure counts

Period(in hours)	**Number of failures**
0 - 10000	72
10000-20000	63
20000-30000	56
30000-40000	69

2. The time between the arrival of each failure is also constant over a period of 100000 hours. Because of this reason, the simulation time is set to 1 year as the trend is expected to replicate after this period. Figure 4.3 attempts to prove the memoryless property of Poisson.
3. Overlaps are very rare yet some accidental overlaps tend to happen due to more number of long distance links. It is normal to observe this trend due to presence of number of overseas links and it's higher MTTR.

It can be observed that majority of simulation time has no or one failures. Rarely, the overlaps happen and the intensity fades away for three and four overlaps. This is illustrated in figure 4.4. This scenario is good for survivability analysis as connection would still be able to survive in the event of single link failures in the system.

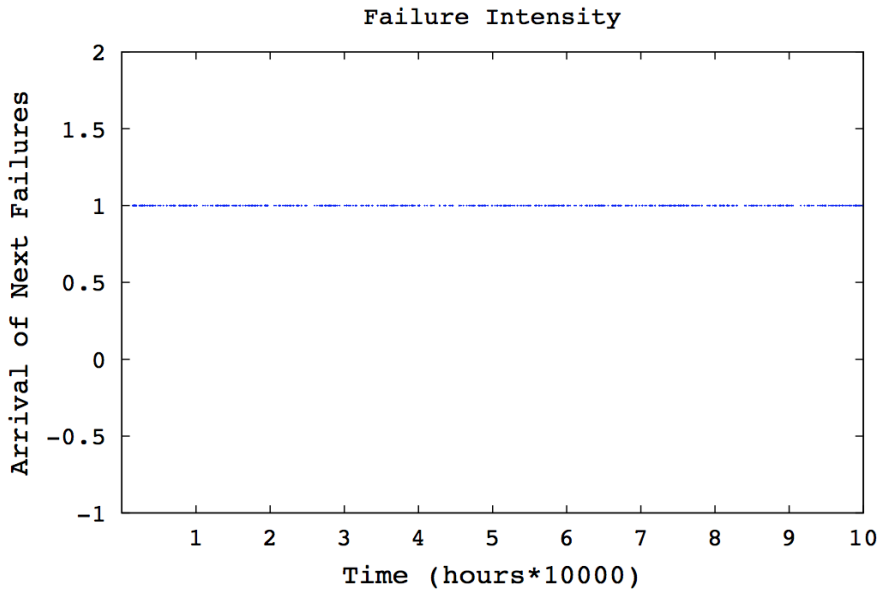


Figure 4.3. Failure arrivals over time

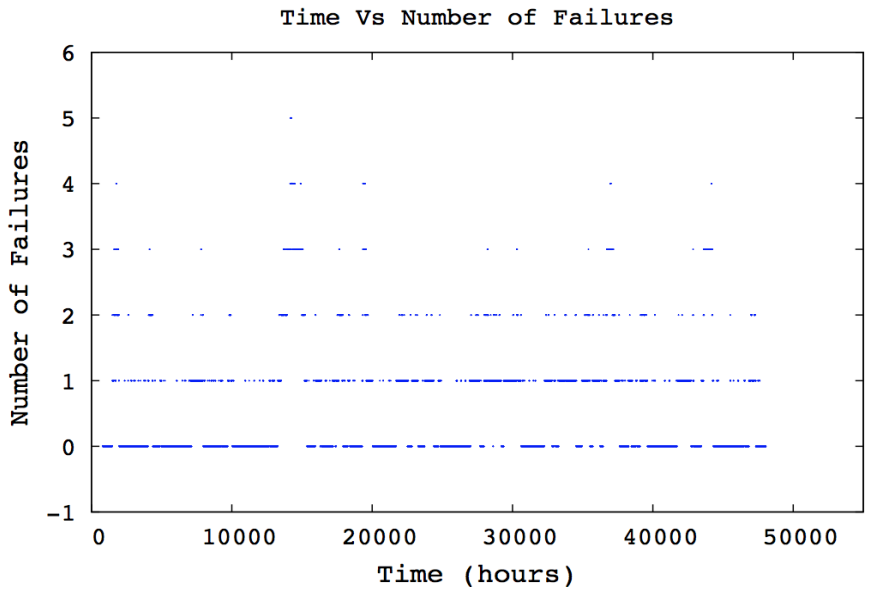


Figure 4.4. Link failure overlaps over time

4.5 Factors influencing failures in the network

when analyzing the failures, there may be different phenomenon that impact failures. These impacts may be the result of network design and could result in the increase or decrease in the failure. This section does talk much about MTTF and MTBF which may directly impact failures. But there could be many of other indirect interpretation that should be explored. Probing in to such details will set it up and makes it easy to examine the results in the upcoming chapters.

4.5.1 MTBF and MTTR

The MTBF and MTTF are parameters of failure and they have direct impact on failures. However, in this section, these parameters are decided to conduct the normal working range of the network.

Table 4.4. Effect of failure w.r.t MTTF

FIT/km	Load	Failure Probability
114	100	0.03
314	100	0.13

From the above result, the increase in the failure is directly proportional to the increase in MTTF. This is due to unavailability of links for longer period. However, the MTTR did not impact blocking.

Usually failure rates can be 114 FIT/km, 184 FIT/km and 314 FIT/km. This research considers 114 FIT/km as backbone networks are assumed to be most reliable.

4.5.2 Distance and hop counts

One underlying assumption in this section is that greater hop counts is directly proportional to the longer path. This is valid as pretty much all networks are built this way. Longer paths

can sometimes be an indication of network in high blocking state and it is about to reach steady state. Further increase in load may not be a good idea as graph is divided into small number of sub-graphs which may result in less number of hop counts and this is different from the shorter paths at lower loads. This is one of the many reasons in this research why the network was not tried for higher values of load.

