uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Shahram Shah-Heydari
AUTEUR DE LA THÈSE / AUTHOR OF THESIS

Ph.D. (Electrical Engineering)
GRADE / DEGREE

School of Information Technology and Engineering
FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Self-repairing Hierarchical Tree-based Link Restoration Scheme for Mesh Networks

TITRE DE LA THÈSE / TITLE OF THESIS

Oliver Yang
DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

W. Grover (teleconference)

N. Zaguia

H. Mouftah (teleconference)

M. Woodside

Gary W. Slater
Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

University of Ottawa

SELF-REPAIRING HIERARCHICAL TREE-BASED LINK RESTORATION SCHEME FOR MESH NETWORKS

Author: Shahram Shah-Heydari

Supervisor: Prof. Oliver W. W. Yang

A Thesis Submitted to the Faculty of Graduate and Postgraduate Studies

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

School of Information Technology and Engineering

University of Ottawa

Ottawa, Ontario, Canada

Your file Votre référence
ISBN: 978-0-494-32419-6
Our file Notre référence
ISBN: 978-0-494-32419-6

ABSTRACT

In this thesis we propose and study a self-repairing tree-based pre-planned link protection scheme for backbone mesh survivable networks. We introduce the concept of hierarchical protection trees for unicast traffic protection on network links. The design of the protection tree is formulated and heuristic algorithms are provided for two separate scenarios: (1) the unrestricted spare capacity assignment where the minimization of spare capacity is the objective, and (2) the pre-designed network scenario where the objective of design is to maximize the total network restorability. Using various models for computation, we apply our design techniques to construct the protection tree in each sample network in each scenario, and compute various restoration performance measures of restorability, redundancy, and average backup path length.

We conduct extensive studies of the performance trends of the hierarchical protection trees on a database that we built from thousands of randomly generated mesh networks with varying topology parameters such as network size, number of nodes, average nodal degree, network radius, and variation in network load.

We have also extended our study to the multiple (double and triple) failure scenarios. We present computational methods to calculate the multiple failure restorability and redundancy requirements for the sample networks from our database.

Finally, we provide a comprehensive scheme that covers the signalling, message formats, algorithms and recovery mechanisms for construction of a hierarchical tree in a distributed manner. The advantages and limitations of this new approach are also discussed.

ACKNOWLEDGEMENT

I would like to thank the members of the thesis committee who kindly spent their valuable time to review my Ph.D. thesis. I am also grateful to my supervisor, Professor Oliver Yang, for his guidance and technical vision, and to my colleagues in Computer Communication Network Research (CCNR) group at University of Ottawa: Wail Mardini, Yuna Zhang, Haomei Liu, Donna He, Peifang Zhou, Mushi Jin, Yiming Zhang and Zenxiao Liu for their continuous help and support.

I have had the privilege of working at Nortel Networks with some of the greatest minds in this field. Nortel partially supported my studies through employee educational assistance as well as direct contributions to the CCNR group. Guo-Qiang Wang and Kent Felske from the Emerging Technology group worked directly with CCNR, and their feedback helped me enormously. I also benefited from the experience and knowledge of my manager and colleagues in Nortel's ASON group: David Martin, Stephen Shew, Mark Allaye-Chan, Gail Huang and Tim Armstrong.

I would also like to thank my wife Mariam and my little children Soroush and Saman for their loving support, as well as my parents to whom this work is dedicated.

And now I start this manuscript in the name of our creator, in whom we trust and on his kindness we rely. May his blessings be with us throughout our lives.

"In the name of God, the most compassionate and merciful".

Algorhyme

I think that I shall never see
A graph more lovely than a tree.

A tree whose crucial property
Is loop-free connectivity.

A tree which must be sure to span.
So packets can reach every LAN.

First the Root must be selected
By ID it is elected.

Least cost paths from Root are traced
In the tree these paths are placed.

A mesh is made by folks like me
Then bridges find a spanning tree.

(*Radia Perlman, the inventor of the Spanning Tree protocol* [Perl85])

Table of Contents

List of Figures

List of Tables

List of Symbols, Notations and Acronyms

		Section of 1st appearance
C_p:	Vector of Protection Capacities	2.1
C_w:	Vector of Working Capacities	2.1
CSF:	Capacity Sharing Factor	2.5
D_{min}:	Minimum Nodal Degree	2.1
E:	Set of network edges	2.1
E_i:	Set of network edges incident to node v_i	2.1
Fc:	Failure Count	5.3
G:	Network graph	2.1
M:	Total number of network edges	2.1
N:	Total number of network vertices	2.1
R:	Network Restorability	2.5
Rp:	Restoration Paths Matrix	2.5
S:	Normalized Spare Capacity Increase	5.4
T:	Set of network edges that form the HP-tree	2.1
T':	Set of network edges that do not belong to the HP-tree	2.1
$T_{sub}(v_i)$:	The portion of tree *T* that forms the child subnet of node v_i	2.6
V:	Set of network vertices (nodes)	2.1
c_{pi}:	Protection capacity on edge e_i	2.1
c_{wi}:	Working capacity on edge e_i	2.1
c_{wpi}:	Protected working capacity on edge e_i	2.5
d_i:	Nodal degree of vertex v_i	2.1
v_r:	Root node of the p-tree	2.6
v_{ppi}:	Primary parent node for vertex v_i	2.6
v_{bpi}:	Backup parent node for vertex v_i	2.6

Chapter 1. Introduction

1.1. Overview

The explosive growth of communication demands in recent years, a direct result of the expansion of the Internet in every corner of the globe, has created the need of high-speed long-haul and metro networks with increasingly dynamic and arbitrary mesh topologies. The huge volume of traffic (data, Voice, video on demand etc) carried by backbone networks requires special attention to the issue of network recovery and protection against node and link failures, because interruption of this large traffic flow (and as a result, the offered user services) could create significant problems that would cripple businesses and cost millions of dollars. Fast restoration of traffic in case of a failure is essential, be it a fibre cut, node failure or higher layer service point failure. In fact, standard organizations have specified maximum acceptable down times for various services as shown in Figure 1-1 [T1A193]. Based on that reference, a voice call may be dropped after a 200 milliseconds interruption, while a data session typically times out within 2-300 seconds. Interruptions of 50 milliseconds or lower are usually transparent to network services.

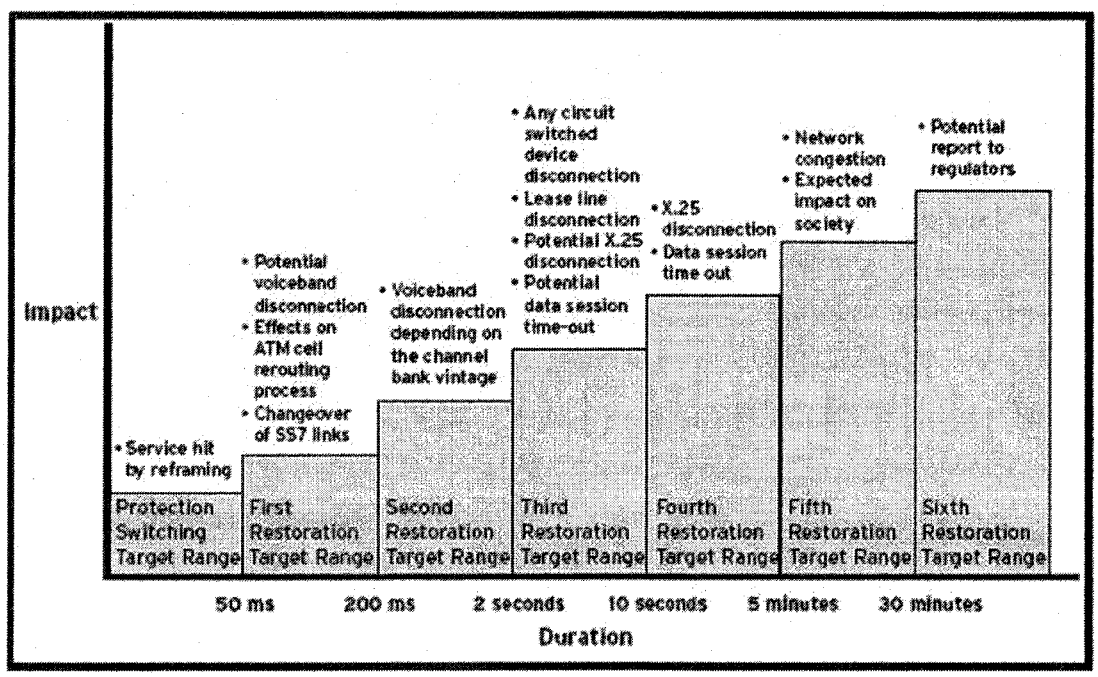


Figure 1-1: Impact of Interruption on Network Services [T1A193]

Accordingly, the level of service restoration offered by network service providers could depend on the type of the service. A *gold* class of service usually refers to dedicated protection [LeGr02] and offers restoration times of 60 milliseconds or less from the time of failure. This is typically the failure recovery time achieved by Automatic Protection Switching (APS) mechanisms in SONET networks [Ayan02], and guarantees against interruption for virtually any network service. Similarly, a *silver* class of service refers to shared link restoration [LeGr02] and might offer restoration times of 100 to 500 milliseconds, typically achieved by link redial (link restoration) mechanisms. A *bronze* class would offer restoration times of 2-10 seconds as typical for IP layer path re-routing. Cheaper classes of services might include unprotected services, or even pre-emptible service where the spare capacity of the network is used for transporting customer data, and the customer traffic could be interrupted if the spare capacity is required for a higher priority purpose, e.g. restoration of higher priority traffic. Our work in this thesis is focused on a shared link restoration scheme, a silver class of service per above classification.

In order to achieve restoration time targets for each of the above classes of service, various methods have been researched and proposed. Automatic Protection Switching in SONET networks can achieve a sub-60 milliseconds restoration time, with a window of 10 milliseconds for fault detection and 50 milliseconds recovery interval. In metro optical networks bi-directional line switched rings (BLSR) of up to 16 ADMs and ring circumference of up to ~1500 km are used while in long haul networks, 1:N protection switching is more dominant [Ayan93]. Both methods use the K1/K2 byte APS protocol to handle the required signalling for protection switching.

Figure 1-2 shows the basic operation of 1:N protection switching and BLSR protection ring. In 1:N protection switching method, N fibers are protected by a spare protection fibre. BLSR could be implemented with 2 or 4 fibers. In BLSR case, the ADMs adjacent to the failure perform a loopback function to switch the working capacity to the protection capacity [Ayan93]. Both BLSR and 1:M APS methods protect against a single fibre cut only.

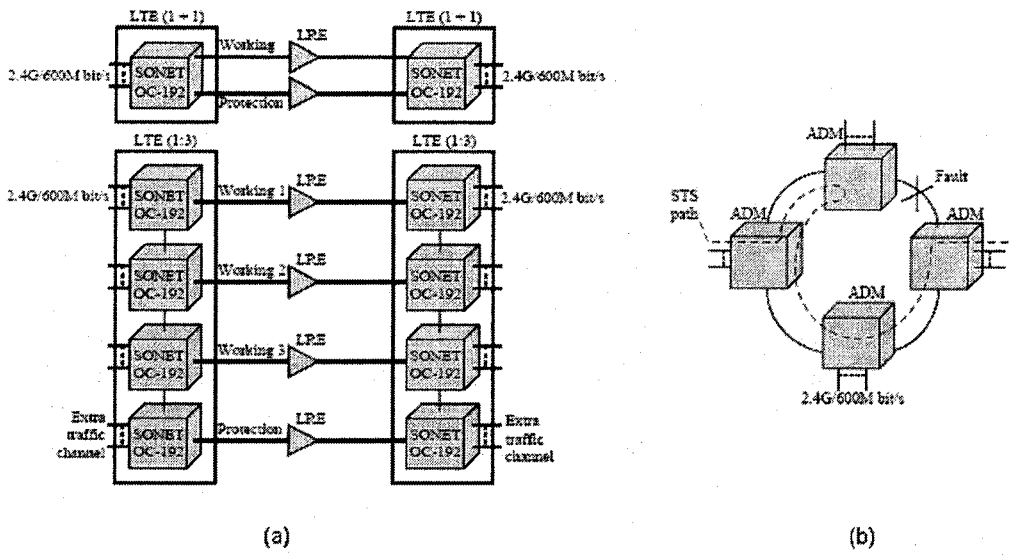


Figure 1-2: a) 1:N Protection Switching, b) BLSR Ring [Mori98]

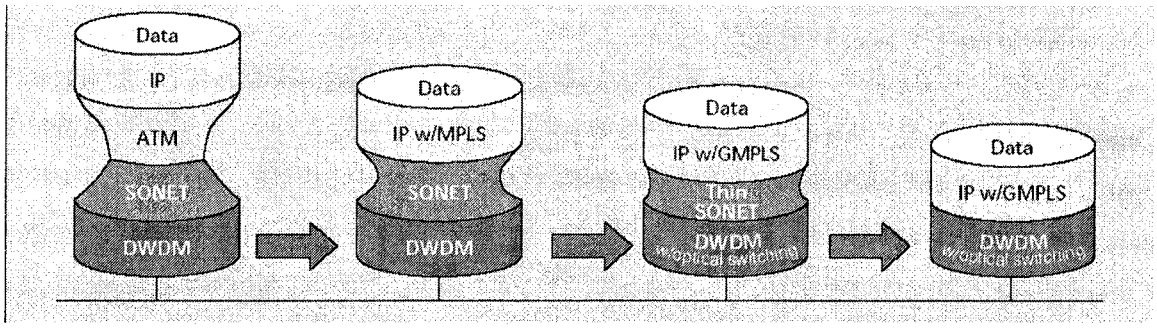


Figure 1-3: Evolution of Network Architecture [BaDr01a]

For the silver and bronze classes of service, mesh protection/restoration methods can be used. Originally, research works on network protection made a distinction between *protection* and *restoration* [Ayan02, RaMu99a]. The Protection category included methods where backup paths were pre-computed or pre-connected, and sometimes the traffic was sent on both primary and backup paths and the best signal was chosen at the destination (such is the approach in 1+1 protection switching and UPSR rings). On the other hand the Restoration category included methods where the backup paths where computed or discovered after network failure, based on the current state of the network. Such categorization also fit the traditional layered network model [BaDr01a] as seen in

Figure 1-3, as well as overall IP/ATM/SONET architecture of backbone network. In this model, protection methods were mostly applied in the core network components that implemented the lower layers of the network model, mostly at the SONET/DWDM (and occasionally ATM) layers. These protection algorithms worked with actual physical components of the network and had to achieve fast response to component (node, link or wavelength) failures. Restoration techniques, on the other hand, were deployed more in the IP/ATM layers of the edge routers and mostly relied on dynamic computation of backup routes based on link state information often available at the network layer (through OSPF, BGP and other similar protocols).

However such distinction between protection and restoration is becoming less relevant. The term *protection* was originally used for schemes such as the 1+1 protection scenarios where the source node would send the traffic on both primary and backup paths, and the destination node would choose the channel with the best signal. This operation allowed continuation of traffic without interruption. The term *restoration* was used for the cases where the traffic would be restored after failure interruption. In recent years some references have started to use *protection* for schemes that reserve backup capacity in advance, and *restoration* for those schemes that do not reserve back up capacity [YaFi04], even though in such categorization a *protection* scheme may still *restore* the traffic only after the failure, as most pre-planned protection schemes do. Also as noted in [Ayan02], the move from the IP/ATM/SONET/DWDM model to IP/SONET/DWDM, and eventually to IP or GMPLS/DWDM, would create a convergence of higher layer restoration and lower layer protection schemes, resulting in an integrated service management that could manage and offer different classes of protection/restoration services. In order to avoid confusing terms, in this thesis we shall commonly use the term *restoration* even in cases where pre-computed and pre-planned backup paths are used.

We also focus on *mesh* restoration. By "mesh" we generally mean that the average nodal degree in the network is larger than two, and therefore the network topology is not ring-shaped. Mesh Restoration techniques are categorized based on their originating point, their use of capacity, and whether the backup paths are computed statically or dynamically [RaMu99a].

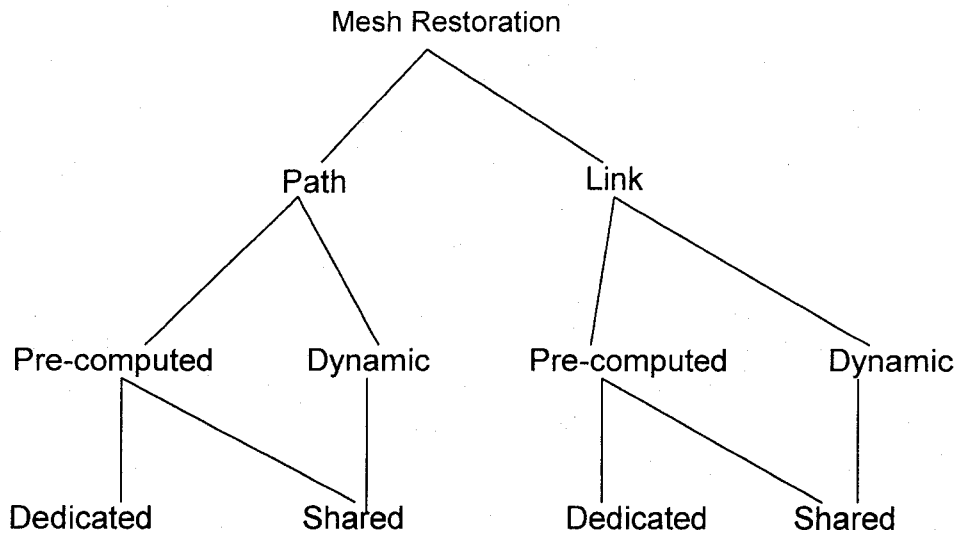


Figure 1-4: Categorization of Mesh Restoration Schemes

Figure 1-4 shows the categorization of the mesh restoration schemes. At the top level, Failure recovery schemes in mesh networks are generally categorized as *Path Restoration* schemes and *Link Restoration* schemes [RaMu99a, RaMu99b]. Path restoration schemes handle a link or node failure by restoring the affected connections individually from their source nodes by calculating new routes. The mechanism for finding the restoration route and establishing the connection is typically based on end-to-end path computation and signalling.

Link restoration schemes, on the other hand, do not route the failed connections individually from the sources. Here the objective is to find an alternative path around the end nodes of the failed link, and to switch all connections on the failed link to the new route as a bundle. The term *link* is used broadly here; it could refer to a multi-fibre span, a single fibre, a single wavelength on a fibre, or even a higher layer logical connection. As such, different wavelengths on a fibre may be re-routed on different paths.

The restoration path in link or path restoration schemes could be pre-determined or decided dynamically once failure occurs. Failure Recovery schemes can use *dedicated protection capacity*, where the required bandwidth for restoration of the traffic is reserved individually for each connection or link, or *shared capacity,* where a pool of protection bandwidth is shared between connections or links, usually optimized to protect network traffic against single or multiple failures. Note that the distinction between shared and

dedicated protection capacity is applicable in the pre-computed case; In dynamic restoration mechanism, individual restoration paths are not identified prior to failure and thus no spare capacity is *reserved* in advance for a specific connection. Dedicated protection schemes would require a far larger amount of redundancy in the network, as the required spare capacity must be reserved on each hop of the backup path.

Combination of the above characteristics allows for implementation of various classes of failure recovery services that were discussed earlier in this section. For instance, pre-computed dedicated link restoration schemes achieve fast restoration times because no time is wasted for computation of backup paths, the spare capacity is already reserved, and the traffic is being rerouted as a bundle locally without having to inform the source nodes for redialling the connection. On the other extreme of the spectrum, shared capacity dynamic path restoration allows utilization of network resources based on the latest state of the network, *potentially* providing a more capacity-efficient approach, however the restoration times will be within seconds.

The above categorization of classes of services and Mesh restoration approaches allows us to 1) define our objective here clearly and identify what service we are targeting; 2) propose a scheme for shared-capacity pre-computed link restoration, and 3) provide a framework to categorize and compare various mesh restoration proposals appropriately, particularly in the literature review in the next section.

1.2. Review of Prior Work

Network failure recovery and restoration has been a topic of interests for many researchers over several decades, and a very large number of methods, algorithms and studies on various classes of protection and restoration have been done. In our review here we focus our attention on those studies that are more relevant to our proposal here. Our approach to network restoration is tree-based link restoration, so our literature review will focus on link restoration techniques, with a particular emphasis on those that use network trees for restoration of traffic. Therefore, this review does not cover the significant amount of prior research on path restoration algorithms Specifically, we review mesh survivable design techniques, ring and cycle-based techniques, and then discuss applications of multicast and unicast trees in network service restoration, root

node selection, dynamic reconfiguration, multiple failure restoration methods and distributed algorithms. For surveys of path restoration schemes, see [RaMu99b, XiMa99].

1.2.1. Mesh Survivable Design

Research on Mesh Survivability schemes has been conducted for about two decades by now. At first, such research efforts focused on taking advantage of connection management capabilities of DCS (Digital cross connect) nodes to re-route the connections of a failed link over k shortest-paths between the end nodes of the failed link or from the source node. Distributed protocols have been proposed to eliminate the need to maintain a central link-state database, such as the SelfHealing Network (SHN) proposal [Grov87]. The performance of SHN was studied in [GrVe90] using simulation of Telecom Canada network for which sub-second restoration time was achieved, and the study confirmed that restoration times in SHN network would increase with network size, in the worst case by N^2. In [DoMo94], the cost curve of DCS-based mesh restoration was studied and it was shown that the capacity optimization in this scenario was intractable. Sakauchi et al studied the distributed self-healing algorithm NETRATS (NETwork Restoration Algorithm for Telecommunication Systems) and stated that the NETRATS algorithm often failed to achieve full (100%) restorability even when enough spare capacity was present in the network [SaNi90]. Also, they addressed the mesh survivable design problem by formulating the spare capacity assignment (SCA) problem as an LP formulation, and introduced a maximum-flow heuristic approach based on finding all partial cutsets in the network. In general, heuristic algorithms are used to find near optimal solutions for the SCA problem, as the original mesh survivable capacity optimization problem is *NP*-complete [DoMo94] and thus difficult to solve for large networks. Another heuristic algorithm, Spare Link Capacity Algorithm (SLPA), used a phased approach: forward synthesis phase followed by design tightening phase [GrBi91]. The performances of the cutset heuristic (ICH) and the SLPA heuristic for SCA span restoration problem were compared in [VeGr03], where the complexity of ICH was shown to be exponential in the worst case, and complexity of SLPA to be $O(W^2M^5NlogN)$, with W being the total number of working links, M total number of spans, and N total number of nodes. This study also showed that the ICH design used less spare capacity than the SLPA design.