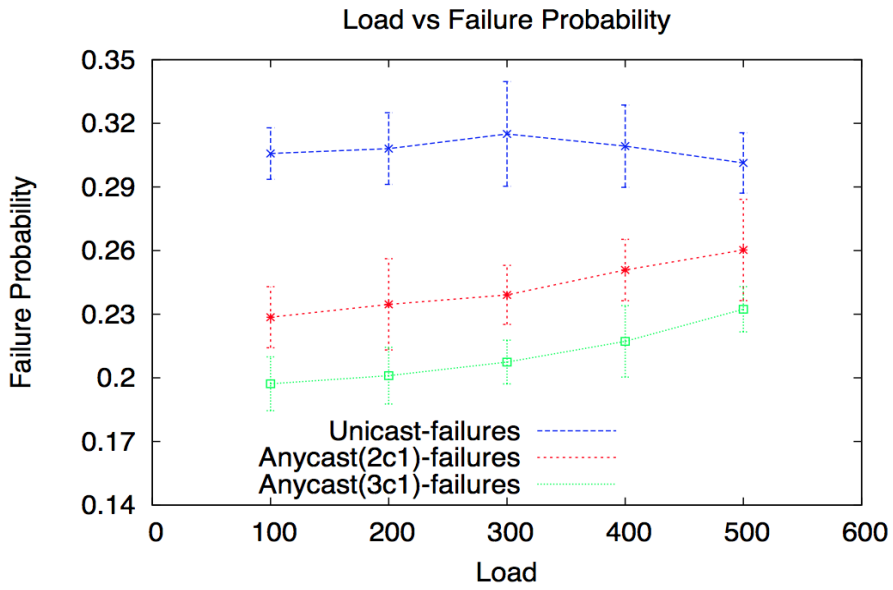


Figure 4.5. Effect of distance w.r.t failure

Connections set up with longer paths are also prone to link failures. Because longer paths cover more number of long distance links to get to the destination thus the probability that one of the links getting failed will also increase. This will also get us in to thinking that connections set up in the network which is in high blocking state would result in more number of failures.

It is evident from [1] that anycast has spatial advantage over unicast but also indirectly effects failures. The anycast in OSCARS 1.0 prototype is built in such a way that a path with minimum cost is preferred over others [1]. The idea here is to decrease the resource allocation which may assist in decrease in failures as well.

4.6 Holding time

Like hop count, increase in holding time of the connection increases the probability of connection getting failed. Because the longer the connection stays, greater the chance of link failures.

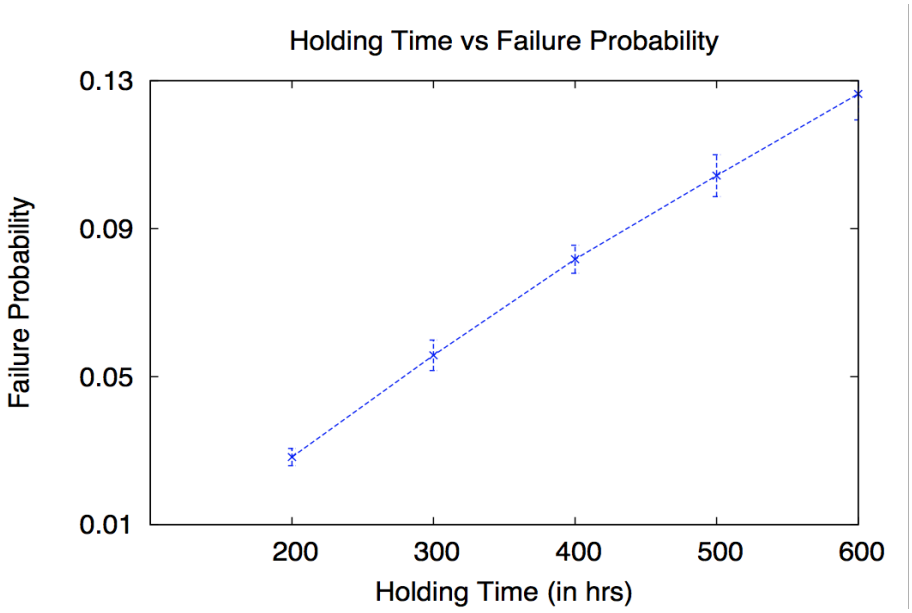


Figure 4.6. Holding time Vs Failure

All these values MTTR, holding time are tuned up in such a way that network could be operated for lower blocking and failure.

CHAPTER 5
PERFORMANCE EVALUATION OF UNICAST SURVIVABILITY

This chapter explains the survivable algorithms and their sharp differences in their blocking, failure and resource allocation by introducing failures described in the chapter 4.