Herzberg and Bye presented an optimal SCA model for link restoration that also included the hop count limit; i.e. a maximum (in number of hops) could be set for the length of the backup path [HeBy94]. Their method approximates integer capacity numbers with real value. Their simulation result (one network example only) demonstrated the trade off between network redundancy and hop limit. More results in [DoGr01a] were presented for the case where the average backup path length was also minimized along with the total spare capacity of the network, indicating a significant increase in spare capacity when the mean backup path length was reduced beyond a threshold.

For the SCA problem, it is assumed that the network topology and the working capacity of each link are known, and then the shared spare capacity is optimized by finding the best backup paths for each unit of working capacity. However, capacity optimization can be improved even further if routing of working demands and assignment of spare capacities are done jointly. This problem is referred as Joint Capacity Assignment (JCA). The JCA problem formulation for path and link restoration scenarios is discussed in [IrMa98], where simulation results were used to show that path restoration could require less capacity than link restoration. A more recent study [LiTi05] introduced another heuristic approach to SCA problem that strikes a balance in the trade off between algorithm time complexity and efficiency of capacity optimization. The reference [IrMa98] also includes a fairly good survey of previous SCA algorithms.

In general, optimal solutions of mesh survivable design formulations for various scenarios are often used as yardsticks for evaluating the performance of other equivalent heuristic methods.

1.2.2. Ring-based Mesh Restoration

Ring architectures were widely used in SONET protection schemes as BLSR rings or USPR rings [Wu92], and many research efforts have focused on re-using ring-based protection mechanisms of SONET networks in mesh networks as well. An example of such schemes is the concept of *ring covers* [Elli00] where protection rings are constructed in a way to cover every link of the network. The approach in [Elli00] includes dividing each undirected span into two directed arcs, and then forming directed rings over directed arcs to create an overall Eulerian protection architecture. It is shown

that a planar graph can be decomposed into such directed rings in a way that each directed arc is used only once in one ring. The paper provides algorithms for forming ring covers in planar and non-planar graphs. In [GaHe94] two ring cover design problems have been discussed: The Min-Sum Ring Cover (MSRC) problem that finds a ring cover with minimum sum of the weights of the links in the ring cover set, and the Min-Max Ring Cover (MMRC) problem that finds a ring cover with minimum sum of maximum weight of each individual ring in the ring cover set. The MMRC problem essentially attempts to minimize the redundancy of the bi-directional network. The paper suggests that the MMRC problem is NP-complete and thus intractable, however the MSRC problem can be solved in $O(N^3 logN)$ time for networks whose nodes can be covered by two edge disjoint spanning trees.

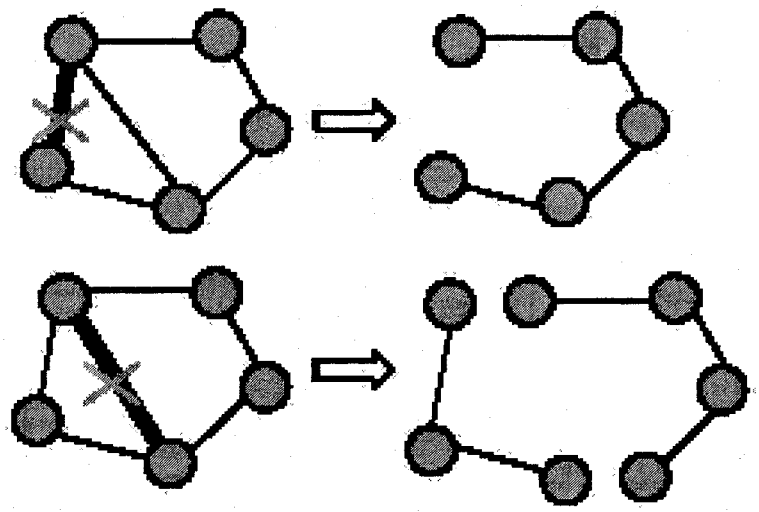


Figure 1-5:Restoration of cycle and straddling links [Zhan04]

An extension of ring-based architecture was proposed in [GrSt98], in which the graph cycles would provide protection paths for cycle links and non-cycle links (called straddling links). The cycle could provide one backup path for each link on the cycle, and two backup paths (one path clockwise around the cycle and another path counter-clockwise between the two end nodes of the straddling link) for each straddling link, as shown in Figure 1-5. Therefore in this case the set of protection cycles does not have to traverse every link of the network; just every node of the network. Uniform cycles were formed from individual capacity units on network links [GrSt98]; thus capacity units on a span maybe protected by different protection cycles. However, proposals based on

forming cycles out of whole spans (as opposed to unit capacity cycles) have also been studied [HuCo02]. Protection cycles can be pre-connected so that link failures can be restored quickly with minimal signalling.

The problem of design of protection cycles for minimal redundancy was also studied in [GrSt98] and an ILP formulation of uniform protection cycle design was presented that required generating first a set of all simple distinct cycles in the graph. An alternative ILP formulation without such cycle enumeration has been proposed by Schupke [Schu04], who also indicated that such problem was NP-complete and thus intractable [Schu05]. According to the same source, the complexity of the ILP formulation with cycle enumeration was unknown. Schupke also presents formulations for various scenarios such as design with limited capacities in [Schu05]. In most cases, the capacity restriction is applied through a limit on total spare capacity available in the network, or a joint working-protection capacity optimization problem is solved in order to create a design that best utilizes the available resources in the network.

Some performance results from these researches indicate that the protection cycle design could approach the capacity efficiency of optimal mesh design. However, the complexity of the ILP formulation would become prohibitive for medium size or large networks especially as the number of candidate cycles in the graph grows exponentially with the size of the network [DoHe03]. Several heuristic algorithms have been proposed to speed up the process of choosing the cycles in the network for protection. The *straddling link* algorithm (SLA) [ZhYa02c] greatly increases the speed of protection cycle design by limiting the search to only a set of small cycles based on each network link along with two shortest paths (if they can be found) around it. However as a result, the capacity efficiency of the resulting protection cycles is no longer optimal. The time complexity of the SLA algorithm is $O(MN\ logN)$ [DoHe03], or $O(N^3 logN)$ in the worst case. These cycles are referred to as primary cycles [Grov03] and are used as an initial set in an extended algorithm [DoHe03] to *Expand* or *Grow* bigger cycles. Results for two samples, US networks and France Telecom networks indicated that the design problem could be solved in a much shorter time but at the cost of higher redundancy in the network.

Limitations on the length of the backup paths in protection cycles have also been a subject of research. Backup path length for cycle links increases linearly with the length of the cycle, and simulation results in [DoHe03] indicate fairly large cycles with average length of up to 15 hops for US network (with total number of 28 nodes), and up to 20 hops for France telecom network (with total number of 43 nodes). The ILP formulations for cycle design could include cycle length limits [KoSa04], however for hop limits below the threshold limit the capacity efficiency of protection cycles sharply decreases in comparison with optimal mesh designs with the same hop limit, with the required redundancy going up to twice of that of an optimal mesh design when the hop limit was set to three hops in a network of 19 nodes.

While solving cycle design problems implies a centralized approach, distributed algorithms and signalling protocols have also been proposed to construct, reconfigure and maintain protection cycles in a mesh network, for instance in [StGr98]. The concept of pre-connected protection cycles has also been used for IP layer protection [StGr00b].

Recently there have been proposals to use cycles in *Protected Working Capacity Envelopes (PWCE)* [Grov04, ShGr04, Schu04a]. The objective of PWCE is to address a practical issue in deploying shared mesh restoration schemes: the administration of establishing and updating backup paths for each working channel in the network in a dynamic environment. In particular when implementing end-to-end shared backup paths, the new link states will have to be distributed to all network nodes and protection paths updated every time there is a change in traffic demand. The proposal for creating protected working capacity envelopes (PWCE) promises to reduce the administrative overhead of updating and maintaining protection structure of the network [Grov04]. In this proposal, the available working channels on each link are identified in advance as protected through pre-planned protection paths. Therefore there is no need to compute backup paths for an incoming demand as long as it is routed through a protected working channel. As a result, network nodes do not have to maintain a global, dynamic link state database to check the availability of protection capacity for new demands. The protection structure can be planned in advance with required engineering margin to tolerate fluctuations in traffic volume, and could be updated in longer intervals upon projected growth in demand or changes in topology.

It could be said that the PWCE proposal enables path restoration schemes to separate the computation of backup paths from the dynamic traffic demand in the network, a benefit that was restricted to pre-planned link restoration schemes before. On the other hand the blocking probability for a PWCE proposal based on a given protection scheme would be higher than the blocking probability for the same protection scheme if it was optimized dynamically, as shown in [Schu04a] for protection cycles. Other results [ShGr04] compared the blocking probability of PWCEs based on protection cycles with that of dynamic shared backup path plans to conclude that PWCE could outperform SBPP in some cases, although it noted that the improved performance in this case was mainly related to the protection routing mechanism and network-wide optimality of this PWCE implementation in comparison with the SBPP method.

1.2.3. Tree-based Algorithms

Trees have been used in a variety of applications in telecommunication networks, in particular in broadcast/multicast routing and traffic restoration. While in unicast routing, communication demands are made between each pair of nodes (a source node and a destination node for each demand), in multicast routing each source node sends its traffic to a group of nodes in the network. In broadcast routing, each source node sends its traffic to all network nodes.

In restoration studies, the above distinction allows for different focus points between multicast and unicast scenarios. In multicast scenarios the traffic is unidirectional from one or more sources to other network nodes, thus the nodes can be separated into *sender* and *receiver* nodes. The purpose of restoration would be to maintain connectivity between each receiver node and its respective sender node. The traffic within a multicast group is homogeneous. Multicast routing and restoration algorithms also take advantage of the fact that in most scenarios only a few sender nodes communicate with a large group of receivers, thus making decisions about root nodes easier. In unicast routing, however, traffic is bi-directional and each node is equally sender and receiver. Link restoration would require re-connection of all connections on the link from difference sources to different destinations and providing protection capacity for different traffic requirements of each connection.

While unicast restoration is the subject of our study in this thesis, we provide a review of multicast trees because graph trees have been used most extensively for multicast routing and restoration, and many ideas are often common between the two scenarios, such as tree growing methods and core selection.

In case of multicast routing, the purpose is to construct a tree with its root at the source, providing paths to every receiver node of the multicast group. A brief survey and performance comparison of multicast routing algorithms can be found in [WeEs94] in which multicast trees are classified into Shortest-Path Trees (SPT) that minimize the delay, and Steiner Minimal Trees (SMT) that minimize the cost. It has been shown that the problem of finding a Steiner minimal tree is NP-complete [Wint87] except in broadcast case (where the multicast receiver group includes every node of the network except the root node) in which it is reduced to finding the minimum cost spanning tree in the network. Approximations for SMT problem have been presented in [Wint87] with algorithms of time complexity of $O(pN^2)$, where p is the size of the multicast group and N is the order of the network. However most of these algorithms assume a static multicast group where the topology of the network and members of multicast group would not change dynamically. Performance results of such approximate SMT trees have been compared to SPTs and Centre-based Trees using simulation of random graph topologies and multicast groups [WeEs94].

An important issue in design of multicast trees is about the amount of information needed to be maintained in each node about each multicast group and its corresponding multicast tree. As the number of multicast groups and applications increase in a network, maintaining a multicast tree for each group may pose a scalability issue. One proposal to deal with this problem is through the use of core-based trees [BaFr93]. In this case instead of having one multicast tree per group, a single multicast CBT is constructed in the network for routing of all groups from their source nodes to their receiver groups. The trade-off here obviously is delay (and cost) optimization, as a CBT would not be as delay (and cost) optimized as trees specifically constructed for each multicast group. However this approach would resolve the scalability problem. A disadvantage of this approach is the single point of failure at the core that would affect all multicast traffic. More recent proposals (e.g. [ScCh00]) have addressed the issue of fault recovery in core-

based trees by enhancing the proposed protocol in [BaFr93] to avoid loops in case of link failures.

Another important area of research regarding core-based trees is on how to choose the core of the tree, as in this case many source nodes route their multicast traffic through a single tree. The original paper [BaFr93] proposed a few arbitrary selections, such as randomly choosing one of the source nodes as the core of the tree. A more detailed research was presented in [CaZe95], where three scenarios of multicast applications were considered: 1) the *All receivers/sources* scenario where every node is both a source and a receiver, e.g. videoconferencing applications, 2) the *single source/distributed receivers* scenario such as video broadcast on the net, and 3) the *local receivers* scenario where the receivers are mostly co-located in the same neighbourhoods while sources can be anywhere in the network, such as a database client-server application. The paper considers four methods for selecting the core node: Arbitrary, Random, Topology-based and Group-based. In case of arbitrary selection the worst-case scenario is studied, i.e. how bad would we do if we just pick any node as core. Topology-based method usually involves choosing a node at the centre of the network, i.e. with average minimum distance from every other node. In case of group-based selection, specific characteristics of the multicast groups are taken into account and the best core is selected. Performance of these methods has been compared [CaZe95] based on delay and bandwidth results from simulation of random networks. It is shown that a topology-based approach can often achieve the best delay except in local receiver scenario where group-based core selection performs better. In all scenarios, group-based core selection achieved the best bandwidth results. However the complexity of the group-based core selection is much higher than other approaches [CaZe95].

Dynamic reconfiguration of core-based trees has also been studied in [RaRa96], where dynamic assignment of the core node was considered. A parameter was introduced to measure the change in the network. For minimal changes, the core would remain unchanged. For moderate changes, the core would be moved one hop toward the location of the change. For significant changes, the core would be recomputed. An algorithm with complexity of $O(N^2)$ was proposed to implement this approach.

The issue of reliability and protection of multicast traffic has been a subject of intense study in the past two decades as well. It is important to note that when a tree is used for multicast traffic routing, any single link failure on the tree would disrupt traffic to part of the network because there is only a single path from the source (root node) to any of the receiver nodes on the tree. An approach was proposed in [ItRo88] to construct two spanning trees in the network in a manner that there would be two edge-disjoint paths from each node to the root node of the tree. The two trees may share some edges though. This approach was further extended in [ZeIt89] by presenting an algorithm to construct three trees to guarantee three disjoint paths between root node and other nodes; and the authors suggested that it might be possible to construct k spanning trees to provide k paths accordingly.

The multi-tree approach was further extended by Medard, Barry, Fin and Gallagher [MeBa99] by proposing algorithms (hereafter referred to as MBFG algorithms) to construct two pre-planned directed spanning trees (red/blue trees) in a network in order to restore multicast traffic in case of both node and link failures. In the MBFG approach, one tree (called red tree) would be used for routing of working traffic, and the other tree (called blue tree) would be used for protection. An example of red/blue trees constructed by this algorithm is shown in Figure 1-6 where the thin arrows indicate the working tree and the thick arrow indicate the protection tree. The MBFG algorithm starts by choosing a cycle of any size in the graph and constructing two directed paths, clockwise and counter-clockwise, on the cycle from the root node to all other nodes on the cycle, and tagging the nodes on the paths along the way. Then the algorithm proceeds by constructing paths from one tagged node on the cycle to another tagged node on the cycle, with all intermediate nodes being untagged, and adding the resulting paths to the red and blue trees, until no untagged node is left in the network. The MBFG algorithm has a complexity of $O(NM)$, or $O(N^3)$ in worst case.

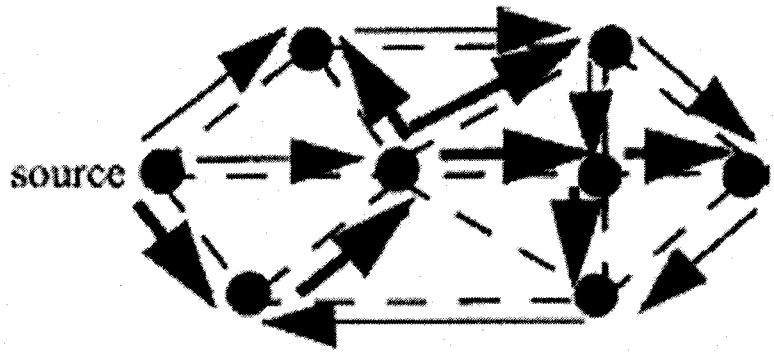


Figure 1-6: Redundant Trees for Multicast Protection (from [MeBa99])

Furthermore, Medard et al established an important basis in their paper for determining the generality of a tree construction algorithm. A tree construction scheme would be a subset of another scheme if the second scheme could generate all trees generated by the first scheme, and if there are trees generated by the second scheme that could not be generated by the first scheme. Medard et al used this principle to demonstrate that the MBFG algorithm could generate trees that the method in [ItRo88] could not, and that every tree generated by the multi-tree approach in [ItRo88] could also be generated by the MBFG algorithm; therefore the method in [ItRo88] was a subset of the MBFG algorithm. However their work only addressed the construction of the red/blue trees. Performance results were not presented in their paper, and no optimization method for the red/blue tree was proposed. Also the issue of how to select the original (first) cycle was not addressed. Medrad et al further generalized their method into a link restoration approach called general loopback recovery where two digraphs (directed subgraph) in the network would provide two edge-disjoint paths between the end nodes of each link [MeBa02]. This study however did not present any result with regard to capacity efficiency or restorability of the generalized algorithm, nor did it discuss the complexity of the algorithm.

Xue et al extended the research on MBFG algorithm by presenting several heuristics to construct pre-planned red/blue trees that would minimize average delay or maximize bandwidth bottleneck in multicast operation [XuTh02, XuTh03]. Performance results were presented to show the improvement in delay and bandwidth over the MBFG algorithm. The complexity of their algorithms were: $O(N^2(M+N\log(N))$ for using

minimal delay, $O(N^2(M+N))$ for using minimal cost, and $O(NM)$ for using maximum bandwidth. So in the case of optimal delay and cost scenarios, Xue's algorithm had a worst case time complexity of $O(N^4)$, an order of magnitude higher than the MBFG algorithm.

Another method for constructing two edge-disjoint trees was proposed in [YeOf94]. The objective in that paper was to conduct convergence routing in switch-based LAN environments, which requires a deterministic bound on maximum route length. Routing was performed through virtual Eulerian rings embedded in each tree and the results in the paper indicated that routing through multiple spanning trees as opposed to single spanning tree improved the maximum rout length. The algorithm, it self builds two spanning trees in the network and then eliminates the common edges between them. It however, required four-edge connectivity in the network, and runs in $O(N^3)$.

The methods that we have described so far address the issue of restoration in multicast networks. Restoration of unicast traffic is essentially different (and more complicated) than multicast. In multicast operation the objective is to maintain reachability from a source node to a number of receiver nodes, and the flow of traffic from the source to the receivers is identical. The routing of the working traffic is often performed through a tree as well and thus the problem is simplified into finding trees that could supply two edge disjoint paths between root and each node. In case of unicast traffic protection, the working capacity could be routed based on any possible algorithm: Shortest path, weighted shortest path, custom routes or even random. Unicast trees for link restoration must supply a backup path between end nodes of each link in the network, and assign sufficient spare capacity to be able to route all working capacities of every link that might be using this segment of the path.

Such trees have been proposed for the design of Metropolitan Area Networks [YaMa86]. Variation of the simple tree topologies have been studied for improved terminal-pair reliabilities [Yang92]. Detailed performance analysis of these trees in terms of traffic intensity and mean path length were conducted and their performance compared [Yang94].

A study of various pre-connected homogenous patterns for unicast link restoration in a given network was conducted in [Stam97, StGr00a]. Each individual unit bandwidth (wavelength) was protected using a loop-back through a pre-defined *pattern* in the network. The network was modeled with pre-determined working and spare capacities on each link. As examples of such patterns, trees, cycles and patterns generated by genetic algorithms were compared. In order to generate the trees, the algorithm first generated the *k*-shortest pathset for each link in the network. Then each link was given a weight based on the number of times that link had been used in the *k*-shortest pathsets of all links in the network, in order to maximize restoration capacity sharing. Then the algorithm proceeded by executing Prim's algorithm to find a single unit maximum weight spanning tree on the spare capacities in the network. Removing those capacity units, the algorithm build another tree and continues until no further tree could be built. A 2^{nd}-step restorability was introduced to protect those working units that are still unprotected after the 1^{st} step through the shortest paths within the remaining spare capacity of the network. Note that in an unlimited spare capacity assignment (SCA) scenario the 2^{nd} step restorability is not applied because the network designer could allocate all required spare capacity where needed.

The above algorithm does not build a single protection tree, but a set of tree-like patterns on the spare capacity units of the network. The complexity of the algorithm was not specified. The design algorithm required pre-compiling and maintaining a database of tree patterns in the network. A number of interesting results were reported in [StGr00] regarding the use of trees as the base pattern for pre-configured protection. For instance it was noted that cycle patterns outperformed the tree-based patterns with regard to restorability in a fixed capacity network, and that the number of cross-connections to be made was less than the conventional k-shortest paths approach. A more theoretical discussion about the selection of such tree-based unit-capacity patterns versus cycle-based unit-capacity patterns was also presented in [StGr00], although it is important to note that this study focused on the lower bounds of network redundancy based on the number of useful restoration paths available at each node through each pattern type. In practice, actual redundancies in a given network could be a lot higher than those bounds.

A tree-based protection scheme was proposed by us in [ShYa01, ShYa04a] for the purpose of unicast traffic protection in mesh networks. The focus of this study was to provide a simple and self-repairing protection method that would avoid the computational complexity of pre-connected patterns in [StGr00]. The underlying method was based on assigning a primary and a backup parent node for each network node in a way that the primary backup paths were organized in a spanning tree structure. The backup parent nodes were used to repair the tree in case of any tree link failure. We called this scheme the "hierarchical protection" tree. The research in this thesis is focused on the study of the concept of these trees, heuristics for improving capacity efficiency and restorability, and their performance in multiple failure scenarios, presented in Chapters 2-6 of this thesis.

Several further studies have extended the original concept in [ShYa01]. The complexity of the distributed tree construction scheme was studied in [ZhSH02]. The results have been discussed in Chapter 6 in this thesis. A variation of the original tree construction algorithm was proposed in [ZhYa02a] to reduce the number of messages for construction of the tree, although this scheme uses shortest paths for protection of tree links instead of backup parent node. As a result the resulting spanning tree is not self-repairing. This approach could be considered a subset of the pre-configured tree patterns in [StGr00]. Further enhancement of the algorithms for this approach with regard to the number of messages was presented in [Zhan02a, ZhYa02b]. One of the approaches proposed in [Zhan02a] for reducing the message volume, was based on the concept of *triangular trees*. We will discuss this approach in more detail in Section 3.2.4 and will enhance it for our HP-tree design. A dual protection tree method was proposed in [LiYa03] in which two directed trees were constructed for unicast link protection. However no comparative performance results on restorability and spare capacity requirement were presented. A more recent study [TaRu05] examined the problem of partial restorability in the hierarchical protection trees and proposed a change in the scheme in [ShYa04a] in order to guarantee 100% restorability. In this approach the non-tree link that was used for protection of a tree link, did not have to be adjacent to the failure point; therefore the topological constraint of the hierarchical protection tree was avoided. However the resulting tree would no longer be self-repairing and could not be used directly for multiple failure restoration.

1.2.4. Multiple Failure Analysis

There have been many studies of the impact of multiple failures on availability and restorability of the telecommunication networks. Here we focus on those studies that are closer in objectives and methods to our objective for multiple failure analysis in Chapter 5 of this thesis. Multiple failure analysis of a network involves computing the restorability of the network when N links fail one after another, before the previous failure could be fixed. In link restoration schemes it is generally assumed that by the time of the second failure, the traffic on the first failed link has already been re-routed to the protection path [ScGr04]. However for path restoration, some studies have also considered the case of k simultaneous failures that in effect could limit the number of available backup paths [LiTi05]. Double failure cases have been studied for specific restoration schemes, for example in [ClGr02, LuMe01a] for k-shortest paths algorithm and in [Schu03] for pre-configured protection cycles, in which hypothetical scenarios were also considered where the protection capacities can be reconfigured fully or partially after each failure. Research has also been done on planning of spare capacities in a manner than guarantees full restoration of both single- and double-failures in a network both for the case where only the working capacity is protected and the case where protection and working capacities are protected (see [Schu05]).

A general classification of two-link failures can be found in [LuMe01], in which such failures have been categorized as fundamental failures, and basic or practical algorithmic failures. Fundamental failures are results of network design deficiencies from which no restoration scheme can recover; for instance multiple failures resulting in network partitioning. An example of algorithmic failures is a backup path hit, where a second failure breaks the restoration path for a first [LuMe01]. Full recovery from such scenario would require dynamic (non-static) backup path computation, which may not meet the speed of recovery requirements of the pre-planned link restoration. Another example is the case where consecutive failures hit two links that share segments of their backup paths, and those segments might have insufficient capacity to restore the traffic of both links. We will provide more details about our modeling of multiple failures in Chapter 5.

Another important distinction was made in [ScGr04] with regard to the way the restoration of working and protection capacity is managed. A protection scheme might

restore either both working and protection capacities, or only the working capacities. In the latter case (working-only restoration), the protection scheme restores working capacities after each consecutive failure, but does not restore the traffic that has already been rerouted through the protection capacity of the network in any of the previous failures. In the former case (which we call all-capacity restoration) all traffic either on main working paths or on backup paths should be restored when a failure hits their path. Obviously this approach would require a much larger amount of protection capacity in the network and more dynamic reconfiguration that might not be feasible in backbone networks that require fast recovery. Various levels of dynamic reconfiguration (from 5-10% to full reconfiguration) for achieving full double failure restorability were studied in [ScGr04] for pre-connected cycle-based protection scheme and their performance were compared.

An interesting scenario was presented in [DoGr02] in which instead of multiple-failure scenarios, a combination of maintenance+failure scenario was considered. The difference is that in such case the working capacity of the link could be rerouted fully or partially to the spare capacity of the same link; and if any spare capacity is left on that link, it could be used for restoration of failures from other links. Therefore it would be expected that a maintenance-immune design would provide higher restorability and lower redundancy requirements than a double-failure scenario. The results in [DoGr02] confirm this point. The paper uses the average link restorability instead of network restorability as the main performance parameter, and employs a mechanism for generating random graphs out of *master graphs* for random simulations. We will compare these methods with our approach in the Chapter 2 of this thesis.

In order to compute the average k-failures restorability in the network, we must calculate the restorability of the network assuming every combination of k failed links out of a total of M network links, and then average the results. This process in general would take $\frac{M!}{(M-K)!}$ computation of single failure restorability for each link. In [ShYa04c] a recursive algorithm was proposed for computation of multiple failure restorability for a given Restoration Path Matrix (see Chapter 2).

Full details of our multiple failure models will be discussed in Chapter 5 of this thesis.

1.3. Motivation

Our focus in this work is on pre-planned shared link protection schemes. We were specifically looking for methods that could compute backup paths in large networks in a reasonable time, in order to avoid exponential computational times as reported in [Schu05]. At the same time *manageability* of the protection scheme was also an important factor for us; i.e. a scheme that could be reconfigured, scaled, maintained and regionalized with ease while still providing sub-second restoration speed. We looked into graph trees for solution. Trees are local by nature and changes on one branch have a limited impact on other branches and higher layers in the hierarchy. This fact also provides simpler handling of multiple-link failure or node failures with tree-based protection schemes. A number of telecommunication protocols for constructing spanning trees already exists in various layers of today's networks and can be modified for construction of protection trees. Network state information and databases that could be used in tree-based algorithms have already been developed and deployed in the networks. It is easier to grow, modify, add branches or repair a spanning tree using the common protocols currently deployed in the mesh networks. In particular when static versus re-configurable networks are being considered, shared protection trees can be managed in a self-repairing manner, where a disconnected node can reconnect itself to the protection structure using primarily local state information and without the need to re-compute the complete set of backup paths for each link. Such computation can be done in advance and stored in the node so that no routing computation would be necessary after failure. As a price for manageability of the protection scheme, we expect tree-based schemes to provide slightly higher restoration times than pre-connected cycles, but still within the requirements of link restoration (a few hundred milliseconds), and that the capacity efficiency would be worse than optimal mesh and optimal cycle but better than 1+1 protection switching, and probably comparable with some non-optimal cycle designs obtained through heuristic methods.

Protection trees could be applied to various layers of the network architecture, in particular in the access networks. It must be noted here too that telecommunication networks are often designed with a core network of high capacity nodes that connect a large number of local access nodes. The topology is not exactly star-shaped, as a local

node might connect to more than one core nodes for reliability and routing efficiency. Tree-based protection allows simple localization of backup paths in such hierarchical architecture, and achieves good efficiency by having the high capacity nodes closer to the root of the protection tree (i.e. the concept of *hierarchical protection*). This point will be revisited again in Section 3.1 when we design our protection tree in a mesh network.

In summary, we would like to devise a complete tree-based scheme with the following characteristics:

- It defines a *link* protection approach.
- It provides *pre-planned* protection paths, hence allowing fast cross connect switching time from the end nodes of the failed link, without having to compute the full backup path post-failure or having to inform the source nodes for making routing decisions.
- Preferably could run on the network nodes in a distributed manner,
- It is based on a self-repairing spanning tree that would continue to provide backup paths for each link even after consecutive failures (as long as the network topology allows),
- Does not require high processing power at each node,
- Provides localization of changes, meaning that in case of a change or failure, the algorithm would run only locally in the affected region.
- Requires reasonable complexity that allows for scheme scalability.

We intend to address all of the above issues in this research.

1.4. Objectives

In this research we intend to provide a comprehensive study of the application of unicast hierarchical protection trees in mesh survivable networks. We study the advantages and limitations of the concept, present distributed and centralized algorithms for construction and optimization of the self-repairing hierarchical-protection tree (HP-tree hereafter) in various scenarios, and measure its performance using analytical approaches and simulation models.

The list of our objectives is as following:

1) Proposing heuristic algorithms for design of an HP-tree in an un-capacitated network with near-optimal spare capacity.

2) Proposing heuristic algorithms for design of an HP-tree in a pre-designed (capacitated) network with fixed link capacities.
3) Analyzing multiple failures in HP-trees.
4) Developing methods for analysis and performance estimation of HP-trees.
5) Designing a distributed algorithm for construction of tree-based link protection architecture in a mesh network, and for maintaining and updating a HP-tree in a distributed manner.

1.5. Approaches and Methodologies

Our approach in this thesis includes four steps: Concept, Construction, Efficient heuristic design, and Analysis of hierarchical protection trees. We first define and detail the concept of HP-trees. We address the question on how to build an HP-tree in different scenarios in a distributed manner, including details on signalling, message formats, sequences, path computation, recovery mechanisms, dynamic reconfiguration etc.

We approach the tree design problem by dividing it into two scenarios that a network planner may face when determining the link protection structure in a network. We study both following cases in our research.

Scenario 1: The spare capacity design scenario where the network topology and working paths in the network are pre-determined usually based on network traffic demand and shortest-path or custom routing. The objective is to plan shared spare capacity in the network in order to protect the working traffic of the network in case of single or multiple failures. For this purpose, the HP-tree should be designed to require minimum spare capacity.

Scenario 2: The capacitated (or pre-designed) network scenario where the network has pre-known working and (left-over) spare link capacities, often determined based on factors other than the optimal values required. Many real cases could involve designing protection schemes for an operating network (without previous mesh protection) where protection capacities would be determined based on the available capacities on individual links. Even in un-capacitated cases, it is unrealistic to assume that the switching nodes could be scaled at will to accommodate working and spare capacities without restriction. In such cases a designer might be interested to determine the best way of utilizing the available spare capacity of the network in an HP-tree for maximizing restorability

without having to modify the current state of the network (physical limitations, routes of the currently operating working paths etc).

We use the following tools and methodologies in our research:

1) Network modeling and assumptions: In our work on link protection schemes, we use a network model that primarily consists of network links with network nodes at their joints. The network nodes are considered as black boxes, so the internal components and operations of a node are not visible to us. We are only interested in the network layer view. Links are primarily identified by their capacities, and parallel fibers are bundled into one span.
2) Real and random network analysis: We study the performance of link protection schemes by applying the scheme to real networks and analyzing the restorability of the network in any single failure scenario. In order to get a more general view of the schemes' performances, we also use random network simulations. We generate network topologies with various sizes, nodal degrees and link capacity distributions. In order to increase the accuracy of our study, for each given set of network parameters (size, nodal degree, capacity distribution) we generate more than 12,000 of random networks and analyze the performance in each, and then use the average and standard deviation of the results for each point.
3) Simulation approach: We use OPNET network modeling tool [Opne70] to create behavioral models of network nodes and the state machines for our distributed algorithms. OPNET modeler provides a pseudo-C language for defining network objects, algorithms and messages in an event-driven message-exchanging environment. We use MATLAB® tool [Matl70] for statistical analysis of our simulation results, construction and analysis of restoration paths matrix and computation of various performance measures.

We also provide an approximate performance analysis of protection trees for special cases of pre-designed networks based on random network analysis. Our intention is to provide approximate closed form formula that could be used for estimating trends in the random networks, such as how restorability would change with network parameters.

1.6. Contributions

The main contributions of this thesis are the following:

1) The concept of a self-repairing shared capacity hierarchical protection tree (HP-tree) for failure recovery in a unicast mesh network.
2) Heuristic algorithms for spare capacity design of the HP-tree.
3) Heuristic algorithms for design of HP-tree in a capacitated (pre-designed) network.
4) Multiple failure performance analysis of HP-tree
5) A distributed algorithm for the construction of a HP-tree in a mesh network, and for maintaining and updating the HP-tree in a distributed manner.

1.7. Organization of this thesis

The remainder of this thesis is organized as follows. Chapter 2 discusses our network model and assumptions, performance parameters, models for random network generation, and the general concept of hierarchical protection tree.

Chapter 3 is dedicated to the study of spare capacity assignment for hierarchical protection trees. We formulate the capacity optimization problem and present a heuristic algorithm for this purpose. We also discuss the topological limitations of the HP-tree and present many simulation results.

Chapter 4 studies the issue of maximizing restorability of an HP-tree in a pre-designed network. We present problem formulation and present a heuristic algorithm to design the protection tree in this scenario. We use simulation of real and random networks to study the performance of our heuristic algorithm.

Chapter 5 presents an algorithm to compute the multiple failure performance of a dynamic protection scheme such as HP-tree. Our multiple failure model is discussed in detail here, and many simulation results are presented to study the performance of protection trees in multiple failures scenarios.

Chapter 6 presents a distributed signalling scheme for construction of a hierarchical p-tree is discussed. We present detailed discussion on the required signalling messages and roles of the nodes, and explain how the nodes would react to dynamic changes in the network.

We conclude this thesis and present ideas for future work in Chapter 7.

Chapter 2. Models, Definitions, Operations and Assumptions

In this chapter we will review various models, definitions and assumptions used throughout the thesis. This review will include basics of our graph theory representation of networks, our models for random graphs, demand models, assignment of working capacities and spare capacities, as well as restoration performance parameters that we will be using throughout chapters of this thesis. Then we will discuss our model for a protection tree, and explore the general characteristics of a protection tree as well the concept of Hierarchical-Protection tree model that will be used in this thesis.

2.1. Graph Theory Basics and Assumptions

We model a network as a connected undirected graph $G = (V, E, C_w, C_p)$, where $V=\{v_1, v_2, \ldots, v_N\}$ denotes the set of N vertices and $E=\{e_1, e_2, \ldots, e_M\}$ denotes the set of M edges. C_w and C_p denote the vectors of working and protection capacities of network links, respectively. The parameters c_{wi} and c_{pi}, $i=1,\ldots,M$ represent working and protection capacities on link e_i, respectively. Alternatively, we may also use (v_i, v_j) to represent the edge $e_k \in E$ that connects the two vertices v_i and v_j. Throughout this thesis, we use the term *node* to refer to a graph vertex. We also use the term *link* to refer to a network edge. All links are assumed to be bi-directional.

In our research we assume graph G is a *simple* graph, i.e. there could be only one link between each pair of nodes. However some algorithms and results are also applicable to cases where several physical links connect two nodes. These cases will be pointed out where discussed. The parameters c_{wi} and c_{pi}, $i=1,\ldots,M$ represent working and protection capacities on link e_i, respectively.

A network is of order N if it has N nodes. It is of size M if it has M links. A nodal degree d_i is associated with each node i to identify the number of links that are adjacent to this node. The average nodal degree of the network is represented by $\overline{D}$. The maximum distance between node i and any other nodes of the network is called the *eccentricity* of node i. The node or nodes with minimum eccentricity in the network is called the center of the network. The eccentricity of the center is called the *radius* of the network, and the

maximum node eccentricity in the network is called the *diameter* of the network [BuHa89].

A network graph is *k-connected* if removing k nodes from it could disconnect it. A graph is *k-edge-connected* if removal of k edges from it could disconnect it. If a graph is k_1-connected and k_2-edge-connected, and has a minimum nodal degree of D_{min}, then the following inequality holds, sometimes referred to as Whitney's inequality [BuHa88]:

$$k_2 \leq k_1 \leq D_{\min}$$

One of the original theorems with regard to graph connectivity (and with significant applications in network restoration studies) is Menger's theorem that stated [BuHa88]: "The minimum number of nodes separating two nonadjacent nodes s and t equals the number of disjoint paths between s and t." Based on Menger's theorem, the following theorem can be derived for k-edge connectivity:

Theorem 2-1- A graph is k-edge connected if and only if every pair of nodes is joined by at least k edge-disjoint paths [BuHa88].

A network cannot be protected (by any method) from every k subsequent or simultaneous failures unless the network is at least $k+1$-edge connected. We will discuss multiple failures models in detail in Chapter 5. In Chapters 3 and 4 of this thesis we focus on single failure restorability, which is a common case study in the literature. For successful full single failure restoration, the network must be at least two-edge connected. However, note that the connectivity requirement is necessary for full restorability but not sufficient, as algorithmic deficiencies might block a given scheme from achieving full restoration. We will provide more details on this point in Chapter 5, where we study multiple failure restorability results for our tree-based scheme.

If all paths between two nodes in the network share one link, then that link is called a *bridge*. Obviously a two-edge-connected graph does not include any bridge. Also note that a single edge-connected graph can still be restored from any single failure that does not impact a bridge in the graph.

Our work here is focused on *mesh* graphs, implying that some of the nodes in the graph have a nodal degree greater than two. A *full mesh* or *complete* graph is defined as a graph of order N in which every node has a nodal degree $N-1$, or in other words, there

exists a direct link between each pair of nodes in the graph. A full mesh network has a diameter and radius of one hop.

2.2. Network Topology Model

Throughout the rest of this thesis we will extensively use not only examples of real backbone networks, but also simulation of random graphs for generating results that could be used to identify trends and create estimations of average relationships between restoration performance parameters such as restorability, redundancy and average path length, and network design parameters such as minimum and average nodal degrees, network size, network order, link capacity factor etc. In this section we explain our method of generating such random graphs.

2.2.1. Existing Approaches

Many of the studies on protection networks rely on collections of real network topologies/demands that are available from the literature, or topologies arbitrarily put together that resemble a real network topology. However an alternative approach is to use simulation of randomly generated graphs with specific topological parameters, e.g. number of nodes, number of links, nodal degrees etc. The argument is that if the simulation sample space is large enough, the obtained results could be used to establish trends in performance of protection schemes versus various topological parameters. In this section we present an overview of some previously-used approaches and then in the next section we explain our method for generating a database of graphs for simulations.

In [Douc05] six artificially constructed master networks were used with orders of 15 to 40 nodes, each with average nodal degree of four, and then classes of networks of different sizes were generated by removing edges from each of those networks one by one. Several graphs of performance trends were generated in [Douc05] based on simulation of such families of networks. This method of generating *graph families* has the advantage that it could more accurately resemble the dynamic expansion or topology changes in a given network, thus proving insights into performance changes when a network topology grows or shrinks. However one should note that the graphs within a family are not independent with regard to their topological characteristics and so the resulting trend would be strongly related to the characteristics of the master graph. By

removing links one by one, most of the nodes retain their nodal degree and eccentricities, most working paths (and thus working capacities) remain unchanged except for those paths that used the removed link, and so on. Many other topological parameters change very little. Thus the resulting network remains strongly correlated to the master network.

We preferred the use of random graphs to artificial graphs for performance evaluation of protection/restoration schemes. It might be true that there is no proof that such random graphs would resemble real networks better than an artificial graph; however, when using random graphs we have the option of generating not just half a dozen graphs, but hundreds or thousands of different graphs with given network parameters. This approach allows us to desensitize the results toward specific parameters of any given graph; an advantage that does not exist in simulations with artificially generated graphs or families of graphs.

The simplest approach for generating a random graph is based on a binomial probability model, also called *Erdos-Renyi* random graphs [Penr03]:

1. For a given number of nodes N, number the nodes from 1 to N,
2. Assign edges between nodes based on a given probability p.

Basically in the Erdos-Renyi model, the network edges are assigned independently. As a result, this model provides no control over the nodal degrees of the network nodes and does not create a graph that would resemble a real network topology as used in the outside world. In addition to complete lack of nodal degree control, the issue of edge locality has also been ignored in this model.

A widely used random graph generating algorithm that takes edge locality into account was introduced by Waxman [Waxm88]. Waxman's model is similar to Erdos-Renyi model except the probability p of having an edge between a pair of nodes now depends on the distance between the two nodes. Waxman algorithm works in the following manner:

1. An *area* of given length and width is defined.
2. N nodes are randomly dispersed over the defined area. This could be done by choosing randomly a pair values of (X,Y) coordinates for each node. It is possible to add conditions here to maintain a minimum and maximum distance between nodes in the network.

3. Create a *distance* matrix L, where L_{ij} denotes the distance between nodes i and j.
4. Choose an edge between each pair of nodes (i,j) with a probability of:

$$p_{ij} = \alpha e^{-\beta \frac{L_{ij}}{L_{\max}}}. \qquad (2\text{-}2)$$

Where α and β represent two model parameters and L_{max} denotes the maximum distance between any pair of nodes in the network. Waxman algorithm provides a flexible mechanism to include the effect of edge locality in the network, by controlling the parameters $0 \le \alpha, \beta \le 1$. Waxman uses the value of 0.4 for them. $\beta = 0$ reduces Waxman model to the binomial Erdos-Renyi model. The lower the value of α is, the smaller the number of edges in the network will be. Another variant of Waxman model even allows the number of nodes in the network to be statistical; by replacing the fixed number N in Step 2 with the result of a Poisson distribution over the size of the area as defined in Step 1 of the algorithm.

However, significant deficiencies undermine Waxman model in its original form for representation of real networks. First and most important of all, Waxman model does not guarantee network connectivity. Therefore, proper checking must be performed on the generated graph to ensure it is connected; or *k-edge-connected* in studies of protection mechanisms. Furthermore, the model provides little control over the total number of edges in the network, and no control at all on the individual nodal degrees of the network.