5.1 Setting up the environment

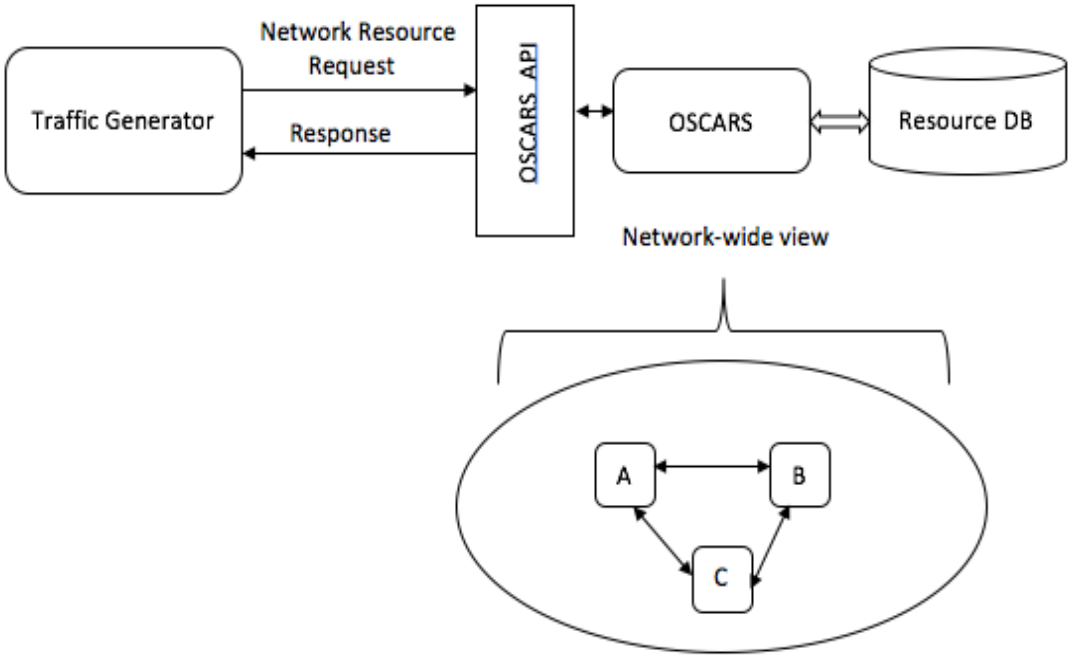


Figure 5.1. Setting up the environment

As discussed in the previous chapters, the traffic generator is merely a client connects to the controller OSCARS and generates traffic making use of API. For every request, OSCARS calculate path and responds with the format specified in the chapter 2. Note that these requests are submitted to OSCARS one at a time.

5.2 Effects of Palindromic Path

In commercial networks, the forward and reverse path for a ($S->D$) does not necessarily have to be the same. In other words, they are mostly non-palindrome (Forward and reverse path are not same). But in a private domain sometimes, palindrome may be strictly preferred over non-palindrome. For instance, OSCARS gives priority to palindrome but also keeps the option of non-palindrome open on demand. This demand may impact failure and blocking. It is presumed that non-palindrome brings more links in to equation to set up a connection and therefore, increases the probability of failure of connection. That said, it is also possible to have reduced blocking in case of non-palindrome because this does not enforce a connection to be palindrome. Figure 5.2 illustrates the working of this PCE in OSCARS.

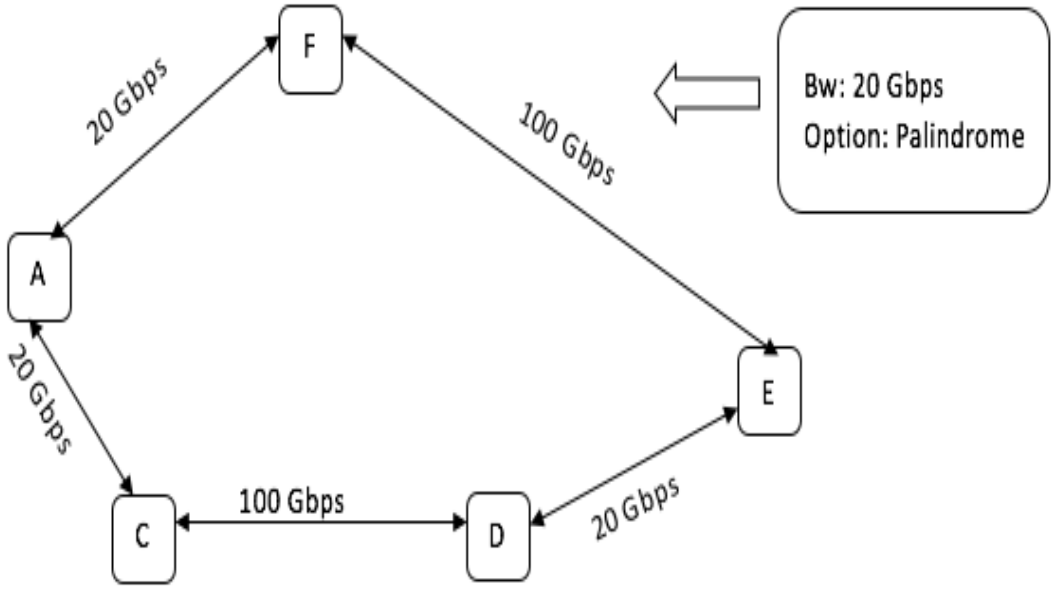


Figure 5.2. Palindromic PCE in ESnet

5.2.1 Palindrome

For the sake of this example, the bandwidths mentioned in the above topology are assumed to be available in one direction. Let's say if a user requests bandwidth 20 Gbps from A to E, controller fails to find the path with the palindrome option. Both the paths AFE and ACDE can only allocate resources for forward path because the maximum available bandwidth is 20 Gbps. In this research, this can be considered as a block as it fails to allocate resources for the connection.

5.2.2 Non-Palindrome

Non-palindrome being an option, controller would be able to identify that for the given pruned network, it is not possible to set a palindromic connection. But it does not give up there. This time it calculates if there could be any other reverse paths available. In the above topology, controller would be able to set up the connection with AFE as forward and EDCA for reverse path. This is a successful set up of a connection.

5.3 Performance Analysis

Blocking probability is the ratio of number of blocked connections to the total number of connections.