Another model has been proposed in [Dunn94] where edge locality is controlled explicitly by adding another input parameter, maximum Euclidean grid length of a span L. In this model, nodes are placed in grid space and the span lengths are specified in grid units. The restriction on maximum span length essentially limits network nodes to connect to their local neighbors within a radius of L. The main problem with this model is that it does not allow a realistic distribution of span lengths. The Waxman model had the advantage that the nodes were more likely to be connected to their near neighbors while the probability of a long edge was controlled by model parameters. But in Dunn's model each edge is equally likely as long as the length of the edge is within the limit. It is claimed in [Dunn94] that this model best represented the published networks of the time when L=1.4.

A more general modeling of today's network topologies was presented in [CaDo97] and [ZeCa97] that reflects the hierarchical nature of modern network infrastructures. In this model three levels of hierarchy have been proposed. Within each layer a number of domains are selected, and locality is applied in the same way as the Waxman model. While this model seems to be flexible and comprehensive enough to capture various layers of Internet topology in general, in our case it would add unnecessary complexity as our focus is on the backbone layer of the Internet only.

2.2.2. Degree- and Distance-Controlled random topology model

In this work we use a model similar to [ScPr04], in which the following characteristics would be taken into account:

- Ability to generate a random graph for a given order (number of nodes) and size (number of links). This would allow us to study trends of restoration performance versus network parameters.
- Ability to set the minimum and maximum nodal degree of the network. This requirement allows us to create more realistic networks. Setting a minimum nodal degree also facilitates creation of networks with a desired connectivity level. The minimum nodal degree must be greater than two; otherwise two-edge connectivity would not be achieved.
- Ability to control edge locality and thus distribution of link lengths.
- Checking k-edge-connectivity for link failure studies.

Our implementation approach is to populate the network to the level that minimum nodal degree is achieved at each node, and then add the remaining edges while checking for maximum nodal degree. An enhanced version of Waxman method is used for assigning edges, thus giving preference to local links while retaining the smaller probability of a few long links in the network, controlled by the parameters of the Waxman model.

A general description of our random graph-generating algorithm follows:

Function topo = random_topo_gen,

Input: N (Number of nodes), M (Number of links), D_{min} (Minimum nodal degree), D_{max} (Maximum nodal degree).

//Initial Setup

1. Check the following conditions to see if the graph is feasible:
   a. D_{min} <= D_{max}
   b. D_{min} >= 2
   c. 2*M/N <= D_{max}
   d. 2*M/N >= D_{min}

2. Define a fixed area length and width (x_{max}=100, y_{max}=100)
3. Assign coordinates to each of the N nodes using a random uniform distribution between [0,x_{max}],[0,y_{max}]
4. Compute Euclidean distance between each pair of nodes.
5. Create NxN matrix Waxman_distance_prob using Equation (2-1).
6. Create boolean local_node_pref=(rand(N)<Waxman_distance_prob), where rand(N)is an NxN matrix of random values between 0 and 1.

// Assign edges to fulfill minimum nodal degree requirement.

For each node k:

7. L=number of additional adjacent links this node needs to achieve minimum nodal degree
8. Pick eligible_neighbors= all remaining network nodes that:
   a. have a nodal degree less than D_{min}
   b. do not already have a link to the node k
9. Choose L neighbor nodes from the vector of eligible neighbors based on the local_node_pref matrix for edge locality preference.
10. If Min nodal degree could not be achieved, start all over.

// Assigning the remaining edges

Until all remaining edges are assigned:

11. Pick at random a source node p where nodal degree < D_{max}
12. Pick eligible_neighbors = all remaining nework nodes that:
    a. have a nodal degree less than D_{min}
    b. do not already have a link to source node p
13. Pick at random an eligible neighbor i for which local_node_pref(i,p)==1. If no such node exist, pick from those for which local_node_pref(i,p)==0. If eligible_neighbors is an empty vector, start all over.
14. Continue the loop until all links have been assigned.

In the above algorithm, Step 1 ensures that the input parameters are correct. For example, the requested average nodal degree is not lower than the minimum nodal degree or higher than the maximum nodal degree. We also check if the minimum assigned nodal degree is higher than two, otherwise a necessary condition for two-edge connectivity would not be achieved. Steps 2-6 allow the implementation of Waxman model for edge locality. In our simulations we used a moderate value for Waxman model parameters, $\alpha=\beta=0.5$ that is also close to values that Waxman used in [Waxm88]. The algorithm then proceeds to 1) fulfill the minimum required nodal degree for every node of the network, and then 2) assign the remaining number of links to achieve the requested average nodal degree of the network.

In order to achieve two-edge connectivity, two possible methods existed. Considering that our purpose is merely to generate a one-time static database of random topologies for future simulations, a most common method used by researchers (see [ScPr04] for example) is to check the connectivity of the generated graphs and discard those that are not two-edge connected (or *k*-edge connected for multiple failure studies). The condition for two-edge connectivity could be tested by removing edges from the graph one at a time and check to see if a spanning tree could still be constructed in the network (graph connectivity test). This is the method we used here to filter the generated graphs. A more time-efficient algorithm was proposed by Tarjan [Tarj74] to find bridges of a graph. If a graph does not have a bridge, then it is two-edge connected. Tarjan's algorithm has a time complexity of $O(M+N)$.

It is also possible to make the graphs *k*-edge connected by modifying Steps 11-14 of our algorithm; for instance using the algorithm in [NaGu97] or using augmentation techniques [EsTa76] instead. However we found our method to be sufficiently capable of generating the required number of *k*-edge connected graphs that we were looking for. Overall, we generated more than 12000 random mesh graphs that were two-edge, three-edge or four-edge connected, with a range of network orders from 10 to 120 nodes, and average nodal degrees of 2.5 to 5. This database of random graphs was used to generate various simulation results in this thesis. Appendix C provides a detailed description of the topological characteristics of our random graphs.

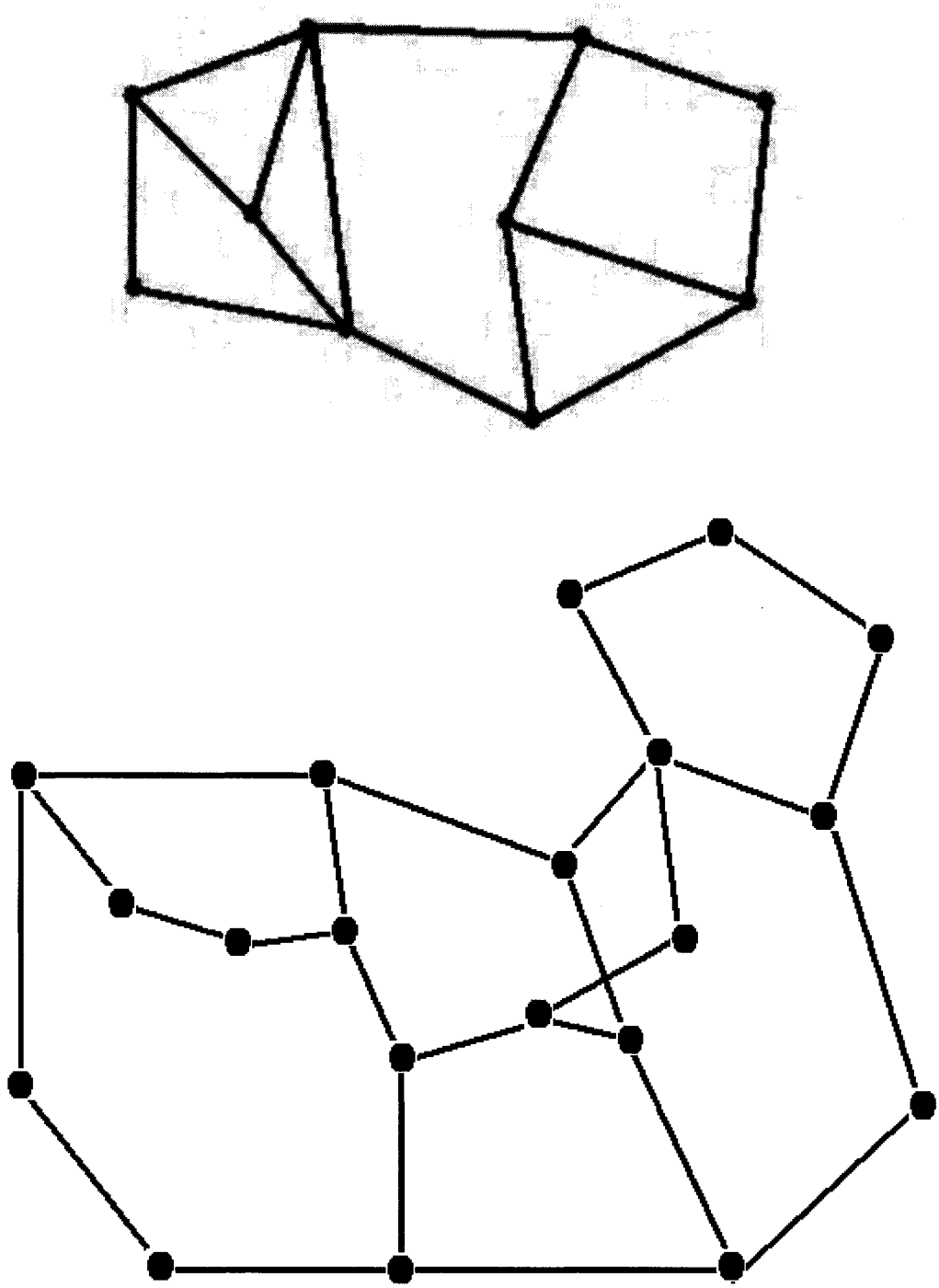


Figure 2-1: Examples of graph connectivities achieved with the degree- and distance-controlled algorithm

Figure 2-1 shows examples of network connectivity achieved using this algorithm. In this case network 1 has 10 nodes and 15 links, and network 2 has 20 nodes and 26 links. Both graphs have a minimum nodal degree of two.

2.2.3. Real Network Models

In addition to random network models, we constructed several sample networks based on real commercial networks that were publicly available. Network topology information for these networks has been provided in Appendix A. Some changes were made in the commercial network topologies to ensure two-edge connectivity requirement. We primarily used these networks for our simulation results with pre-designed networks in Chapter 4, to show how the HP-tree would perform in a network with some characteristics of real commercial networks.

2.3. Demand Models

Random demand models are used to specify the end-to-end connection requests between each pair of network nodes. Some studies consider a dynamic demand model where the demands arrive at and depart nodes at a rate based on Poisson distribution (e.g. see [HoMo04]). Such model could also include connection time lengths for tearing down the connection after a random period of activity. In this case typically an on-off model could be used [Schu04a]. Dynamic demand models are more appropriate for studying demand blocking probabilities. In studying of Restorability and network redundancies, static demand models are often used, a number of them reviewed in [Douc05]. The most common model that was also used in [Douc05] is the uniform random model that assumes connections between pairs of nodes with no bias. An $N \times N$ demand matrix is constructed and populated by random integer numbers between one (or zero if no demand case is allowed) and a maximum demand intensity that indicates the scale of difference in demand between nodes.

Other demand models include *attraction* models that assign higher demand intensities between nodes with larger nodal degrees, and *inverse distance gravity* models that also assign higher demands between nodes with shorter distances [Douc05]. Both models can be deterministic or probabilistic. However the basic assumptions underlying these models are disputable. The attraction model assumes that high demand or high traffic nodes would have a higher nodal degree, which ignores the major transient nodes that form the basic backbone of the network that often have large nodal degrees but may not necessarily be located in major population areas, and thus may not initiate as many

connections as this model suggests. The assumption behind inverse distance gravity could be disputed based on the fact that high volumes of demand often exists between major population centers that are far from each other. The emergence of the Internet as the main source of backbone network traffic also contributes to a more uniformly random demand pattern, as Internet sites and data centers are scattered in domains connected to any of the backbone nodes. Thus, similar to the approach in [Douc05] we will also use a uniform random model to construct the demand matrix in our simulations.

In our study of pre-designed networks in Chapter 4, we also use a semi-dynamic model in which single unit demands are continuously generated between randomly chosen pair of nodes in the network, and the performance of our scheme in a pre-designed network under increasing network load is studied.

2.4. Working and Spare Capacity Model

In our research we considered two scenarios: In the *capacity design* scenario (also called *spare capacity assignment* or SCA problem) in Chapter 3 we assume that working demands are routed between the source and destination nodes according to a weighted shortest path routing algorithm. During the execution of our shortest path routing algorithm, the cost of a link is increased based on the working demand that is routed through it. This approach implements a form of load balancing on the network links. The spare capacity on each link in this scenario is assigned *as required* for maximum restorability with our tree-based scheme.

In the *pre-designed (or pre-capacitated) network* scenario (Chapter 4), we study cases where the total capacity of a link is pre-defined and limited because of the limitations of network equipment already deployed or other factors (discussed already in Section 1.3 and 1.4). Several options exist for modeling of the link capacities in a pre-designed network scenario. In case of real networks the actual normalized values of the total capacity of the link would be used. However there is no generally accepted method to model the total link capacity for random graphs. We use the following models in different simulation scenarios:

1. Homogeneous link capacity model where all links have the same total capacity, and the working demands are routed based on the shortest path up to the total capacity of each link. A similar model has been used in [HoMo94].

2. Homogeneous link load model where the ratio of working-to-total capacity on every link is constant. This model basically requires a balanced routing algorithm that would result in a constant load across the network.
3. Randomly distributed capacity model based on a uniform distribution of link capacities within a given range, or a normal distribution around a mean value. This model assumes that the routing of individual demands in a fixed capacity network based on various parameters could be approximated by randomly distributed link capacities.

Specific details about the usage of each of the above models in our simulations will be provided in Chapter 4.

2.5. Restoration Performance Parameters and Measures

The following are some common parameters and measures used throughout this thesis.

2.5.1. Restorability

Single failure Network Restorability is defined as the percentage of the network working capacity that, given a fixed amount of spare capacity in the network, can be protected against any single link failure in the network. In other words, it is the ratio of the total amount of protected working capacity in the network to the total network working capacity.

Consider a network represented by graph $G(V,E)$ where each link is denoted by e_i. The total working capacity on link e_i is c_{wi}, and the spare capacity on link e_i is c_{pi}. We also assume that the network spans are protected by a predetermined static shared link restoration scheme, which is the subject of our study of protection trees. Backup paths may have common edges.

Let us define the Restoration Paths Matrix, hereafter called the RP matrix, as a binary-valued (or True/False) matrix whose components identify a backup path for each network link. Variants and extensions of this matrix have been widely used in ILP optimizations of mesh restoration schemes (see [Grov94, LiTi05]). It is an $M \times M$ matrix in which each row or column is associated with a network link. For example, row i and column i of the matrix are associated with link e_i of the network. Each element of the RP matrix is defined in the following way:

$$r_{ij} = \begin{cases} 1 & e_j \in D_i \\ 0 & e_j \notin D_i \end{cases}, \quad 1 \le i, j \le M \tag{2-2}$$

The RP matrix could also be used in more generic cases where the working capacity on a span is protected by more than one backup path. In such cases we could divide the span into parallel links in a way that each network link e_i was protected through a backup path D_i independently, thus defining the concept of link as a collection of span working capacity units that use the same backup path. However this is not the case for the protection tree algorithms studied here in which all working capacity units on a span are protected by the same backup path. Note that the RP matrix is computed based on both the network topology and the protection scheme mechanism for choosing backup path. Therefore the RP matrix for a tree-based protection scheme in a given network would be different from the RP matrix for a shortest-path protection scheme or cycle-based scheme in the same network.

We can obtain the value of c_{wpi}, the total protected working capacity on link e_i, and R, the total network restorability, as following:

$$c_{wpi} = Min\left(c_{wi}, \underset{r_{ij}>0}{Min}\{c_{pj} \times r_{ij}\} \right) \tag{2-3}$$

$$R = \frac{\sum_{1}^{M} c_{wpi}}{\sum_{1}^{M} c_{wi}} \tag{2-4}$$

where r_{ij} identifies the (i, j) element of the RP matrix. Considering that the maximum number of non-zero elements in any row in the RP-matrix is *N-1* (because the length of a backup path in a link restoration scheme can never be longer than *N-1*), the computation of Equation (2-4) would have a worst case complexity of $O(MN)$.

The above definition for network restorability is the one commonly used in many references (e.g. [ClGr02, VeGr93, ShYa01]). However some references [StGr00A, Schu03a, DoGr02] use an alternative definition, *Average Link Restorability*, which is computed by averaging the ratio of protected to total working capacities on each network link. The average link restorability is defined as following:

$$R_{avg} = \frac{1}{M} \sum_{i=1}^{M} \frac{c_{wpi}}{c_{wi}} \tag{2-5}$$

The two restorability measures in (2-4) and (2-5) are equivalent as long as either only full restorability ($R = 1$) are studied, or where all links have equal working capacities (the *homogeneous network* scenario [HuCo02]). The average link restorability could be utilized where the path availability is the parameter of interest, while network restorability indicates the amount of restored traffic more accurately. For instance, suppose in a network under a certain partial restoration scenario (such as multiple failures) a working link of 100 Mbps is left unprotected, while another working link of 1 Mbps has been fully restored. Average link restorability would produce a restorability of 50%, while in fact only one Mbps out of the total of 101 Mbps working capacity has been restored. Network restorability would report a restorability of 1/101 in such case.

Based on the above argument, we use the definition of network restorability as defined in (2-4). Any reference to *restorability* in this thesis would mean network restorability.

2.5.2. Redundancy

If c_{pi} is defined as the minimum spare capacity required on network link e_i in order to protect all working capacity units that use link e_i as a segment of their backup path, then network redundancy, defined as the ratio of the total spare capacity in the network to total working capacity of the network, can be computed as following:

$$N_r = \frac{\sum_{i=1}^{M} c_{pi}}{\sum_{i=1}^{M} c_{wi}} = \frac{\sum_{i=1}^{M} \max_{1 \le j \le M} \{c_{wj} \times r_{ji}\}}{\sum_{i=1}^{M} c_{wi}} \tag{2-6}$$

where r_{ji} identifies the (j , i) element of the RP matrix, and c_{wi} is the total amount of working capacity on link e_i. An inverse ratio, i.e. $1/N_r$, which is called Capacity Sharing Factor (CSF) [ShYa04a] is used in this thesis.

Network Redundancy is often used as an important optimization objective in network spare capacity formulation problems, and as a yardstick for comparing the efficiency of protection schemes in reducing the amount of spare capacity required for protection in the network.

2.5.3. Backup Path Length

We define the length of the backup path as the number of hops on the backup paths between the source node and the destination node. The mean backup path length in the network, defined as average value of all backup path lengths in the network, is another parameter that we use to evaluate the performance of protection schemes. The summation of a row in the RP matrix gives the length (in number of hops) of the backup path for that network link. Therefore the average protection path length (in number of hops) in the network, H, is calculated as following:

$$H = \frac{1}{K}\left\{\sum_{i=1}^{M}\sum_{j=1}^{M} r_{ij}\right\} \quad , \quad \text{where} \quad K = \sum_{i}\left(\sum_{j} r_{ij} > 0\right) \tag{2-7}$$

If the link length vector is available, (2-6) can be converted into actual distance instead of number of hops. if d_i is the length of link e_i in miles, the average protection path length in miles would be:

$$H = \frac{1}{K}\left\{\sum_{i=1}^{M}\sum_{j=1}^{M} r_{ij} d_j\right\} \quad , \quad \text{where} \quad K = \sum_{i}\left(\sum_{j} r_{ij} > 0\right) \tag{2-8}$$

In Chapter 3 we also compute the value of maximum backup path length in the network for comparison with hop-limit protection designs.

2.5.4. Sharability

The Sharability of link e_i is defined as the number of backup paths for other links that traverse through this link e_i. Sharability of a single backup wavelength is equal to the sharability of the link on which this wavelength is carried, divided by the number of backup wavelengths on the link [ZhZh04]. This definition assumes equal sharability for all wavelengths on a link. Network Sharability could be defined as the average value of the sharabilities of all network links (or backup wavelengths). A dedicated protection scheme will have a sharability of one. A design parameter, Maximum Allowed Sharability (*MAS*), could be used to place an upper bound on the sharability of each backup wavelength. The total working capacity units that use link e_i in their backup paths could be computed from the Restoration Paths Matrix as follows:

$$S_i = \sum_{j=1}^{N} c_{wj} r_{ji} \tag{2-9}$$

Maximum Allowed Sharability could be used to control how much the protection capacity in the network could be shared by backup paths for different links. Increasing sharability would improve the capacity efficiency in the network, but causes higher vulnerability to multiple failures and dynamic changes, as failure of a link with high sharability would expose a larger number of network links that use that link in their backup paths.

2.5.5. Availability and connectability

The performance measures that we mentioned so far are all deterministic. However, a number of probabilistic parameters have also been introduced in the literature for measuring the reliability of telecommunication networks. *Availability* is defined as the probability that a system will be found in the operating state at a random time in the future [ClGr02], and is widely used as a yardstick for service guarantee in telecommunication networks. It is probably the most relevant performance measure for customers as it indicates the service impact that they should expect.

Connectability is defined as the estimate of the probability that a path can be constructed between a set of nodes at any time in future using the spare capacity of the network [MaGr93]. It could be used to determine the overall efficiency of routing in a reconfigurable network; however, [MaGr93] notes that connectability is a metric of high computational complexity.

In this work we focus on deterministic aspects of network restoration and failure recovery; therefore availability and connectability are not considered in this work. We feel that Restorability is general enough as a performance measure for protection schemes; especially because availability could be easily computed from restorability if the probability of failure for each link is available (see [ClGr02] for example)..

2.6. Modeling of Protection Trees

In this section we define the models for protection trees in a mesh network. We start by reviewing general characteristics of a backup tree, and then describe our Hierarchical-Protection Tree (HP-tree) model and review its advantages and limitations.