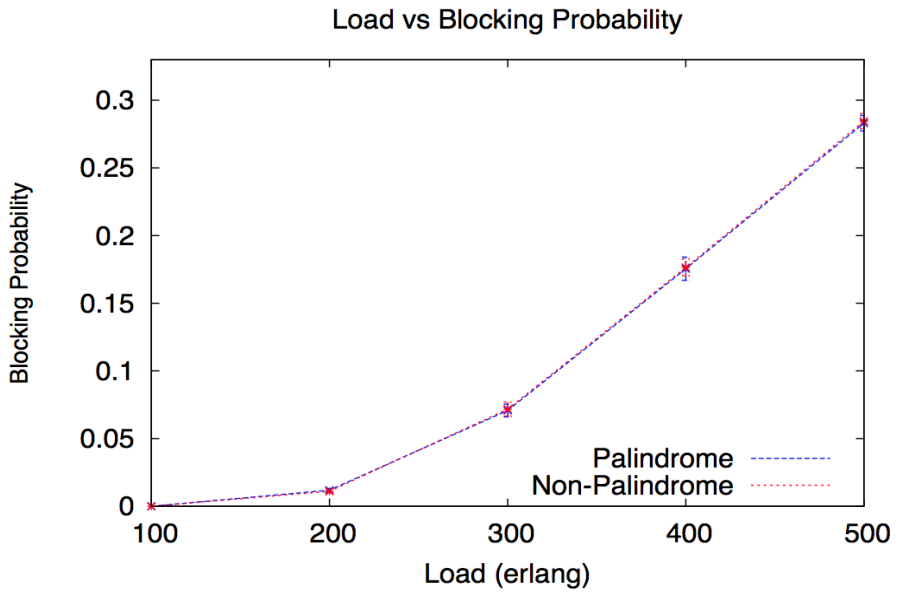


Figure 5.3. Palindrome Vs Non-Palindrome - Blocking

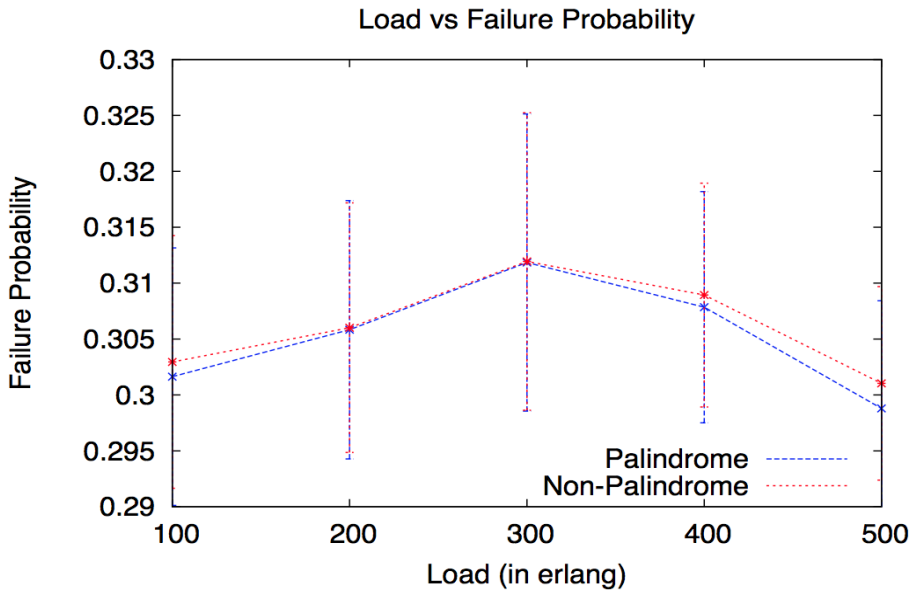


Figure 5.4. Palindrome Vs Non-Palindrome - Failure

Failure probability is the ratio of number of failed connections to total number of successful connections.

Above results seems to suggest that Non-palindrome did not have any advantage in terms of blocking probability if the forward and reverse bandwidths for connection are same (not shown here). This trend seems to continue for asymmetric bandwidth requests as well. The trend lines of palindrome and non-palindrome overlap during the entire period. In fact, non-palindrome performs worse for some of the load values. Because the return paths of some of the connections tend to take a longer route, thus contributing to the blocking of new logical connections. Failure plots further strengthen palindrome because non-palindrome seems to fail little more often. This is due to the reason briefly explained in the chapter 4.5.

The data above helps in directing the future analysis of survivability to fix on palindrome. Therefore, the simulation is run for specifications, Palindrome and Symmetric Bandwidth.

5.4 Comparative analysis of unicast survivability

This section discusses how the survivability algorithms, Iterative SP and Bhandari perform and tries to explore the effective resource allocation of these two survivable algorithms.

5.4.1 Blocking Probability

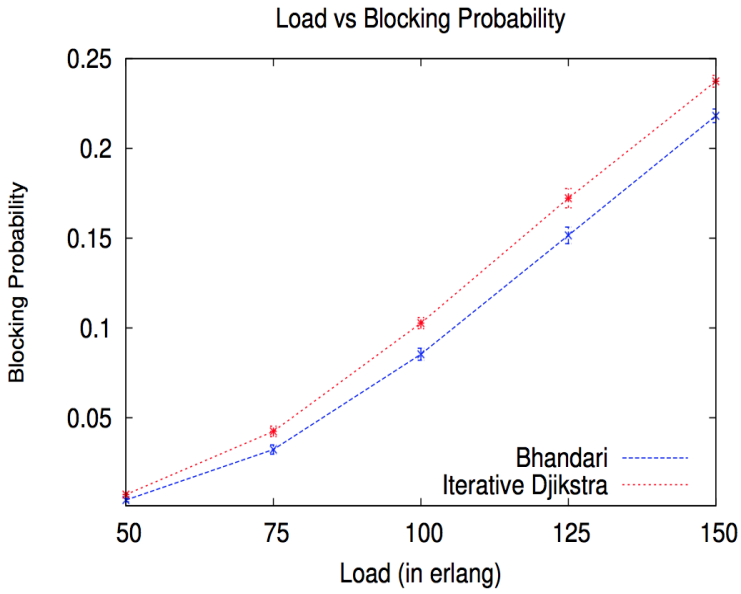


Figure 5.5. Iterative disjoint SP Vs Bhandari - Blocking

This behavior is already discussed in the chapter 2.3.2. Iterative Djikstra is greedy in allocating resources and therefore, it may lead to more blocking than Bhandari. This chart confirms that in this network, iterative SP has effected roughly up to 2% increase in blocking probability at higher loads.

5.4.2 Resource allocation

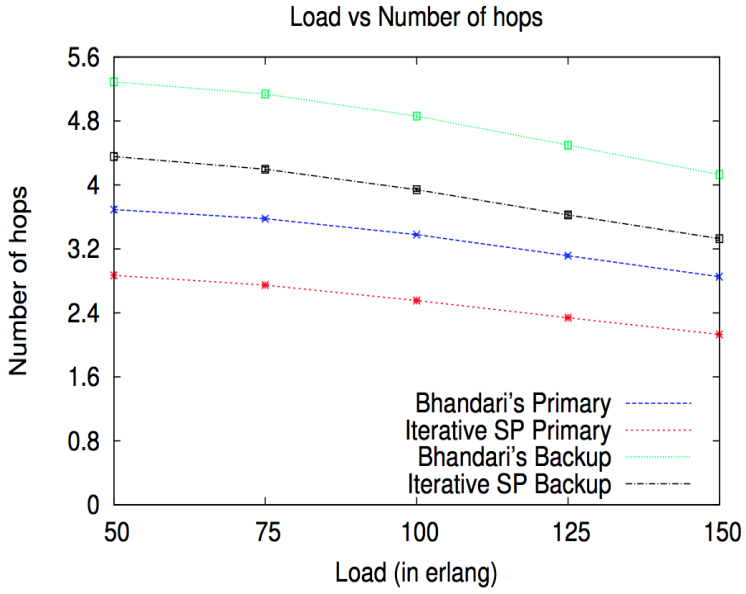


Figure 5.6. Iterative SP Vs Bhandari - Hop counts

The Number of hops is the average number of hops for all successful connections from a source to destination in one direction.

Figure 5.6 shows that average hop counts decreases by 1 for both primary and backup paths of iterative Djikstra. This shows that the logical connections set up with Bhandari may be slower and using more resources than iterative Djikstra even at lower loads with almost zero blocking. But decreased hop counts may positively approach failures. Let's see if this assumption is true in the following section.

5.4.3 Failure Probability

Surprisingly, the reduced hops of iterative Djikstra does not provide any advantage over Bhandari for failure. Shorter paths are expected to experience less number of failures. Iterative Djikstra, although takes less number of hops to reach the destination, it still should

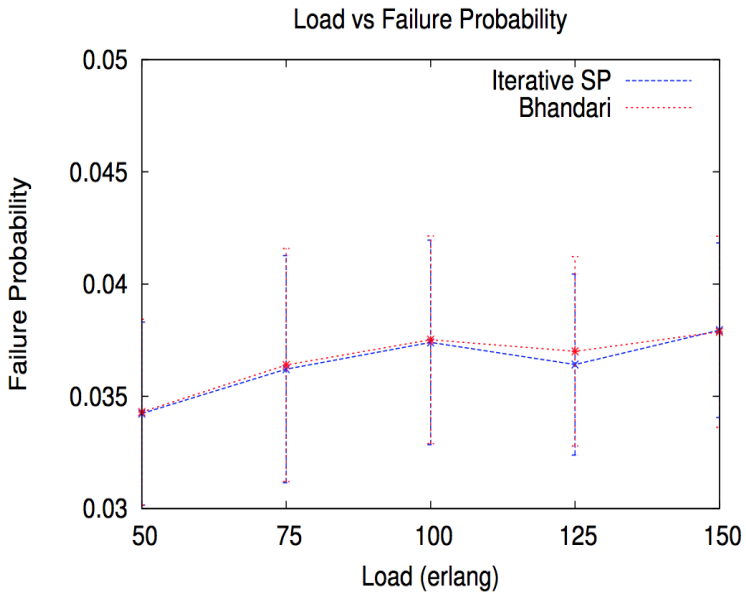


Figure 5.7. Iterative disjoint SP Vs Bhandari - Failure

travel the same distance as Bhandari. Due to this, paths of iterative Djikstra may use more number of long distance links as a result of reduced hop counts. Remember, the decrease in the anycast failure is because the destinations are different and controller OSCARS may choose the destination with shortest path based on cost. But in survivability, there is only one destination and its distance from source for primary paths in both the approaches are almost same.

Fig. 5.8 explains the case where the iterative Djikstra is forced to use a link of distance 2018 km whereas Bhandari could reach with medium or short distance links. Remember, the failure of the each link is independent and directly proportional to distance. Hence, in the below example, both the paths have equal chance of failure.

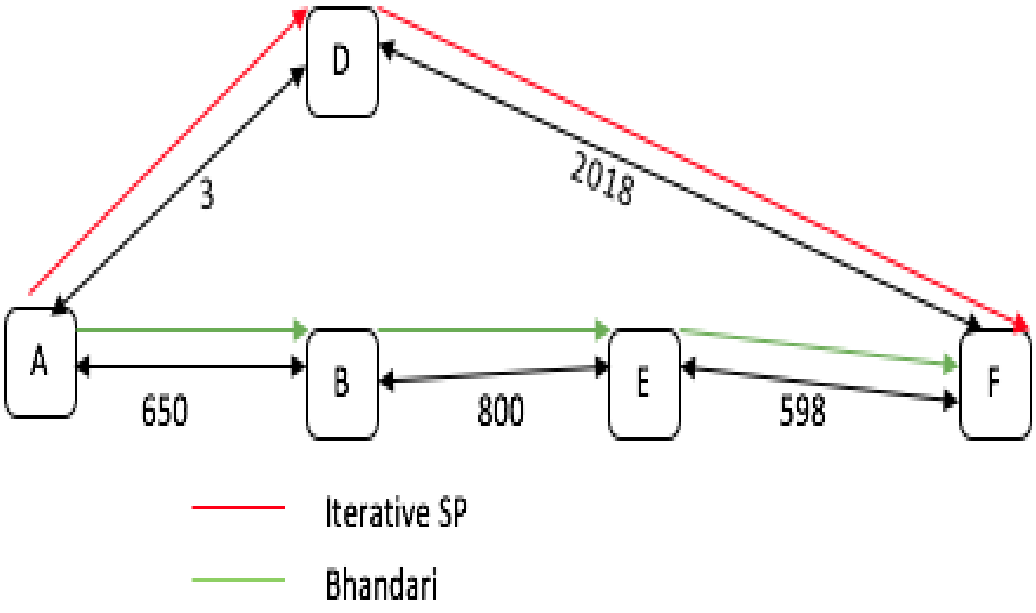


Figure 5.8. Illustration of path distance for primary and backup paths

To explore more on this, the failure probability is calculated for individual paths to see if that is true. It appears that backups have failed more than primary but between primary or backup, there is not much difference. This proves that backup paths are actually taking the longer route in terms of costs, hops and distance for both iterative approach and Bhandari.