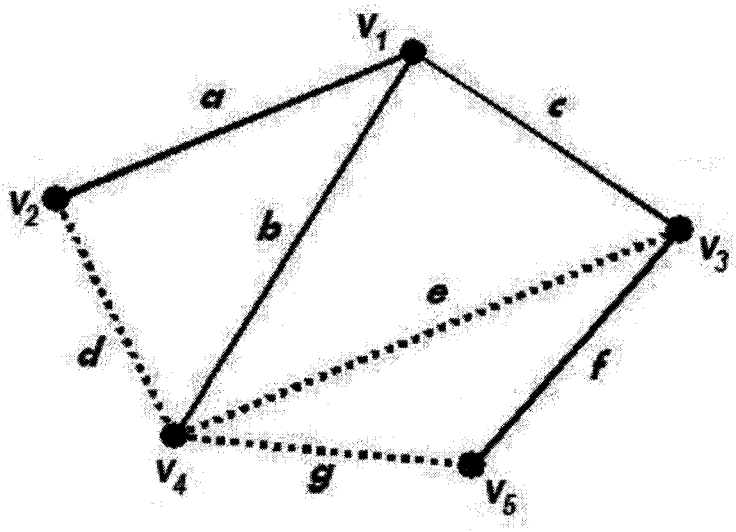


Figure 2-3: Example of a spanning tree in a five-node graph

2.6.1. General characteristics of a backup tree

Suppose we model the network as connected graph G(V,E). A *spanning tree* $T(V, E_T)$ is an acyclic, connected subgraph of G that includes all nodes of the graph (V). It can be shown that an acyclic connected subgraph that connects every node of the graph must have a size of $|V| - 1$ [BuHa89]. Figure 2-2 shows a sample graph with one of its spanning trees, where links *a, b, c* and *f* construct the tree. Note that the number of spanning trees in a network grows rapidly with the number of nodes. The Cayley formula computes the number of spanning trees in a complete graph of N nodes as N^{N-2} [WuCh04], which could also be considered as an upper bound on the number of spanning trees in any graph of order N. Several algorithms for enumerating spanning trees of a network exist (e.g. see [Mats93, ShTa97, WuCh04]), however the time complexity of this task often prohibits practical applications of spanning tree enumeration in telecommunication networks.

The following theorem establishes a basic condition for a spanning tree in Graph G:

Theorem 2-2: Let $T(V_T, E_T)$ be a sub-graph of a connected graph $G(V, E)$. T is a spanning tree of G if the following two conditions hold: (1) $|E_T| = |V| - 1$, (2) For all (u, v) in $E - E_T$, there exists a path P between u and v such that for all link l in P, l is in E_T [TaRu05].

Proof: A proof for this theorem from [TaRu05] is provided in Appendix B.

Suppose we use a backup tree in the network to provide protection capacity for link restoration. Based on the above theorem, the backup tree will provide a backup path for every non-tree link; for instance in Figure 2-1, the tree provides backup paths (*a, b*) for link *d*, (*b, c*) for link *e*, and (*b, c, f*) for link *g*.

However because any tree link failure would disconnect the tree, a static spanning tree cannot protect against the failure of a tree link. A large part of research on protection trees is focused on devising algorithms to deal with tree-link failures and we have already reviewed some approaches in Section 1.2.3. However, here another theorem from [TaRu05] establishes a minimal basis for protection of tree links in a two-edge connected graph:

Theorem 2-3: Given a two-edge connected graph *G* and a spanning tree *T* of *G*, for any tree links there exists a path between the two end nodes of the link that contains exactly one non-tree link [TaRu05].

Proof: The proof for theorem 2.3 could be found in Appendix B.

The above theorem implies that a single link failure on a spanning tree can be remedied by using exactly one non-tree link in the protection path. This result forms the basis for tree-based protection algorithms in general. We use the term *generic shared backup tree* here to refer to a protection tree built based on the above principle. Our HP-tree proposal is a special case of this generic model, where we limit the use of non-tree link in the protection paths to the link adjacent to the failure, thus achieving simpler set up, more robust failure management including ability to handle multiple failures and faster restoration time, as will be described in Section 2.6.2.

2.6.2. The HP-tree model

In our approach to unicast network restoration using trees, we intend to form a shared, pre-planned spanning tree in the network that is rooted at one of the vertices, called the root node and denoted by v_r. A node is considered to be in a higher layer than another node if its path has fewer hops to the root node of the tree. The first hop on the path from each node to the root is called the *primary parent* of that node. Each node is a *child* node of its primary parent. If v_i is a child node of v_j, then we call v_i the *downstream* node of the link (v_i, v_j), and v_j the *upstream* node of the link. A *child subtree* of node v_i consists of this

node plus all of its children and grandchildren and so on; in other words, all nodes that connect to the root node through this node.

Suppose we denote the tree by $T(v_r) \subseteq E$. This subset includes all edges that belong to the tree. We also define T' as the subset that includes all edges that do not belong to the tree. Naturally, $T \cup T' = E$.

We use the subset $P_{ij}=\{(v_i, v_k),..., (v_q,v_j)\}$ to represent the set of edges that form the path between the pair of vertices v_i and v_j through the spanning tree T. As mentioned before there is a unique backup path for each pair of nodes, through T, because the spanning tree T does not contain any loop. Naturally, $\forall v_i, v_j \in V, \ P_{ij} \subseteq T$. The spare capacity available on the tree for link restoration is shared among all nodes.

As proved in theorem 2-3 [TaRu05], the backup path for each tree link has to make use of one non-tree link only. Our tree model mandates that in case of a tree link failure, the backup path should be routed through a non-tree link adjacent to the downstream node. This mechanism amounts to repairing of the disconnected tree in the same fashion as done in the SPT protocol [IEEE04] for routing trees between bridges in the MAC layers. The self-repairing mechanism of the tree provides the backup path again for the failed link; which now stays out of the tree and is treated as a non-tree link. By using a non-tree link adjacent to failure for repairing of the tree, the restoration of the traffic will be faster and with less signalling than the generic backup tree based on theorem 2-3, because the repair of the protection tree is performed in the immediate vicinity of the tree disconnection and therefore fewer nodes would be involved in the signalling. In order to pre-plan this mechanism prior to failures, we assign not just a primary parent node for each node of the network, but also a number of backup parent nodes for each node that are connected to it through non-tree links. The backup parent nodes specify the adjacent links that will be used for self-repairing of the tree if a tree link fails.

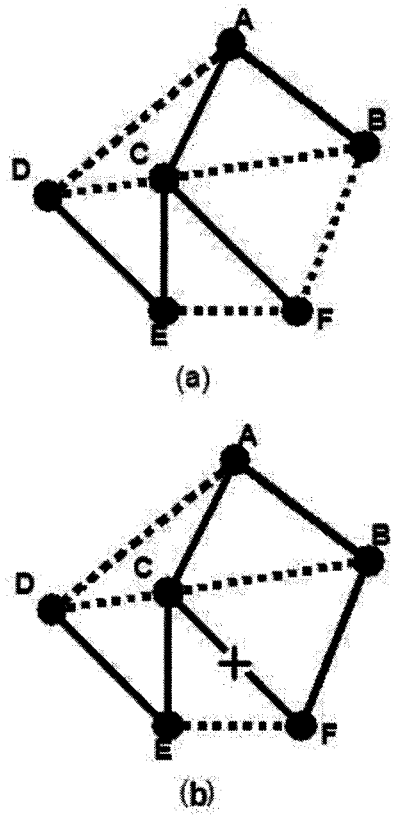


Figure 2-3: Protection of tree links and non-tree links

Figure 2-3 provides a description of this operation. Suppose the thick lines indicate the HP-tree links and the dashed lines indicate the non-tree links in the network. Here the traffic on link EF can be protected via backup path CE-CF, and traffic on the link BF is protected via backup path AB-AC-CF. Now suppose the tree link CF fails. Node F has lost its connection to its primary parent on the protection tree. It now has to switch to its pre-defined backup parent node, for instance node B. The tree has thus reconfigured itself and the traffic on link CF is re-routed through BF-AB-AC. This dynamic reconfiguration also effectively updates the backup path for link EF, now to BF-AB-AC-CE. The network links must be capacitated properly to have enough protection capacity for rerouting of the traffic of those links that they protect. The assignment of primary and backup parent nodes can be done during the design stage so that no routing decision has to be taken upon failure. Note that the root node of the tree is never a downstream node of any link, and as such would not need primary or backup parents.

We call this approach for unicast shared link restoration the *Hierarchical-Protection tree* (HP-tree for short). It provides a self-repairing spanning tree for unicast traffic, and could reconfigure itself quickly with minimal changes in case of dynamic topology changes or multiple links and node failures. The self-repairing shared protection tree described in the above provides a non-static and re-configurable protection for working capacities, however the backup paths do not need to be recomputed for the whole network when a failure hit. If a non-tree link is hit, the backup paths remain unchanged (only the available spare capacity on the tree links will decrease). If a tree link fails, only the downstream node of the link has to re-align itself with the tree in a pre-planned manner, thus only minimal changes are applied locally to the affected node.

On the other hand, there is also a price to pay for this minimal reconfiguration and simplicity of operation, as any limitation on reconfigurations will have a direct impact on the optimization of redundancy in the network. While full reconfiguration of all paths in the network would requires less total protection capacity than cases with partial reconfigurations, however, the full reconfiguration time and complexity of such algorithms may become unacceptable for large networks. We will study the redundancy requirements of our scheme in Chapters 3 and 4 of this thesis.

Note that the non-tree link chosen for this purpose must specifically connect the two subtrees of T that have been disconnected by the link failure; therefore the backup parent of a node cannot be chosen from a node on its child subtree because it would form a loop and not a spanning tree. To understand this point, consider node C in Figure 2-2a. Here node A is the primary parent node of node C, and we have two choices for backup parent nodes: Nodes B and D. But node D is in the subtree of node C (through node E). Suppose we had chosen node D as the backup parent node for node C. In such case if link AC failed and the traffic were switched to the backup parent node through link CD, a loop would have been formed between *CD-DE-EC*. For this reason the only valid backup parent node for node C in this graph is node B. We take this point into account in our HP-tree design algorithm in Chapter 3.

The HP-tree mechanism is also able to handle multiple link failures or node failures in a fairly simple way. As long as there is a local backup parent node available and no network nodes is completely disconnected from the network, the downstream node could

connect itself back to the protection tree. Again in Figure 2-2b, suppose after failure of link *CF* and repairing of the tree through link *BF,* then link *BF* failed too. In this case node *F* will switch to its second available backup parent node, which is node *E.* The tree self-repairs again through link *EF* this time. In order to handle multiple-failure scenario, the network link must be capacitated accordingly to handle the failure of two links. Models for such operation and detailed study of the performance of the scheme under multiple failure scenarios will be presented in Chapter 5, and algorithms and signalling for managing of this operation will be discussed in Chapter 6 of this thesis. Note that we assume the network remains connected after multiple link failures. If multiple link failures disconnect the network, then link restoration mechanism would fail.

2.6.3. HP-Tree Restoration Time

Generally speaking, restoration times in path or link restoration schemes could include some or all of the following delay elements:

t_{det}: Failure Detection Time.

t_{notif}: Total time to notify the source and destination nodes.

t_{SM}: The required time for the system management and control layer at each node to process connection switching messages and to dispatch a message to the optical cross-connect. Could also include the queuing time in the OXC buffer.

t_{sw}: The required time for the optical cross-connect to switch the light path.

t_p: Propagation delay from one node to the next hop.

t_{RT}: The round trip propagation delay for sending the control messages from source to destination, calculated by adding the t_p along the backup path.

Other more detailed delay elements could be included in one of the delay items in the above. The restoration time in a common unreserved path restoration scheme can be computed as following [YaFi04]:

$$t = t_{\text{det}} + t_{notif} + t_{RT} + \sum_{i-1}^{N} (t_{SM}(i) + t_{sw}(i)) \qquad (2\text{-}10)$$

in which N is the total length of the backup path. This computation assumes the following restoration procedure:

1. The failure is detected
2. The source and destination nodes are notified

3. The source sends a control message to the destination node to reserve the capacity along the path.
4. At each node the OXC is instructed to switch the connection, and to proceed to the next node if successful
5. Once the path is successfully established all the way to the destination, the destination responds with a success message and allows the source to switch the traffic.

The restoration time in the HP-tree scheme includes the same delay elements as other pre-planned link a restoration schemes, in which there is no need to notify the source and destination nodes as the connections are re-routed locally around the failed link. Therefore Step 2 of the above process is skipped.

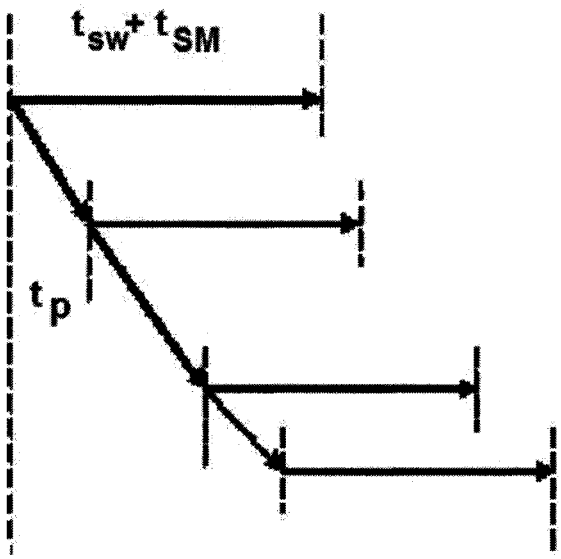


Figure 2-4: HP-Tree Restoration Time

A further enhancement could be proposed [Grov87, YaFi04] to invoke the OXC switching in parallel with sending a control message to the next hop, as illustrated in Figure 2-4. In a pre-planned restoration scheme with guaranteed protection capacity, the traffic can be switched over to the backup path as soon as the OXC switching is completed. The total restoration time would be determined by the longest OXC switching and processing delay in the nodes as following:

$$t = t_{\text{det}} + t_{RT} + \max_{i=1...K}(t_{SM}(i) + t_{sw}(i)) \qquad (2\text{-}11)$$

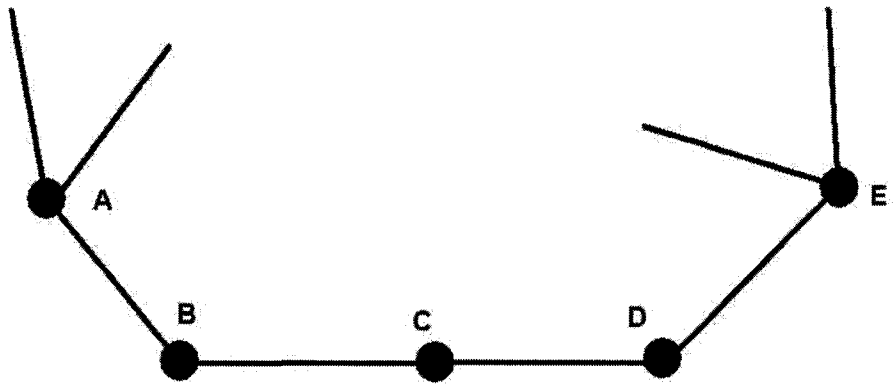


Figure 2-5: Problem with chains in protection trees

in which K is the length of the backup path (in number of hops) between the end nodes of the failed link. In link restoration schemes, the length of this backup path is often shorter than the end-to-end backup path length in Equation (2-11) for path restoration schemes. This factor further contributes to faster switching times for link restoration schemes in comparison with unreserved path protection schemes.

2.6.4. HP-tree Topological Constraint

As noted, Theorem 2-3 indicates that a non-tree link would be sufficient for the backup path of a tree link; however, there is no guarantee that an adjacent non-tree link could be found for this purpose. Furthermore as explained before, in our scheme no adjacent node from a node's child subtree could be used as backup parent node. This would create cases where the HP-tree algorithm might not be able to build a protection tree for full restorability.

This problem often happens when a chain of three nodes or more with nodal degree of two appears in the network. Consider a partial network as depicted in Figure 2-5. Nodes B, C and D each have a nodal degree of two and form a chain. In this case no matter how the primary and backup parents are chosen for the HP-tree, at least one node of those three nodes is left out with no backup parent because the HP-tree scheme is based on using an adjacent non-tree link for repairing the tree in case of a tree link failure.

Therefore when a 3-node chain is present at least one tree link will remain unprotected. We call this limitation a topological constraint of our scheme.

The more ring-like a network is, the lower the HP-tree restorability becomes. In fact for a ring with N nodes, the restorability of our shared protection tree scheme (as described in the above) drops to 1/(N-1). In Mesh networks, however, only one link in each chain of 3 nodes will remain unprotected and thus the total restorability of the network would be related to the number of chains, as we show with simulation results in Chapter 3. There could be a number of different methods to handle the topological constraint of the HP-tree scheme:

- The affected connections could be re-routed from the source node of the chain instead of the node adjacent to the failure, for instance using the concept of meta-mesh [GrDo01]. In such case, the HP-tree protection scheme is applied to the *meta-mesh topology* of the network, which would not have any node with degree of two.
- Alternatively, the unprotected link could be protected by an extra backup path independent of the tree. In this case the downstream node can be tagged to use a pre-planned backup path independent of the HP-tree for this specific link. Spare capacity for this backup path could still be shared with the HP-tree.
- Also the possibility of using non-adjacent non-tree links for backup paths has been studied in [TaRu05]. It requires further designation of children of a node as *primary child* node and *backup child* node, and additional signaling for rerouting of the traffic. However this approach would increase the restoration time, and the resulting tree is no longer self-repairing. It also fails to handle multiple failure scenarios, for which [TaRu05] proposes a best effort restoration class of service. For the above reasons, we stick to our original HP-tree scheme and provide simulation results and statistics to quantify the impact of the topological constraints on the performance.
- In addition to the above, if possible during the topology design phase, such chains could be avoided by adding a single link that would connect the node without backup parent tree to another upstream node in the tree. In case of the network in Figure 2-3, an additional link between nodes A and D eliminates the topological constraint.

Aside from the above approaches, our HP-tree design algorithm should take various factors into account to ensure successful assignment of primary and backup parent nodes

as far as possible. Nodes with nodal degree of two must be placed at tree leaves or root if possible, because an intermediate tree node should have at least three adjacent links: to a child node, a primary parent node and a backup parent node. In addition to topological limitations, the mechanism used by the algorithm to grow the tree from its root node could potentially create conditions where a node would be left with no backup parent node. This limitation could be categorized as algorithmic deficiency. For instance, suppose in Figure 2-3 the link *EF* did not exist in the network. If an HP-tree algorithm created the protection tree from links $\{AC,AB,CE,CF,DE\}$, then node *E* would have been left without a backup parent node. This is not a topological constraint, as no 3-node chain exists in the network. Such algorithmic deficiencies could be avoided by proper HP-tree design; e.g. in Figure 2-2 if link *EF* did not exist, the algorithm should have constructed the tree from links $\{AD, AC, AB, CE, EF\}$. We will explain how we avoid such problems in our design algorithms in Chapters 3 and 4.

2.6.5. Node failure recovery with HP-tree

The self-repairing mechanism in the HP-tree could also handle node failures as long as the root node is not affected and the rest of the network remain connected after a node failure, although the restoration mechanism for the failed traffic may take longer to complete, as we will show here. Note that if the root node fails, the HP-tree cannot repair itself until a new root node is selected. Selection of a new root node can be done through network central management system or in a distributed manner by the nodes. We will revisit this issue in Chapter 6. Generally speaking, node failures can be considered as multiple link failures, because failure of a node essentially disconnects its adjacent links. Obviously the connections that initiated from or terminated at the failed node would be lost in this case; however, those connections that were re-routed through the failed node could be recovered by the downstream nodes of the links adjacent to the failed link.

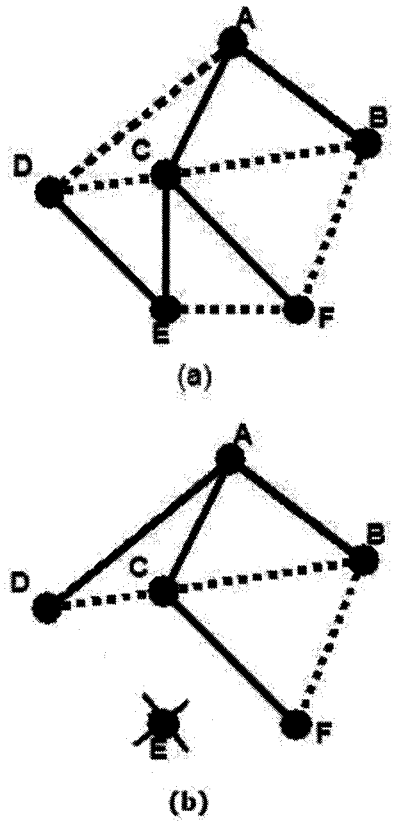


Figure 2-6: HP-tree recovery after node failure

However, re-routing of the failed traffic in this case will be more complicated than a link failure scenario, because one end node of the failed links in this case is no longer accessible through the network, therefore the failed connections cannot be re-routed around the failed links. In such case, the network control plane could fall back to a bronze class path restoration mechanism to restore the traffic, while allowing the HP-tree to repair itself for the next failure. To understand this mechanism, consider the network in Figure 2-6a. Suppose Node *E* fails. As a result, links *EF, CE* and *DE* will be disconnected. Node *D* was the downstream node of link *DE*, and thus it reconnects to the HP-tree through link *DA*, also using Node *C* as it new backup parent node. Node *F* remains connected to its primary parent node *C* and updates its backup parent table by using node *B* as its new backup parent. The HP-tree is now repaired and consists of links *AD, AC, AB* and *CF*. The failed connections would be re-routed from their source node.

Chapter 3. Protection Tree Spare Capacity Design in Mesh Networks

In this chapter we will present algorithms for designing the HP-tree in a given mesh network with known topology and working capacities, and for assigning spare capacity to individual network links for maximum restorability with HP-tree. In this design scenario the input includes the topology and working capacity of the network, and the objective is to construct backup paths for each link and allocate spare capacity for protection as required. Variations of this scenario with limitations on different network parameters will also be discussed.