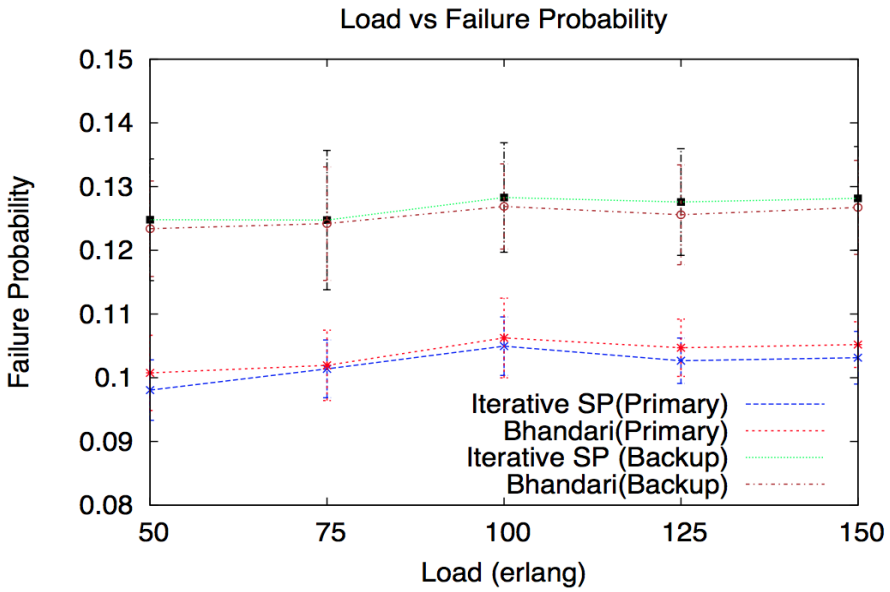


Figure 5.9. Individual path failures of Iterative and Bhandari

Although resources impact network, blocking is given preference as connections need to be committed. Besides, failures also did not seem to favor iterative Djikstra. Further, resource allocation of Bhandari is still optimal at higher load. Because of these reasons, Bhandari seems to be a convenient option for protection. But, this is not the end of the story for iterative Djikstra because the blocking probability of both the survivable approach at very low load does not seem to differ much. Can we exploit this behavior? Read on to know more about this in the next chapter.

CHAPTER 6
RISK-AWARE PARTIALLY DISJOINT ITERATIVE DJIKSTRA

This chapter proposes a slight modification to Iterative SP and tries to argue that how such an approach can be periodically beneficial to the network. That said, the conclusions derived from chapter 5 cannot be ruled out at higher loads.

6.1 Proposed Modification to Iterative SP

The conventional Iterative SP without modification is discussed in the chapter 2. To briefly recall, it runs Djikstra iteratively by pruning the links from the previous path. But this is termed to be greedy and proved to be blocking more than traditional disjoint algorithm, Bhandari. The proposed modification tries to minimize this blocking to resist Bhandari even at moderate or higher loads guaranteeing minimum resources. To achieve this, it tries to streamline the routing of both primary and backup by only pruning the links in the primary path that are vulnerable to failures. Let's discuss this in more detail.

6.1.1 Link Pruning

The Pruning of links is an important decision to make as it plays a major role in not just blocking but also failure. While it's not completely disjoint, it may contribute to failures. Also, this approach is trying to reduce the blocking by introducing the same load in the network and at the same time, to take failures in to consideration. Hence, it is important to make a crucial decision on pruning. However, this decision depends on the network topology under test. Remember, the ESNet connects sites, institutions across the country which results in large number of long distance links. Not just this, there are links that goes to different continent as well. Referring to the topology considered in this research, it is

Table 6.1. Link counts and failures based on distance

Threshold distance (km)	**Number of links**	**Number of failures in 1 year**
$distance > 1000$	19	44
$500 < distance < 1000$	9	19
$distance < 500$	44	9

evident long that distance links have smaller MTBF and thus failing more often than short distance links that has greater MTBF.

Hence, it is important to decide on a threshold distance which helps in filtering out the failure-prone links in primary path. Figure 6.1 illustrates the division of links based on distance and their corresponding failures.

Almost one-third of the links have distance greater than 1000. It is evident from their MTBF that these links fail often for the simulation period chosen. One approach is to take average of all the links for threshold. But the link distances are diversified enough to ignore this assumption. Another approach is to manually look at the number of failures that each set of links undergo, inside a period of 8760.

Also, almost half the failures are caused by the links of distance greater than 1000 km. The medium distance links however, contributes to the failure, their link count is small enough to ignore. Moreover, these medium distance links may play a crucial role in inter-state routing. So, pruning these links may increase blocking and thus, reduce this to iterative Djikstra. One more option would be to cut down half of the medium distance links. However, exploring these cases is out of the scope of this research yet very interesting. As it happens, this research sticks to the threshold value of 1000 km. Figure 6.1 explains Risk-aware routing.