In Section 3.1 the ILP formulation for a generic shared backup tree (as defined in Section 2.6.1) is reviewed. Then we proceed to present our heuristic algorithms to design HP-trees in a given network. Simulations and analysis of the results are presented subsequently.

3.1. Optimal Capacity Tree Design Problem Formulation

3.1.1. Formulation for on-demand spare capacity assignment

The problem of choosing the tree links in a given network and assigning minimum spare capacities on the links for a generic shared backup tree (as defined in Section 2.6.1) was formulated as an ILP problem in [TaRu05]. We present this formulation to compare the simulation results of our HP-tree heuristics with the optimal shared protection tree case as the performance benchmark. This scenario considers the case where spare capacities can be assigned where needed with no limitation. The input to the problem is the topology information; i.e. the Edge vector E and the working link capacity vector W. The output would be the Tree link vector T and the spare link capacity vector S.

Let $G(E, V)$ represent the input graph where E is the set of edges and V is the vector of network nodes. Also let (i, j) in E denote the bi-directional link between node i and node j. Each bi-directional link (i, j) in E is divided into two arcs, $(i \rightarrow j)$ and $(j \rightarrow i)$. The following parameters are also used in this formulation:

w_{ij}: Working capacity on edge (i, j).

c_{ij}: Link Cost on edge (i, j).

T_{ij}: Tree link indication vector. will be =1 if edge (i, j) is on the tree, =0 otherwise.

s_{ij}: Spare capacity on edge (i, j).

F_{ij}^{mn}: will be =1 if the backup path of the edge (m, n) includes arc $(i \rightarrow j)$, =0 otherwise.

ϕ_{ij}^{mn}: will be =1 if the edge (m, n) is protected by the edge (i, j), =0 otherwise.

δ_{ij}^{mn}: will be =1 if the edge (m, n) is protected by the tree link (i, j), =0 otherwise.

Then the generic backup tree optimization problem can be formulated in the following way:

Problem Formulation

Minimize $$\sum_{(i,j)\in E} c_{ij} s_{ij} \tag{3-1}$$

Subject to:

$$\sum_{(i,j)\in E} T_{ij} = N - 1 \tag{3-2}$$

$$\sum_{i} F_{ij}^{mn} - \sum_{k} F_{jk}^{mn} = \begin{cases} -1 & m = j \\ 1 & n = j \\ 0 & otherwise \end{cases} \qquad \forall (m,n) \in E, \quad j = 1 \cdots N \tag{3-3}$$

$$F_{mn}^{mn} = 0 \qquad \forall (m,n) \in E \tag{3-4}$$

$$\phi_{ij}^{mn} = F_{ij}^{mn} + F_{ji}^{mn} \qquad \forall (m,n) \in E, \quad \forall (i,j) \in E \tag{3-5}$$

$$\sum_{(i,j)\in E} \phi_{ij}^{mn} - \sum_{(i,j)\in E} \delta_{ij}^{mn} = T_{mn} \qquad \forall (m,n) \in E \tag{3-6}$$

$$\delta_{ij}^{mn} \le \phi_{ij}^{mn} \qquad \forall (m,n) \in E \quad \forall (i,j) \in E \tag{3-7}$$

$$\delta_{ij}^{mn} \le T_{ij} \qquad \forall (m,n) \in E, \qquad \forall (i,j) \in E \tag{3-8}$$

$$\phi_{ij}^{mn} + T_{ij} - 1 \le \delta_{ij}^{mn} \qquad \forall (m,n) \in E, \qquad \forall (i,j) \in E \tag{3-9}$$

$$w_{mn} \times \phi_{ij}^{mn} \le s_{ij} \qquad \forall (m,n) \in E, \qquad \forall (i,j) \in E \tag{3-10}$$

The above formulation sets up an optimum spare capacity assignment problem for the generic shared protection tree model in Section 2.6.1. As shown there, this model could provide full restorability in every two-edge connected network as long as sufficient spare capacity is allocated, thus the problem objective (3.1) would be to minimize the allocated spare capacity. Constraints (3.2) and (3.6) in the above implement the requirements of the

Theorems 2.2 and 2.3, i.e. set up an acyclic subgraph of *N-1* links that provides a path between each pair of nodes in the network. Constraint (3.3) defines the F^{mn}_{ij} parameter that specifies the role of each directed arc in a backup path. Constraint (3.4) guarantees that no arc can be part of its own backup path. Constraint (3.5) combines the contribution of the two directed arcs in each link toward the backup path on which this link resides. Constraints (3.7), (3.8) and (3.9) are essentially required by the definition of δ^{mn}_{ij} that specifies the tree links involved in the protection of a given link. Constraint (3.10) ensures that the protection capacity assigned on a backup path is sufficient for protection of the working capacity on the link that is protected by this backup path.

3.1.2. Formulation for Fixed Capacity Scenario

Suppose each network link (*i, j*) has a fixed total capacity. An example is the pre-designed fixed capacity problem that is the subject of Chapter 4 of this thesis. In this case the objective of the formulation should be changed, because both working and spare capacities $\{w_{ij}\}$ and $\{s_{ij}\}$ become the input parameters to the optimization problem. Therefore, the constraint 3-10 would no longer be applicable. The objective of optimization would be to maximize restorability using the available spare capacity of the network. Based on the definition of restorability in Equation 2-4 and by introducing p_{mn} as the amount of protected working capacity on link (m,n), we can write:

Objective:

$$Maximize \sum_{(m,n)\in E} p_{mn} \tag{3-11}$$

Subject to constraints (3.2) to (3.9), and:

$$p_{mn} <= w_{mn} \qquad \forall (m,n) \in E \tag{3-12}$$

$$p_{mn} <= \phi^{mn}_{ij} \times s_{ij} + K(1-\phi^{mn}_{ij}) \qquad \forall (m,n) \in E, \quad \forall (i,j) \in E \tag{3-13}$$

in which K is an arbitrary integer number that must be larger than every link spare capacity in the network. Constraints 3-12 and 3-13 implement the definition of protected working capacity as in Equation 2-3.

The formulations in Section 3.1 were for a generic shared protection tree in which, as discussed in Section 2.6.1, the non-tree link in the backup path did not have to be

adjacent to the failure. In the next section we focus our attention specifically on providing heuristic algorithms for spare capacity assignment in HP-tree design.

3.2. Heuristic Algorithm for Tree Design in the SCA scenario

In the following sections the HP-tree construction algorithm is presented. We first discuss our design objectives in Section 3.2.1. The main body of the heuristic algorithm for tree construction is presented in Section 3.2.2. The correctness and complexity of the algorithm will be examined in Sections 3.2.3 and 3.2.4, respectively.

3.2.1. Design Objectives

We would like to construct a heuristic algorithm that, given a mesh network topology with assigned working capacities on each link, could design a hierarchical protection tree that conforms to the requirements of the HP-tree model in Section 2.6.2. We would like our algorithm to have an acceptable level of time and processing complexity and could construct a capacity efficient HP tree with high restorability.

In designing our algorithm, one important objective is to attempt to avoid any topological constraints that might arise from algorithm deficiencies (see Section 2.6.4) by trying to form the tree in a way that primary and backup parent nodes are available for each node. Also the hierarchical protection scheme would attempt to take advantage of the presence of high capacity core nodes (often high traffic centres of population or major switching nodes) in creating a hierarchical structure in the network, and to reduce the length of the backup paths for high capacity links.

3.2.2. Tree Construction Algorithm

The overall protection tree design algorithm is executed in two steps: First it forms the HP-tree by choosing primary parents for each node, and in the second step backup parent nodes are chosen for network nodes.

The tree construction sub-algorithm in our heuristic design is as following:

HP-tree Construction algorithm
#Choosing the primary parent nodes
1. Create Vector of Nodes V, empty tree node vector U and empty tree edge vector T.
2. Sort V by HPT node placement algorithm (Section 3.2.3)
3. $U(1) = V(1)$ (*Add the root node v_r to the tree vector*)

4. Repeat the following loop of steps until all nodes are added to the tree node vector U

4.1. Going from top to bottom in V, choose the first node v_i that:

- is not a member of tree vector U, and
- is connected to a member of tree vector U

4.2. Create vector L of all members of the sorted vector V that are connected to v_i. L is thus a subset of V and sorted in the same order as V.

4.3. Going from top to bottom in L, choose the first node l_k that has at least two non-tree links adjacent to it, unless:

- l_k is the root node of the tree, or
- no node with the above condition exist in the tree, AND all other remaining (non-tree) nodes have a nodal degree of two.

In each of the above cases, choose the first entry l_k in L.

4.4. If an l_k was found in the previous step:

- add v_i to the tree node vector U.
- designate l_k as primary parent of v_i.
- add (l_j, v_i) to T.

4.5. If reached the bottom of the node vector, reset the pointer to the top of the node vector.

4.6. Return to Step 4.

Some general characteristics of our algorithm are as following:

1. In order to minimize the impact of chains, nodes with nodal degree of two must be placed preferably at the leaves of the tree. The rational is that an intermediate node with one primary parent node and one child node will be left with no adjacent edge to a potential backup parent node. As explained in Section 2.6 about the protection mechanism of the HP-tree, a backup parent of a node could not be in the child subtree of that node.
2. The tree edges are chosen in a manner that each node remains adjacent to at least one non-tree edge, if possible. This is to facilitate the selection of backup parent nodes for protection of tree links.
3. The tree construction algorithm grows the tree by adding nodes to it in an ordered way decided by some pre-determined sorting criteria that we call node placement algorithm. We will examine various node placement criteria based on the network characteristics such as available capacity, nodal degree, Eccentricity or others. This approach has some similarities to the node numbering approach used in a few other

tree-construction schemes e.g. [MeBa99] and [XuTh03], except here we actually sort the nodes based on network parameters instead of merely labelling them with sequential numbers.

As mentioned before, in an HP-tree a link would be protected through its parent node in the tree, but not through its child node. Essentially a node on the HP-tree is responsible for protection of those non-tree links that connect the child subtree of its downstream node to the other parts of the network. Thus, the purpose of the node placement algorithm is to control the position of nodes on the tree in a way to minimize the length of the backup paths on the tree for higher capacity links, and to place more spare capacity higher up the tree so that more links could share it on their backup paths. Various possibilities for node placement criteria will be examined in Section 3.3. The HP-tree algorithm sorts the nodes based on the node placement criteria, and then browses through the sorted vector of nodes to add nodes to the tree one by one. The top node in the sorted vector of nodes would become the root node.

Note that a node cannot be added to the tree if none of its neighbors is in the tree yet. A possible scenario here is a node that its rank in the node placement criteria is higher than all its neighbors. We call such nodes *degraded* nodes. The algorithm skips such nodes in its sweep of the node vector. Once one top-down sweep of the node vector is done, the algorithm resets the node pointer back to the top of the vector to check for degraded nodes. This loop continues until all nodes are added to the tree.

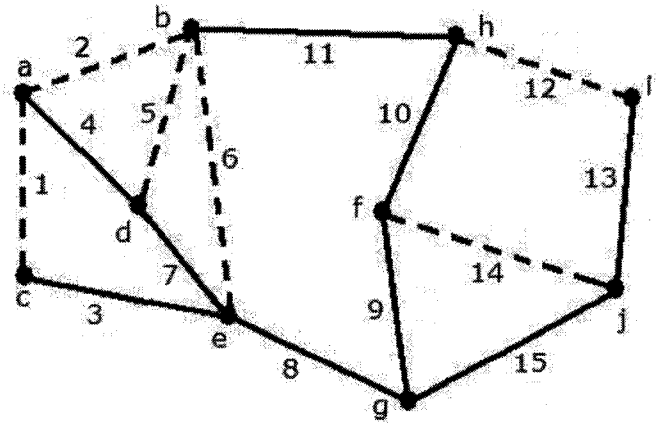


Figure 3-1: An example of HP-tree in a mesh network

Example

An HP-tree built by this algorithm in a sample network is shown in Figure 3-1. A random mesh network was generated with 10 nodes and 15 edges using our random topology generation algorithm in Section 2.2. The thick lines indicate the tree links and the dashed lines indicate the non-tree links. The nodes are tagged with alphabetic letters from *a* to *j*, while the edges are numbered from 1 to 15. The HP-tree has been constructed using a node placement method based on nodal distances, as we will explain in Section 3.3. The vector of nodes was sorted as following: $V=\{g, f, j, e, h, I, c, d, b, a\}$, therefore node *g* acts as the root node of the tree. Note that even though node *b* has a shorter path to *g* (through *e*), but connects to the tree through node *h* instead to allow one non-tree link (*b*-*e*) to be used for backup parent node for node *e*.

We show here that the algorithm for selection of primary parent nodes is able to compute a spanning tree in any connected network. The proof is based on the following theorem from basic graph theory about trees [GrYe99]:

Theorem 3.1: Suppose in the undirected, connected graph $G(V, E)$, subgraph $T(V', E') \subset G$ forms a tree. Suppose an arbitrary node $v \in (V - V')$ that is in G but not in T, and has a neighbor u in T. Then $T'=T+(v, u)$ is also a tree.

Proof: T' is a tree if and only if it is connected and acyclic. Because T is a tree, then there is a path between u and every other node in T. Therefore, v is also connected to every other node of T through u. The rest of the nodes in T' maintain their connectivity from T. Therefore T' is connected. Now T is a tree and therefore acyclic. The new node v cannot have two edge-disjoint paths to any other nodes in T', because it is connected to the rest of the nodes in T' through the single link (v, u). Therefore, no cycle exists in T'. T' is connected and acyclic, and therefore, it is a tree. **QED**

Now let us use the above theorem to show that our tree design algorithm does indeed create a spanning tree. Our algorithm starts from a single node tree (root node), and adds other nodes to this tree one by one. Therefore based on the above theorem, it is creating a new tree in each step. Once the algorithm ends successfully by adding all nodes, the resulting subgraph would be a tree and would include all nodes of the network; i.e. a spanning tree. Therefore we showed that our algorithm constructs a spanning tree.

Now we must show that if the graph is connected, the algorithm always ends successfully. The algorithm scans through the vector of nodes and connects those nodes to the tree that have an adjacent link to the tree. Then it restarts at the top of the vector until all nodes are added. Is it possible to have a node x that could not be added to the tree? That would mean that node x has no adjacent link to the rest of the nodes that, by the end of the algorithm, have all been added to the tree. Therefore, node x must be a disconnected component of graph G, which contradicts the original assumption that G was connected. The above argument can be generalized for the case of an isolated cluster of nodes, instead of single node x. Therefore in a connected graph G, our algorithm is always able to add all nodes to the subgraph, which according to the theorem 3.1 will be a spanning tree.

3.2.3. Tree node placement criteria

The tree node placement criteria are used to sort network nodes in the order that they will be added to the tree in the HP-tree design algorithm. The top node on the list would become the root of the tree. In general the nodes can be sorted arbitrarily, randomly, or based on topological parameters. We focus our study on topology-based sorting of the nodes and show later that its performance surpasses the random case.

We consider four different criteria here: Two capacity-based methods (Average Adjacent Capacity (AAC) and Maximum Adjacent Capacity (MAC)), and two topological methods (Nodal degree (ND), and Distance from Center (DIST)). The nodal degree or Distance from Center criteria would attempt to create a tree with shorter backup paths and more capacity sharing; which could be more efficient for spare capacity design scenario that we study in this chapter. The capacity-based methods would attempt to improve the restorability and reduce redundancy by placing the large capacity nodes higher up in the tree hierarchy. These methods are more suitable for the pre-defined network scenario that we will study in Chapter 4, in which the protection capacities are fixed and known in advance and our objective would be to maximize restorability. A detailed discussion about the pre-designed scenario will be presented in Chapter 4.

As explained in Section 2.6.2, a node with nodal degree of two must be put preferably in the leaves of the tree. Therefore in the AAC and MAC methods we check the nodal degree of the nodes while sorting, and if the nodal degree of one node is two and the

other one higher than two, the latter will be put higher up in the HP-tree. The placement methods based on nodal degree already satisfy this condition.

In discussing the complexity of the following node placement criteria, we assume that a typical sorting algorithm with maximum complexity of $O(N \log N)$ is used [Knut97] where applicable.

Nodal Degree (ND)

One method for sorting the nodes in the HP-tree is to use the nodal degree of the nodes as criteria. In this way, the HP-tree becomes wider rather than deeper, and restoration paths on the tree will be shorter. The topological constraint of the HP-trees is addressed in this method by putting the nodes with larger nodal degrees in the higher layers of the HP-tree, thus giving more options to the lower layer nodes for primary and backup parent nodes. The ND method is more appropriate for network design problem where spare capacities can be allocated as needed. But in a pre-designed network scenario with fixed spare capacity (as the scenario in Chapter 4) restorability may suffer, because the protection capacity of a node is not taken into account in ordering of the nodes.

The ND method requires computation of nodal degrees for each node and then sorting the vector of nodes accordingly, which would require a total complexity of $O(N*D + N \log N) = O(M+N \log N)$.

Distance from Center (DIST)

Another option is to try to minimize the depth of the tree (and thus the length of the backup paths) by sorting nodes in reverse order based on their eccentricity, which is equivalent to choosing a root node at the *center* of graph and then building a breadth-first tree around it. As defined in Section 2.1, the center of the graph is a node whose eccentricity is equal to the radius of the network. If there are several nodes at the center with the same eccentricity (a center-path), one of the nodes on the center-path is selected arbitrarily.

This approach efficiently eliminates the degraded node problem in any network, as nodes are added based on their distance from the center and thus each node will already have at least one neighbor in the tree. Therefore the vector of nodes is scanned exactly once and the complexity of degraded nodes is eliminated. It also creates shorter backup

paths by placing the root node at the center of the network and using the distance parameter to minimize the length of the tree paths. Nodal eccentricities could be computed from the node distance matrix, which itself can be built with a time complexity of $O(MN)$ [BuHa89]. Efficient algorithms with complexity of $O(N^2)$ also exist for dynamic computation of tree root when network condition changes [RaRa96].

Average Adjacent Capacity (AAC)

AAC is defined as the mean value of link capacity on the links connected to each node. In the design scenario in this chapter, we use the average working capacity on incident links and compute for node v_i :

$$S_i = \frac{\sum_{j \in E_i} c_{wj}}{d_i} \tag{3-14}$$

where Ei is the subset of edges incident to node v_i .

Clearly, node placement based on AAC criteria favours nodes that have no low capacity link adjacent to them. For a pre-designed network it provides a fairly good chance to ensure existence of both primary and backup parent nodes, thus avoiding un-restorable edges, by pushing up nodes that have a higher average capacity rather than those with one very high capacity link and several low capacity ones.

The AAC node placement algorithm requires a further comparison of nodal degrees to ensure that those with nodal degree of two would be placed either at root or leaf nodes, if possible. However because the computation of average adjacent capacity has the same order of complexity as nodal degree, the total computational complexity of the AAC node placement algorithm will also be $O(M+N \log N)$.

Maximum Adjacent Capacity (MAC)

The MAC criterion arranges nodes according to the maximum link capacity of their adjacent links. If the highest adjacent link capacity for both nodes is the same, the second highest capacity links are compared, and so on.

This method is also more suitable for pre-designed network scenario in Chapter 4. As opposed to AAC criteria, MAC pushes up nodes that may have only one high capacity

link and many low capacity links connected to them. In doing so, we expect a decrease in the number of degraded nodes, because a high rank node (which in the MAC method would be a node with an adjacent high capacity link) in this case would always be at least adjacent to another high rank node (the other end node of its adjacent high capacity link). As such, a node would be degraded if its high capacity neighbor is degraded, which is less likely than the normal case described earlier. In MAC too we give priority to a node with nodal degree higher than two over a node with nodal degree of two.

While the MAC method reduces the number of degraded nodes, it increases the complexity of the node placement algorithm, because the maximum capacity adjacent link at each node must be found from a list of eligible link candidates. With a sorting algorithm, the complexity of the MAC node placement criterion is $O(N D \log D + N \log N) = O(M \log D + N \log N)$.

3.2.4. Backup parent selection

After the selection of the tree edges that identify the primary parent nodes of each node, the last step in construction of the HP-tree is to determine which non-tree edges will participate in protection of tree links; i.e. the selection of backup parent nodes for each node. As mentioned in Section 2.6.2, in our approach we would like to minimize the amount of signaling and the restoration time by mandating that the failure of a tree link must be restored through an adjacent non-tree link of its downstream node, i.e. a path to the *backup parent* node of the downstream node. The question of how to choose the backup parent node from the neighbors of a node depends on the protection scenario. We present here a method based on the concept of triangular subtrees for the selection of backup parent nodes in the spare capacity design scenario. Another method will be presented in Chapter 4 for the pre-designed, fixed capacity scenario.

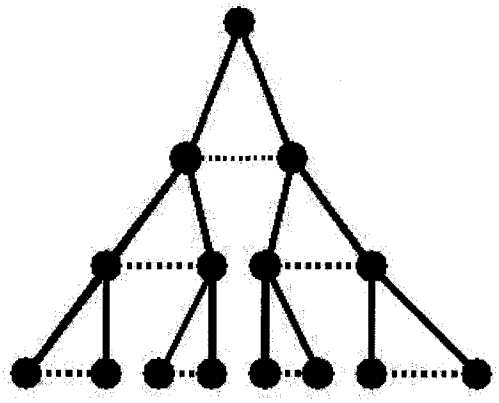


Figure 3-2: Triangular Binary Tree

Triangular subtrees for backup parent selection

Triangular subtrees have been used in the past in tree growing algorithms for reducing the number of signaling messages for tree construction [Zhan02a]. We will show here that assigning backup parent nodes based on triangular subtrees would minimize the number of non-tree links that participate in protection process and thus require allocating spare capacity on them. As discussed in Section 2.6, spare capacity is only required on the tree links of the HP-tree as well as those non-tree links that connect nodes to their backup parent nodes. Other network links do not participate in traffic protection and thus do not require spare capacities.

Let us consider the hypothetical network in Figure 3-2, which is composed of inter-connected triangles. We call this ideal case a *complete triangular tree.* The protection tree links are illustrated by solid thick lines, and backup parent links are shown in dashed lines. Other links might also exist between nodes but are not shown as they are not part of any protection path. Each node has two children that are siblings, and acts as primary parents for those nodes. The siblings act as backup parent nodes for each other. The tree links that connect each child node with its primary parent node are called the triangular links. The backup parent link that connects the two siblings is called the triangular base.

The following simple lemma is the basis for our backup parent assignment in the HP-tree:

Lemma: Among all HP-trees in a network, a triangular protection tree has the minimum number of backup parent links.

Proof: In a general case the *N-1* nodes in the network (with the exception of the root node that does not need any backup parent because it is not the downstream node of any link) will require a maximum of *N-1* links to connect to their backup parent nodes. In the complete triangular tree in Figure 3-4, each backup parent link is shared by two nodes; each one of them uses it to connect to the other one as its backup parent node. The total number of backup parent links is, therefore, *(N-1)/2* in this case. Because a link can at most be shared by two nodes (its end nodes), it is impossible to go beyond this *(N-1)/2* limit in any hierarchical protection tree. Therefore, the triangular tree achieves the minimum number of backup parent links.

The above argument indicates that in order to minimize the protection capacity in the network in an HP-tree, it would be preferable to attempt to form as many triangles as possible on the protection tree. Note that in a triangular tree, the protection paths lengths are also likely to be minimized. Three cases exist:

1. The tree links are protected through the backup parent node, which is the nearest sibling of the downstream node. Therefore the protection path for the tree links will have the minimum length.
2. The backup parents are the minimum distance neighbors on the tree and thus the backup path for the backup non-tree links will have the minimum length.
3. Other links between the nodes are protected through the tree, and the protection path length for them is on average similar to any other hierarchical protection tree.

Based on the above arguments, among the *M* links and *N* nodes of a network, we have *(N-1)* tree links and *(N-1)/2* triangular bases that can be protected by minimum length protection paths, and the rest of the network links have an average protection path length equal to those provided by any other hierarchical protection tree of the same size.

Our backup parent selection algorithm has been summarized in the following:

HP-tree backup parent algorithm

```
1. For each node x without backup parent, do:
1.1.  Sort all neighbours based on their tree distances
1.2.  Exclude neighbours that are in the subtree of x.
1.3.  Choose the neighbour that:
        • Its connecting edge is not in the HP-tree
        • Does not have a backup parent node
```

- Is not the root

1.4. If such neighbour is found, assign it as backup parent of node x, and assign node x as backup parent of this neighbour.

1.5. If not found, choose the neighbour with minimum distance that its connecting edge is not in the HP-tree. Assign it as the backup parent node of node x.

1.6. End of the loop.

Once the primary parents are selected, we select the backup parent nodes from the *Minimum Tree Distance neighbours*. In a spanning tree such as an HP-tree, we can define the *tree distance* between any pair of nodes as the number of hops on the path from one node over the spanning tree to the other node. For any node, the closest neighbour on the tree (minimum tree distance neighbour) is its parent node or its immediate child nodes, all having a tree distance of one hop from this node. But the backup parent must be different from the primary parent node, and a child node cannot serve as backup parent node. For each network in the network except the root node (that does not have a backup parent), our algorithm first eliminates the parent node and child nodes from the list of neighbors, and then sorts the remaining nodes based on their tree distance from the examined node to pick the nearest one as backup parent node. This is the reason we took special consideration in the HP-tree construction algorithm in Section 3.2 to leave at least one non-tree upstream link at each node in order to maximize the chance of finding eligible backup parent nodes at this stage.

In addition to the above, our algorithm also takes into account the benefit of forming triangles on the tree, by having a pair of nodes use each other as backup parent nodes. The ideal case is presented in Figure 3-3 where every node chooses its sibling for this purpose. In our algorithm when we are examining the eligible neighbours, we first consider those that do not already have a backup parent themselves, and assign the two nodes as each other's backup parent. For instance if we are checking node X with neighbors {A, B, C, D} sorted based on their tree distance, if only node A already has a backup parent itself, we assign node B as the backup parent of node X and node X as the backup parent of node B, thus creating a multi-node triangle between node X, node B and the HP-tree. Note that nodes X and B may not be immediate siblings but cousins instead, therefore the minimum tree distance may be larger than two if no neighbor sibling exists.

If nodes X and B were immediate siblings, then the triangle would reduce to the minimum form of Figure 3-4.

If all eligible neighbors of node X have already been assigned backup parents, we choose the node with the minimum tree distance. For instance in the previous example if vector {A, B, C, D} identifies all eligible neighbours sorted base don their tree distance from node X, and if all nodes in this vector already have backup parents, then node A would serve as the backup parent of node X. The reverse assignment would not be applied in this case, thus the triangle would not be formed in this case. But because we chose the backup parent with minimum tree distance, the total protection capacity would be minimized.

The tree distances could be computed in advance and stored in a matrix, as their numbers do not dynamically change often. An example of a tree distance algorithm computation is presented below. A general graph distance matrix can be computed in $o(NM)$ time using a BFS tree at each node [BuHa89]. Because the number of edges on a spanning tree is equal to total number of nodes $N-1$, therefore the time complexity for computing tree distance matrix would be $o(N^2)$.

HP-tree tree distance matrix algorithm

```
1. For each node i
1.1.  For each node j where distance(i,j) is unknown
1.1.1.  Set the distance(i,j) to zero
1.1.2.  Start from node i
1.1.3.  Do:
            • Move up one hop on the HP-tree
            • Increment distance(i,j)
1.1.4.  Until reach the common grand parent of nodes i and j
1.1.5.  Do:
            • Move down one hop on the HP-tree toward node j
            • Increment distance(i,j)
1.1.6.  Until reach node j
1.1.7.  End of loop 1
1.2.   End of loop 2
```

Now let us present a comparative example to illustrate the result of the tree construction algorithm based on different node placement criteria.

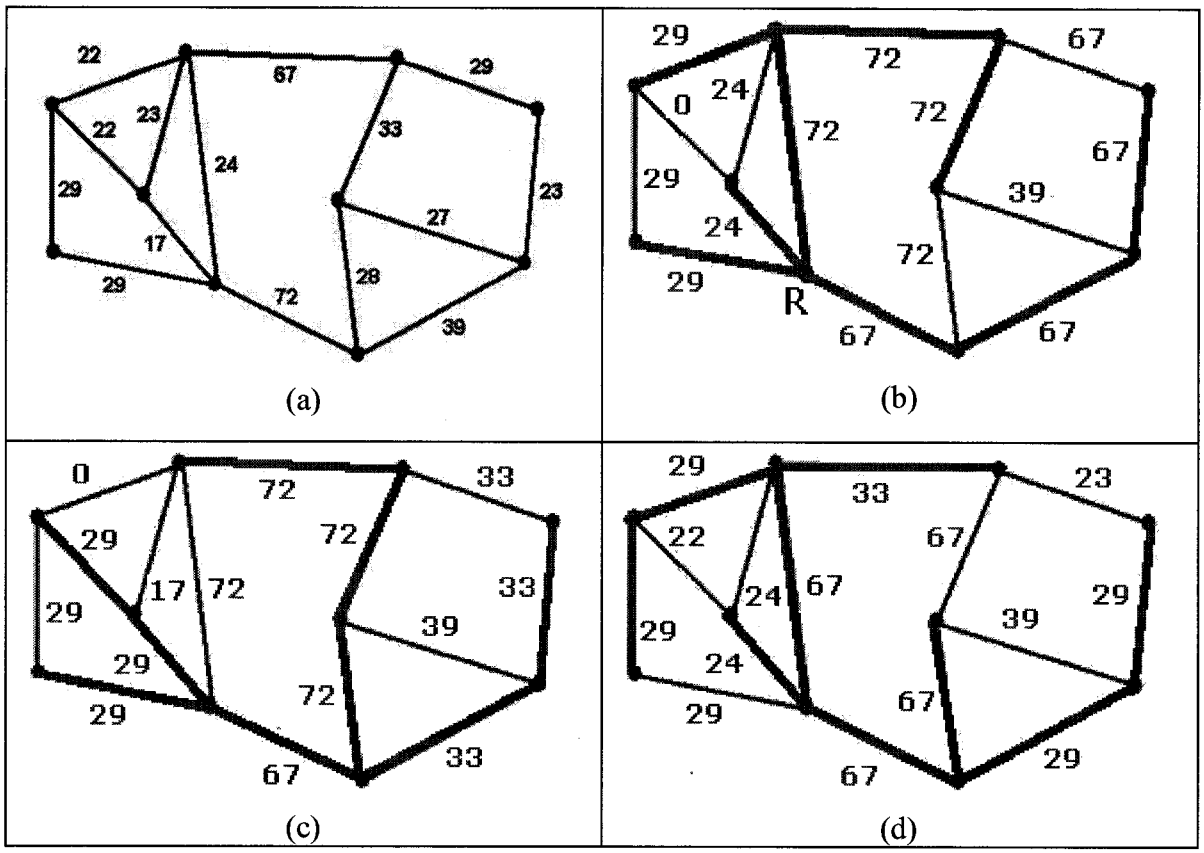


Figure 3-3: a) Sample mesh network, b) HP-tree using ND method, c) HP-tree using DIST method, d) HP-tree using AAC and MAC methods.

3.2.5. Comparative Example

Figure 3-3 gives examples of HP-trees built by using each of the node placement criteria. The same random network topology as in Figure 3-1 was used here, as shown in Figure 3-3a. A bi-directional, uniformly distributed random demand of size [0 10] was generated between each pair of nodes, and routed using a weighted shortest path algorithm as described in Chapter 2. The resulting working capacity units on each edge have been shown in the figure 3-3a next to each edge. Then the HP-tree algorithm of Section 3.2 with various node placement criteria was applied. The resulting trees are shown in Figure 3.3b (for ND method), 3-3c (for DIST method), and 3-3d (for AAC and MAC methods). The numbers next to each link in Figures 3-3b to 3-3d indicate the allocated spare capacity, based on the formula that we will describe in Section 3-4. The HP-trees shown in Figure 3-3b and 3-3c can provide 100% restorability to the network if appropriate amount of spare capacity is allocated, while the HP-tree in Figure 3-3d could provide 85% restorability. On the other hand, when we generated one randomly built

spanning HP-tree in this network, it could only protect 72% of the working capacity. This result indicated the advantages provided by using appropriate node placement criteria in our algorithm as opposed to adding the nodes randomly to the tree, particularly as the node placement method better avoids the topological constraints of a protection tree.

Each HP-tree in Figure 3-3 indicates the unique focus of the node placement criteria used in that method. The tree built by the ND method chooses the node with the largest nodal degree as the root node, while the tree built by the DIST method chooses the centre node of the network. As this example shows, the tree built by the DIST method has the minimum depth comparing to the other trees. HP-trees built by the AAC and the MAC methods in this example turned out to be similar, even though the ranking of the nodes in the node vector were different for the two methods. In small networks such as the one in Figure 3-3, it is not unusual to have various methods computing the same tree.

Note that the redundancy for the network is about 151% in Figure 3-3b, 129% in Figure 3-3c, and about 119% in Figure 3-3d. However because the HP-tree in Figure 3-3d does not provide 100% restorability, its redundancy results cannot be directly compared with others. We will expand on this point in Section 3.5.4. In comparison, the following results were computed by Professor Wayne Grover (TRLabs) for the network in Figure 3-3 using (more complex) optimal mesh and p-cycle designs:

- Redundancy for an optimal mesh design with backup paths computed for individual capacity units and no hop limit, was about 98%.
- Redundancy for a p-cycle design with five cycles and no hop limit, was about 103%.
- Redundancy for an optimal mesh design with backup paths computed for whole spans and no hop limit, was about 117%.

As mentioned in Chapter 1, we expect to pay a price in redundancy for the benefit of simplicity and lower computational complexity of the HP-tree. In the above example, the additional redundancy cost was about 10%-25% comparing to the optimal mesh and p-cycle designs. This result is consistent with further comparative results that will be presented in Section 3.5.4.

3.3. Algorithm complexity

Let us first consider the tree construction algorithm in Section 3.2.2. The first part of the algorithm sorts the network nodes according to node placement criteria, whose computational complexity were examined in Section 3.2.3 for various cases. The rest of the tree construction algorithm includes traversing of the node vector; and at each point the algorithm conducts an examination of the adjacent links of each node in order to select the best primary parent nodes. Therefore the time complexity of computations at each node would be $O(D)$, with D being the nodal degree of that node. A run through the vector of nodes would require time complexity of $O(ND)=O(M)$.

The above computation does not include the processing time for determining whether a node is degraded or not. In such a case, the algorithm might have to browse through the vector of nodes several times. However degraded nodes are processed in these sweeps as long as they have no neighbor node in the tree. Therefore, the only computation regarding the degraded nodes is to check to see if any of its neighbors is already in the tree. This can be done with a maximum complexity of $O(D)$ for each node.

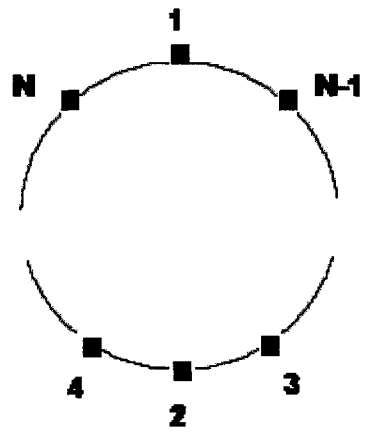


Figure 3-4: Worst-case number of degraded nodes

The worst-case scenario for degraded nodes is a case in which a given node placement criteria would arrange the vector of N nodes as shown in Figure 3-4. The node numbers indicate the placement of each node in the vector of nodes. In this case only two nodes can be added in each sweep, so $\left\lceil \frac{N}{2} \right\rceil$ sweeps of the vector of nodes are required to add all nodes to the tree. The first sweep would add two nodes to the root node and skip N-3

degraded nodes (the other tree nodes are the root node plus its two neighbor nodes that have just been added to the tree); the next sweep will add two more nodes and skip *N-5* nodes and so on until all nodes are added. The complexity in such worst case scenario is $O(NM)$. However such case could be avoided by choosing appropriate node placement criteria that would eliminate or reduce degraded nodes in such networks. In the above ring example, a node placement algorithm based on nodal degree would completely eliminate the problem of degraded nodes.

The complexity of the backup parent selection sub-algorithm is determined by:

- Construction of the Tree distance matrix, $O(N^2)$.
- Exclusion of backup nodes from the child subtree, which can be done on average in $O(D)$ for each node and $O(M)$ in total.
- Selection of the backup parent from the remaining list, $O(D)$ for each node.

Based on the above, we expect the complexity of the backup parent sub-algorithm in the pre-designed case to be $O(N^2+M)$.

A simulation study of the time complexity of the algorithm will be presented in Section 3.5.2.

3.4. Spare Capacity Assignment

Once the HP-tree is computed, the required spare capacity on each link can be computed using the following formula:

$$c_{pi} = \max_{1 \le j \le M} \{c_{wj} \times r_{ji}\} \tag{3-15}$$

in which c_{wj} is the amount of working capacity on link e_j and r_{ji} is the respective element of the *RP* matrix. In our implementation of the spare capacity assignment scenario, we first constructed the HP-tree using the algorithms in Sections 3.2.2 to 3.2.4, then computed the *RP* matrix for the network, and finally computed the spare capacities using the Equation (3-15). The computation of the *RP* matrix and the computation of Equation (3-15) both have a worst case complexity of $O(NM)$, because each row of the *RP* matrix (corresponding to the number of segments on a backup path) could only have *N* non-zero elements. Our results in Section 3.5 will indicate that the average number of hops in a backup path on the HP-tree increases with *log* (*N*); therefore, the average complexity of this computation would be closer to $O(M \log N)$.

Thus all pieces of our HP-tree design algorithm for spare capacity assignment scenario are complete. Now we will proceed to examine the performance of the HP-tree design algorithm using simulation of random and real network in different scenarios.

3.5. Performance Evaluation Scenarios

In this section we provide details of our computation methodology, and compare the algorithm execution time for different node placement criteria. Then we use simulations to evaluate the performance in terms of maximum restorability, network redundancy, and mean backup path length as a function of various network topological parameters such as the network size, the number of nodes and the average nodal degree.

3.5.1. Methodology

The general simulation models for this research were discussed in Chapter 2. We have generated a database of more than 12000 mesh graphs with varying parameters (network size, number of nodes, average nodal degrees and minimum nodal degrees) using the algorithm in Section 2.2.2. The generated graphs were at least two-edge connected. Detailed information about the topological characteristics of the generated graphs is presented in Appendix C. We also collected additional topological information about each graph, such as radius, diameter and eccentricity, as well as the number of 3-node chains in the graph for the purpose of studying the impacts of topology constraints as discussed in Section 2.6.2. About three quarters of random graphs in our set did not have any 3-node chain, while one quarter had one chain or more. About 5% had more than eight 3-node chains.

We generated random bi-directional demand matrices for each graph. The size of demand between each pair of nodes was uniformly distributed between zero and ten units between each pair of nodes. At each node, demand was routed from source to destination based on the weighted shortest path algorithm described in Section 2.4. The total working capacity on each link was equal to the sum of the demand units routed through that link. For the purpose of comparison, in some graphs we have also provided the results for a *homogeneous link capacity* scenario, where we assumed the working capacities on all network links would be equal. The intention of this comparison is to assess the impact of the variation of link capacities on the performance of the HP-tree scheme.

For each graph, we applied our HP-tree design algorithm to construct the HP-tree in the network using the four suggested node placement methods. In design of the HP-tree and performance computation for each network we used the placement method that provided the best restorability for that specific network, unless comparative performance results of the node placement methods were discussed (such as Figures 3-5, 3-14, 3-15, 3-16 and Tables 3-2 and 3-5) in which case the results were collected separately for each placement method in each network. Therefore the following steps were taken for each graph:

- A bidirectional random demand matrix was generated.
- The random demands were routed through the network using the weighted shortest path method and the total working capacity on each link was computed.
- The HP-tree design algorithm was applied using each of the four node placement methods (AAC, ND, MAC, DIST).
- The RP matrix was computed for each design.
- The restorability was computed using Equation (2-4).
- The design that provided the highest restorability among the four node placement methods was selected for this graph.
- The rest of the performance parameters (redundancy, average backup path length etc) were computed for the selected graph.

The most time consuming part of the above process was the initial computation of the working capacities based on the demand matrix to create the input network to our tree design algorithm. Based on the calculations in Sections 3.2 to 3.4, the process of tree design algorithm, selection of the best node placement method and assignment of spare capacities would have a worst case complexity of $O(NM)$ altogether.

In most of the graphs we have presented curves of average results from our sample networks. The 95% confidence interval for the results was computed using the student-t distribution. Because our sample size for each graph point was often larger than 100, the confidence interval was simplified to $\pm 1.96\,\sigma/\sqrt{N-1}$, in which N is the number of sample networks and σ is the standard deviation of the results.

In this chapter we are studying the spare capacity design scenario. Therefore in our simulations the spare capacity on each link is equal to the minimum amount required to achieve maximum restorability using the HP-tree scheme for that given network.

3.5.2. Algorithm Execution Time

We recorded the execution time of various stages of our HP-tree design algorithm for the set of random mesh network graphs that we had generated beforehand. The following computing environment was used for the simulations in this chapter:

1. Various functions of the HP-tree design Algorithm were implemented as M-functions and M-scripts in MathWorks' MATLAB® version 7.0 modeling environment.
2. The simulations were performed on a Personal computer with a 3-Ghz Pentium CPU and 1-Gig RAM, running Windows 2000.

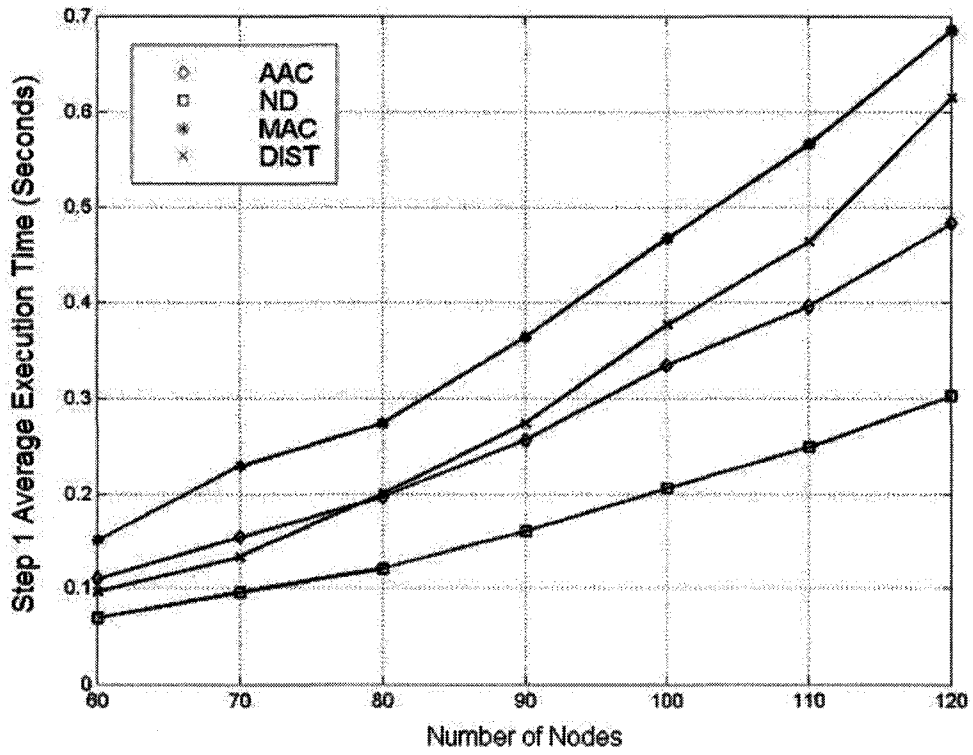


Figure 3-5: HP-tree Design Step 1 Execution Time (seconds)

The execution times for the main two steps of the algorithm were recorded separately. These two steps were:

1. Step 1 (Section 3.2 and 3.3) to form the HP-tree using the node placement criteria and select the primary parents of nodes.
2. Step 2 (Section 3.4) to choose the backup parents for each node.