6.1.2 Risk aware PDID Routing

The steps are as follows,

6.1.2.1 Iterative Shortest Path

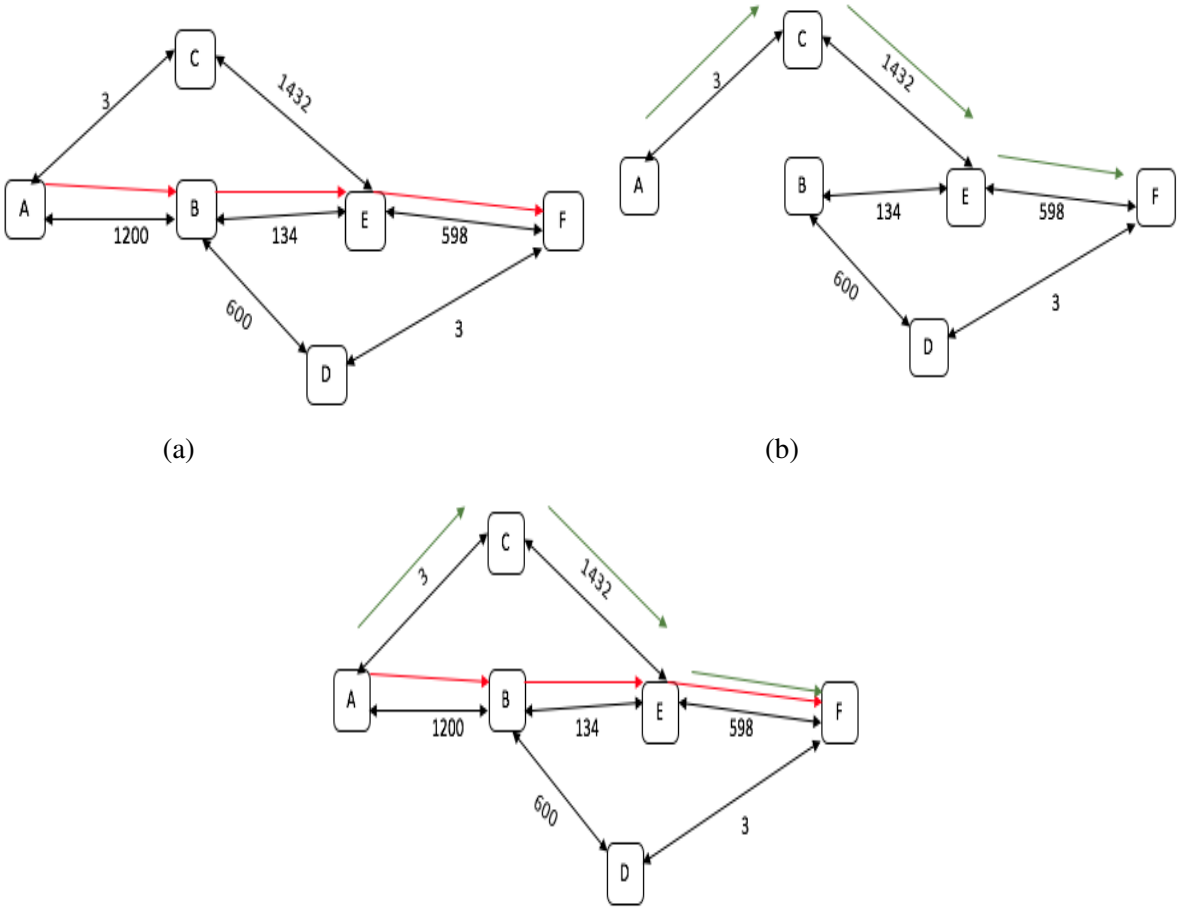


Figure 6.1. Risk-aware Iterative partially disjoint shortest path (a) Find the shortest path in the network (b) Remove the links as specified in the section 6.1.1 and run the Djikstra again (c)Add both the paths back on

6.1.3 Effects of Risk PDID link pruning

The expected increase in the failure with this approach is not just due to the partially disjoint paths. Risk aware PDID approach can only ignore the vulnerable path in the primary link and there is no guarantee that such a link would not exist in the backup path. In the figure 6.1, the backup path still has one of the links greater than 1000 km. But this scenario is common in the other algorithms as well. That said, Risk aware PDID still cuts down the probability of failure of a connection due to long distance links in to half just like the other two.

6.2 Performance analysis of Risk aware PDID

6.2.1 Blocking Probability

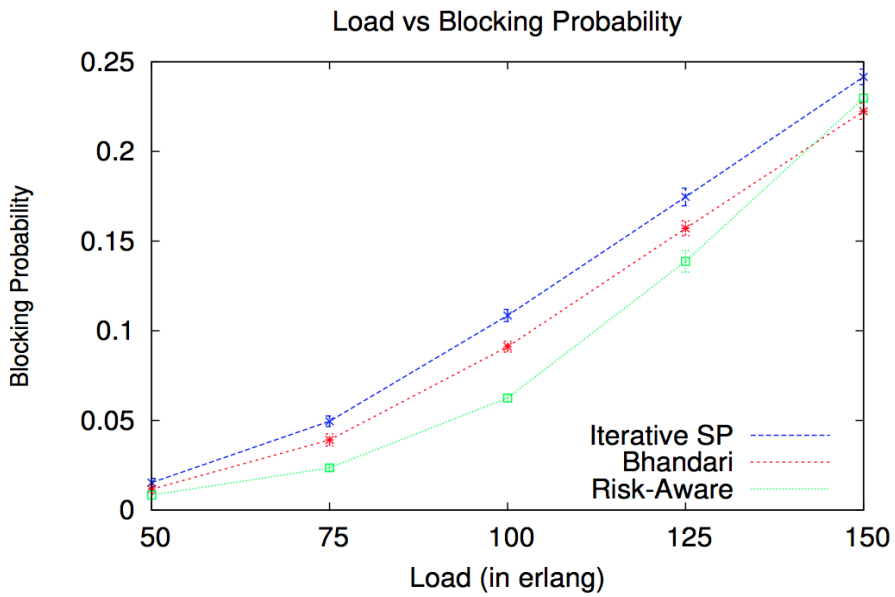


Figure 6.2. Comparative analysis of Iterative shortest path, Bhandari shortest path and Risk aware PDID for blocking

Unsurprisingly, the blocking probability is decreased for Risk aware PDID compared to Iterative Djikstra. But the plot seems to suggest some interesting facts between Risk

aware PDID and Bhandari. At lower loads, Risk aware PDID shows better performance than Bhandari. Although this decrease is very small in the beginning, it increases with the increase in the load. At moderate load, Risk aware PDID shows at most 3% decrease in blocking. This indicates that there are enough links still available for Risk aware PDID to allocate resources greedily for the connection and continues to outperform Bhandari. However, this does not last longer as further increase in load contributes to closing this gap between Bhandari and Risk aware PDID. At higher load, Bhandari still recovers as Risk aware PDID pays price for being greedy just like iterative Djikstra.

6.2.2 Failure

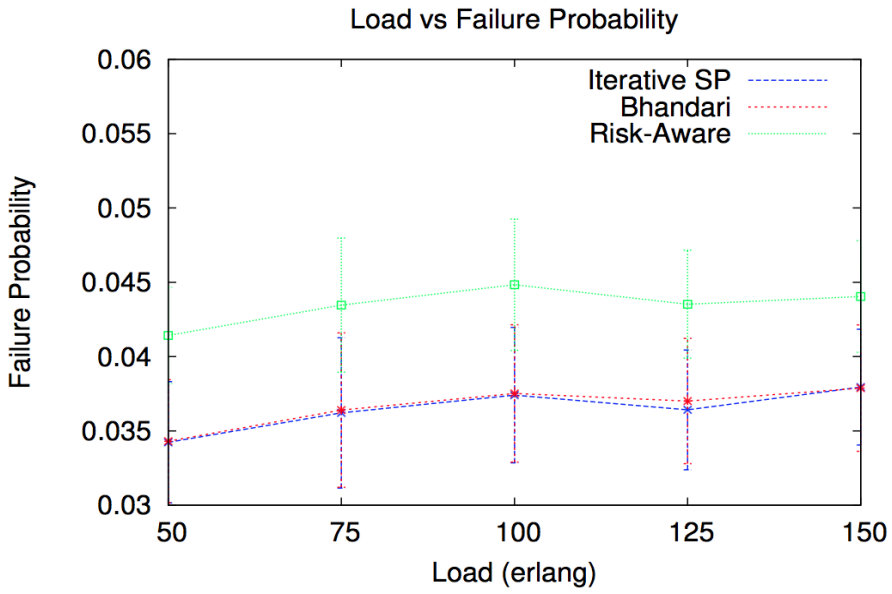


Figure 6.3. Comparative analysis of Iterative shortest path, Bhandari shortest path Risk aware IPDSP for failure

As expected, the failure for Risk aware PDID dominates both Iterative Djikstra and Bhandari. The primary reason being disjoint, the probability that a failure of medium or small distance links would lead to connection failures as well. Moreover, removing the long

distance links in the primary path may lead to longer backup path and hence makes it prone to failures. That said, the increase in failure is just 0.08 % while it can offer 3 % decrease in blocking.

6.3 Releasing the resources of failed path(s)

For all the simulation results obtained above, the failed paths remain in the network blocking new connections. Revoking the bandwidth of failed path(s) make a way for new connections. You may see the drop in the blocking.

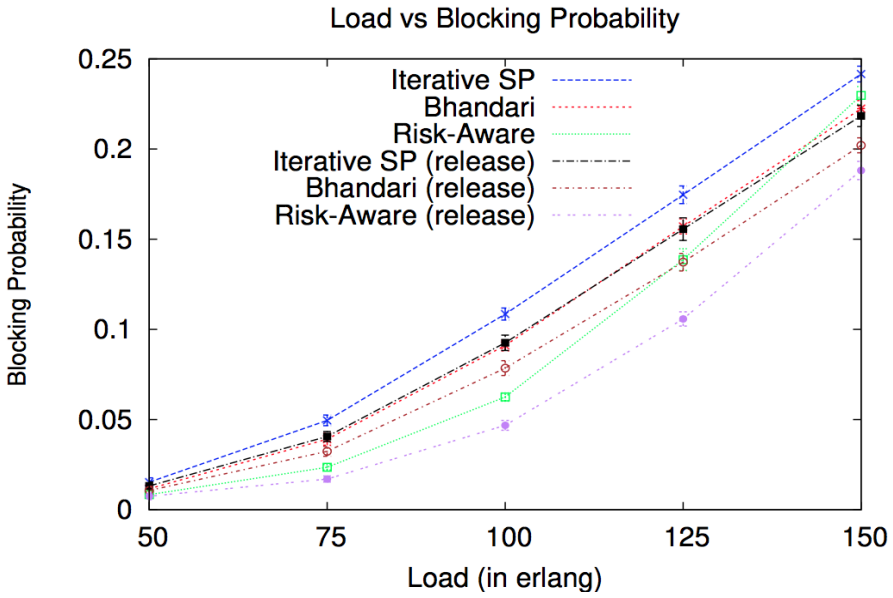


Figure 6.4. Comparative analysis of Iterative Shortest path, Bhandari shortest path Risk aware PDID for blocking after releasing resources of failed connections

As you can see, the Iterative Djikstra with release of resources shows similar blocking characteristic with Bhandari without release. This is an indication of how poor iterative Djikstra performs for blocking. At the same time, risk-aware Risk aware PDID without release performs much better than Bhandari with release at low loads but cuts very earlier at moderate loads. Likewise, risk-aware with release does not seem to cut Bhandari with

release because it's still operating at lower loads as a result of releasing the resources of failed connections. Thus, by releasing the resources you can prolong the benefits of risk-aware Risk aware PDID.

CHAPTER 7
CONCLUSION

Although Bhandari does better than other two at higher loads, it still uses more resources and blocks more than Risk aware PDID at lower loads. Using iterative SP and risk aware PDID, the average resources for all the connections is brought down by 1 for both primary and backup. While this may not be beneficial for link failures, it may be a crucial factor for node failures. Since Bhandari distributes the load better than others, it has high chances of connection disruption due to nodal failures. This thesis argues, a slightly modified iterative approach at lower loads can offer less blocking with a neglectful increase in failure. Results showed that Risk aware PDID could benefit network operating at 13 % blocking for the traffic characteristics discussed in chapter 3 and 4. Table 7.1 tabulated the results obtained above based on considered parameters in this research,

For this network, the working range of load is found out after so many trials. However, this threshold can be easily obtained for the intended traffic behavior. Since we considered that connections are set in advance, it is very complex to predict the future traffic behavior and apply the approach. Also, it is very hard to see a load invariant traffic characteristics in a network. That said, the approached could still be applied to varying loads in the network

Table 7.1. Load driven survivable approach

QOS	Holding Time	Survivability approach	Load
High priority	Long	Bhandari	High
High priority	Short	Iterative shortest path	Low
Low priority	Long	Bhandari/Risk aware IPDSP	Low
Low priority	Short	Risk aware IPDSP	Low

depending on the holding time of the connection as it may vary from minutes to even years. For example, if the connection has short holding time requested at lower operational load, Bhandari may not be clever option. At the same time, if the holding time is longer and the operator is unsure of future traffic characteristics, Bhandari is the safest option. However, an intensive research should be conducted based on intended input traffic characteristics, failure characteristics and load in order to apply the above approaches.

LITERATURE CITED