The execution times were measured for network sizes from 60 to 120 nodes. For networks smaller than 60 nodes the algorithm execution times were too small to be measured accurately.

Figure 3-5 shows the average execution time of the Step 1 of the algorithm. Each point on a curve is the average value for all networks with the same number of nodes. The points on the curves were connected by straight lines. As expected from the discussion in Section 3.3 regarding the complexity of node placement algorithm, the time complexity varies among various methods. The ND method, as expected, provided the fastest placement mechanism while MAC had the longest execution time. As the slopes of the curves indicate, the DIST method had a larger order of complexity than other methods.

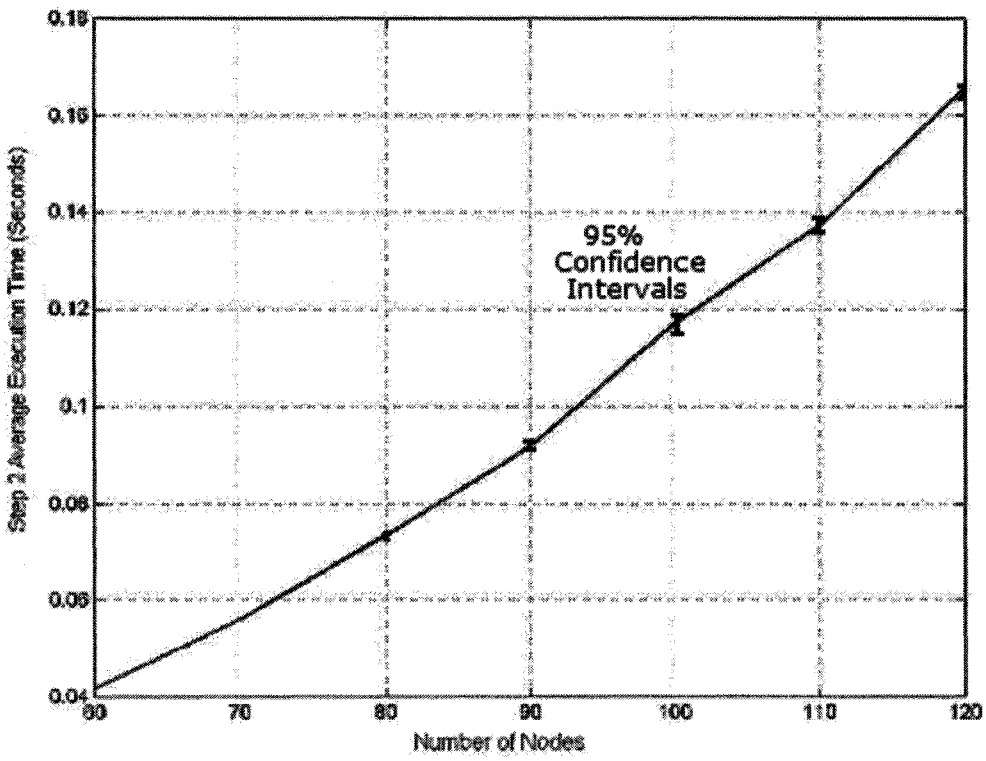


Figure 3-6: HP-tree Design Step 2 Execution Time (seconds)

The Step 2 of the algorithm (Section 3.4) to choose the backup parent nodes (and thus, the non-tree links that participate in protection paths) is common to all various node

placement criteria. Figure 3.6 shows the execution times versus number of nodes. The curve in Figure 3.6 indicates an $O(N^2)$ average time complexity for our implementation of the backup parent selection step. The 95% confidence interval at each point on Figures 3.5 and 3.6 was within ±1.5% of the average value at that point, which was shown on Figure 3-6 but it was too small for visual measurement. For this reason, in this thesis the 95% confidence interval numbers are generally provided in the text instead of being shown visually on the curves.

The total execution time (Step 1 + Step 2) indicates that even for networks as large as 120 nodes, our non-optimized implementation was able to design an HP-tree for the whole network in less than a second. Note that the design phase is pre-planned and not related to the actual post-failure restoration time. This shows that running the HP-tree algorithm for every change in the network can be feasible, even though the HP-tree could self-repair or grow itself without a complete overhaul.

It must also be noted that while MATLAB® scripts are used extensively in our research for its simple and powerful scripting language and particularly easy handling of matrices, its execution speed is less than a low-level C program optimized in memory and CPU usage. Therefore, we expect the execution times presented in this section will improve further implemented using a low-level optimized language.

3.5.3. Maximum Restorability and the impact of topological constraint

We collected network restorability results from our set of random mesh graphs using the four different node placement methods (AAC, ND, MAC and DIST). As discussed in Section 2.6.2, certain topological constraints might block full restorability using an HP-tree in general. An example of such constraints was the presence of a 3-node chain; i.e. three nodes with nodal degree of two in a row. Aside from such physical limitations, a poor choice for node placement criteria or other algorithm deficiencies could also reduce restorability if along the selection process the algorithm ends up with no choice of backup parent for a specific node. Therefore a detailed analysis of the achievable level of restorability using our algorithm is necessary.

Table 3-1: Restorability Results for HP-tree

	Restorability: Random Demand	Restorability: Homogeneous
All Graphs	0.961±0.002	0.969±0.001
Graphs with no 3-node chains	0.995±0.0003	0.996±0.0003

Table 3-1 shows the average achievable restorability using our HP-tree algorithm over all random graphs in our database. The second number in each case indicates the 95% confidence interval. The confidence interval is very small, mainly due to the large number of samples. As shown, the designed HP-tree could achieve an average restorability of 0.96 in general case for any 2 edge-connected graphs in our database. For those graphs that did not contain any 3-node chain, the restorability rose to about 0.995. This result includes networks with a wide range of network parameters; nodal degrees of 2.6 to 5, network sizes from 13 to 240 links, and number of nodes from 10 to 120 nodes.

It is important to note that the achieved restorability in the above is based on the best HP-tree design, i.e. the node placement method that provided the best restorability. In total we computed more than 12000×4=48000 different HP-tree designs, i.e. four different node placement methods for each of the 12000 sample networks. The average network restorability over all of these 48000 designs was 0.94, increasing to 0.96 if the best design was chosen, and 0.995 when there was no 3-node chain in the network. Out of these 48000 designs, 83% had a restorability equal or above 90%. When the best HP design for each network was chosen, 86% had a restorability equal or above 90%. The lowest recorded restorability was about 0.42 among all 48000 design samples, and about 0.60 among those with no 3-node chain. When the network only consisted of nodes with nodal degree of three or higher (such as the case when the meta-mesh topology would be used), the lowest achieved restorability using any node placement method was 0.83, while if the best design was chosen for each network, the lowest recorded restorability increased to 0.91. If the minimum nodal degree was increased to four, the worst achieved restorability was 0.984. Individual results for each node placement method, including the statistics on full restorability for each method, will be presented later in this section.

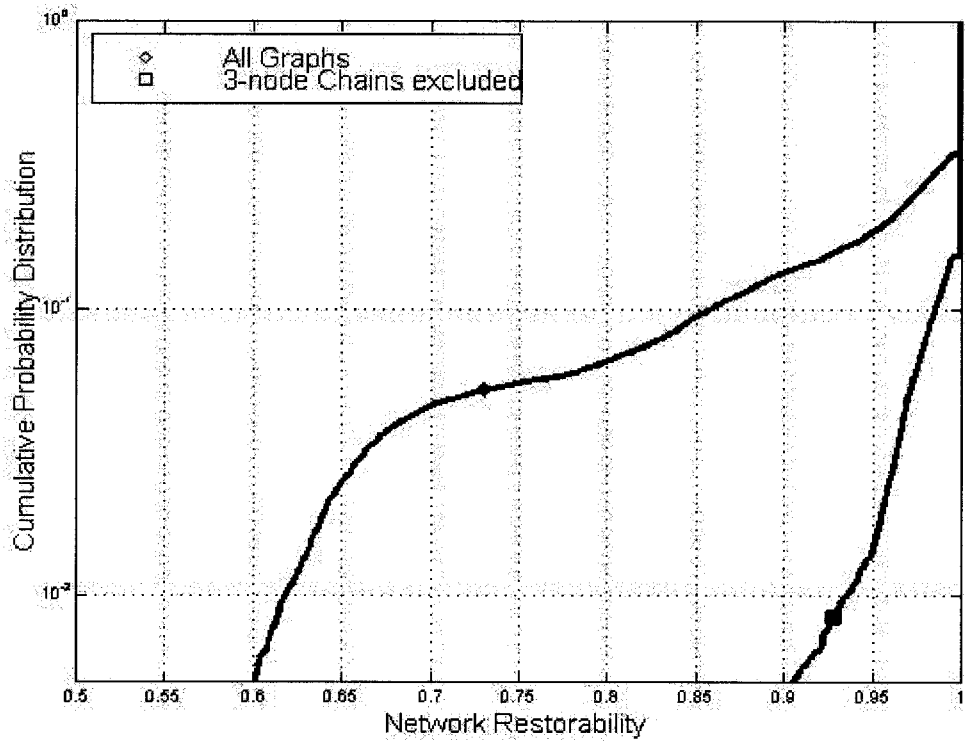


Figure 3-7: Distribution of HP-tree Restorability in graphs with and without chains

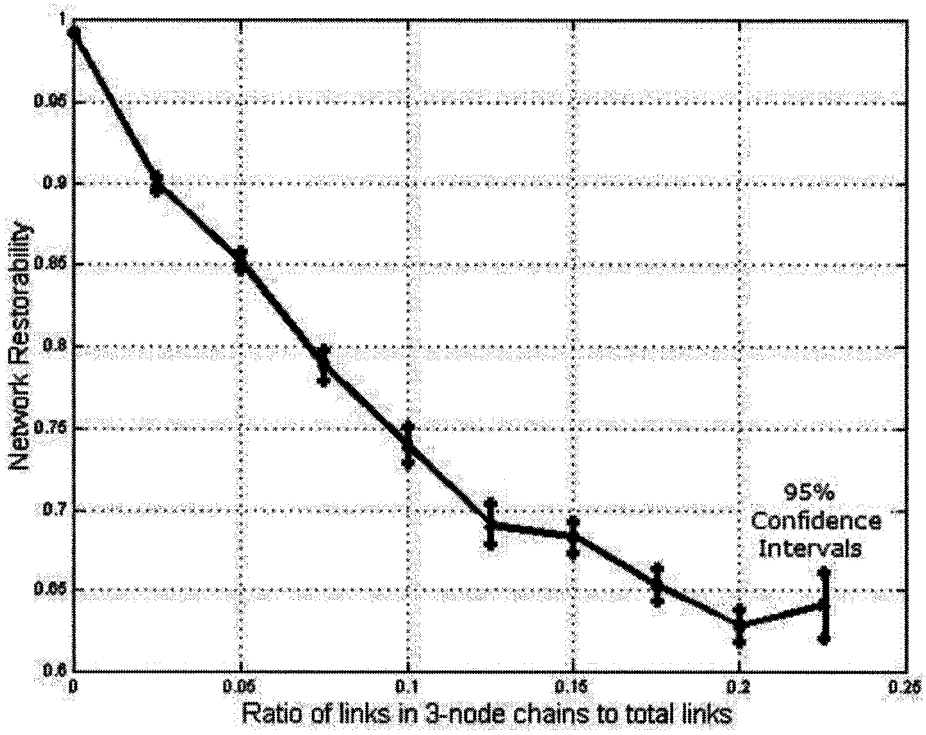


Figure 3-8: Impact of 3-node chains on Restorability

Figure 3-7 shows a graphic illustration of the results by showing the cumulative distribution of network restorability results for the random demand case (presented in Table 3-1). When the graphs with 3-node chains were excluded, the algorithm was able to achieve restorability of higher than 95% in 97% of the cases. With 3-node chains included, the achievable restorability was higher than 95% in about 80% of the cases, and higher than 90% in 86% of the cases. Because a total sample space of about 12000 networks was used for stat collection, the results below 0.01 were not considered statistically accurate enough.

Figure 3-8 shows the decrease of maximum network restorability with the number of 3-node chains. In this case, the ratio of the number of 3-node chains to the total network size (number of edges) has been used. In order to compute the points on this curve, the whole range of the ratio was divided into ten intervals of 0.025 width each (there were not enough samples for the ratios of 0.25 and higher), and the results for each interval was average as one point on the curve. The points on the curve were connected by straight lines. The 95% confidence intervals are also shown on the curve, which at the worst point (the interval of [0.225 0.25[ ) was about ±3.2% of the average value at that point. As mentioned before, each 3-node chain affects one node. Therefore, a chain ratio of 0.2 in a network of 240 links would mean that about 48 nodes in that network were located in 3-node chains. The higher the ratio, the more the networks are ring-like rather than mesh-like. This curve confirms that as a result of its topological constraint, the HP-tree does not perform well in sparsely populated (or ring-like) networks with many node chains; however the more mesh-like the network is, the higher the restorability of the HP-tree. In networks with no chain where each 2-degree node is connected to at most one other 2-degree node, the restorability remains close to 100%.

We also collected the restorability stats separately for each of the proposed node placement methods; i.e. AAC, MAC, ND and DIST. The result for each method is reported in Table 3.2, for all graphs and for graphs with no 3-node chains. The second number next to the restorability in the first column indicates the 95% confidence interval. For the statistics collected from graphs with no 3-node chain, the 95% confidence interval was smaller than ±0.001.

Table 3-2: Mean Restorability for each node placement method

	Mean Restorability for all graphs	Mean Restorability for graphs with no 3-node chain
AAC	0.939±0.002	0.979
MAC	0.940±0.002	0.979
ND	0.950±0.002	0.989
DIST	0.957±0.002	0.993

The DIST method provided the best restorability at the cost of higher complexity. The ND method appears to be a good tradeoff in the spare capacity assignment scenario. For graphs with no 3-node chain, all methods achieved restorability of over 0.97. Among the four options, the DIST method achieved full restorability (R=1) in 60% of cases, and in 80% of cases where the network did not have any 3-node chain. The respective numbers for other options were: 50% and 65% for the ND method, and 32% and 42% for AAC and MAC. The results for the AAC and MAC methods were very close. The average results in Table 3-2 indicate that even in cases when full (100%) restorability was not obtained, the achieved restorability was usually very high (over 90%).

The restorability of the HP-tree is affected directly to the minimum nodal degree in the network, which is a controllable design parameter. Notably, a minimum nodal degree ≥ 3 is sufficient condition for no 3-node chain (but not a required condition). The higher the minimum nodal degree of the network, the better chance that a backup parent node could be selected for each network node.

Table 3-3: Impact of minimum nodal degree on Restorability

	$D_{min} = 2$ (all)	$D_{min} = 2$ (no chain)	$D_{min} = 3$	$D_{min} = 4$
Mean Restorability	0.92	0.99	0.995	0.999
95% Confidence Interval	±0.003	±0.001	±0.0003	±0.00005

As the results in Table 3-3 indicate, the HP-tree restorability jumped to >99% once the graphs had no 3-node chain.

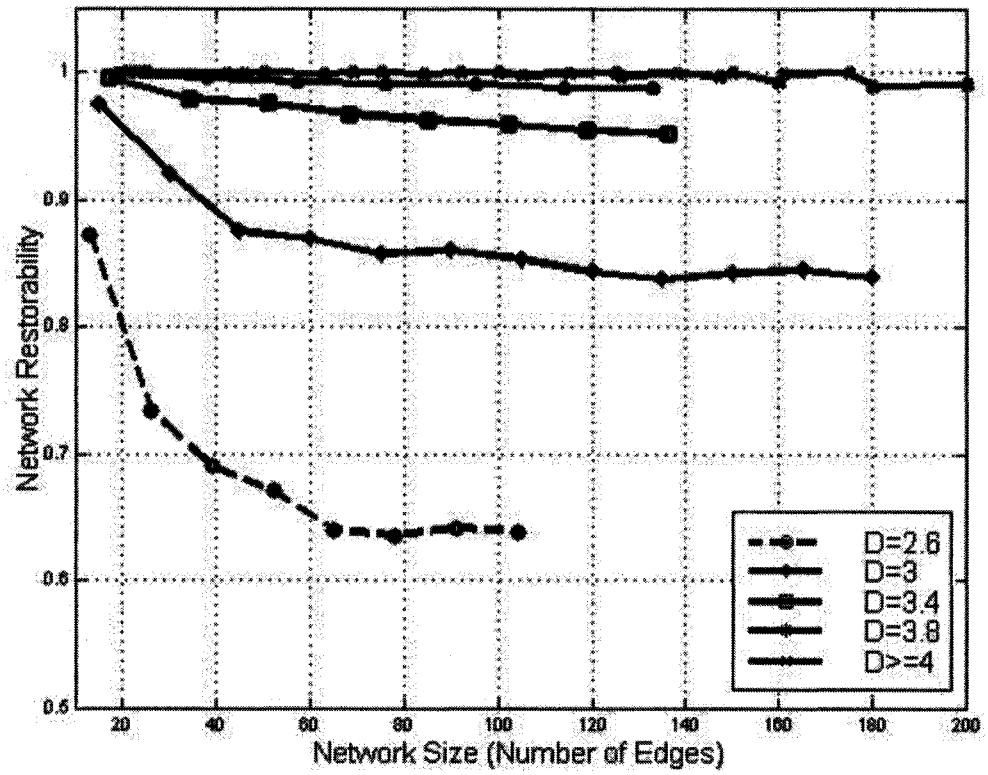


Figure 3-9: Restorability versus network size

Figure 3-9 shows how the average restorability changes with number of edges in the network. For this graph, results from all network topologies (with and without 3-node chains) were used. The points on each curve are the average values obtained for the best HP-tree designs for networks with the same size and nodal degree. For networks with average nodal degrees of 4 and higher, restorability stands at near 100% regardless of network size. For lower nodal degrees, the increase in network size negatively affects the ability of the algorithm to guarantee both primary and backup parent nodes for each node of the network, as well as increasing the number of 3-node chains in the network topology. For average nodal degree of 3.4, the restorability drops to 95% at M=136 links. At average nodal degree of 3 (which implies that a large portion of nodes have a nodal degree of two), the average restorability drops to about 83% for a network of 180 links.

When the average nodal degree is reduced to 2.6, the network topology becomes closer to ring and the number of 3-node chains sharply increases. As a result, the restorability of the HP-tree design falls sharply. For networks of 13 links with average nodal degree of 2.6 (10 nodes), the mean network restorability was about 0.87. For a network of 104 links, restorability fell to 0.64. The 95% confidence interval for the points on the curves of D=3.4, 3.8 and 4.0 was within ±0.003 of the mean value at each point. For the curves of D=2.6 and 3.0, the 95% confidence interval was about ±0.015 of the mean value at each point.

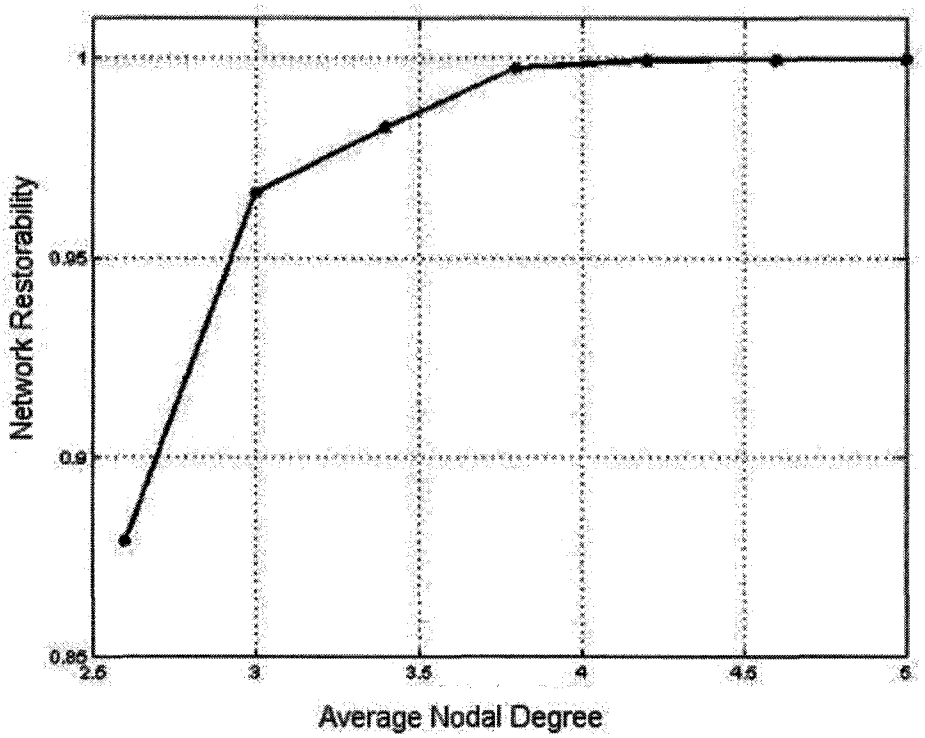


Figure 3-10: Restorability versus Average Nodal Degree

For the results in Figures 3-10 and 3-11, we studied the isolated impact of graph topological parameters on restorability in absence of 3-node chains. Therefore in these curves only networks without 3-node chain were considered. Figure 3-10 shows how the network restorability changes with nodal degrees on average. The points on the curve are average values of restorability for networks of the same average nodal degree. The points on the curve were connected by straight lines. As this Figure shows the restorability on

network graphs with average nodal degree of 3.8 and higher was very close to 100%, while it dropped for lower nodal degrees down to about 88% for D=2.6. The 95% confidence interval was within ±2% of the mean value at D=2.6, ±0.7% at D=3.0, and ±0.25% (0.0025) of the mean value for $D \geq 3.4$.

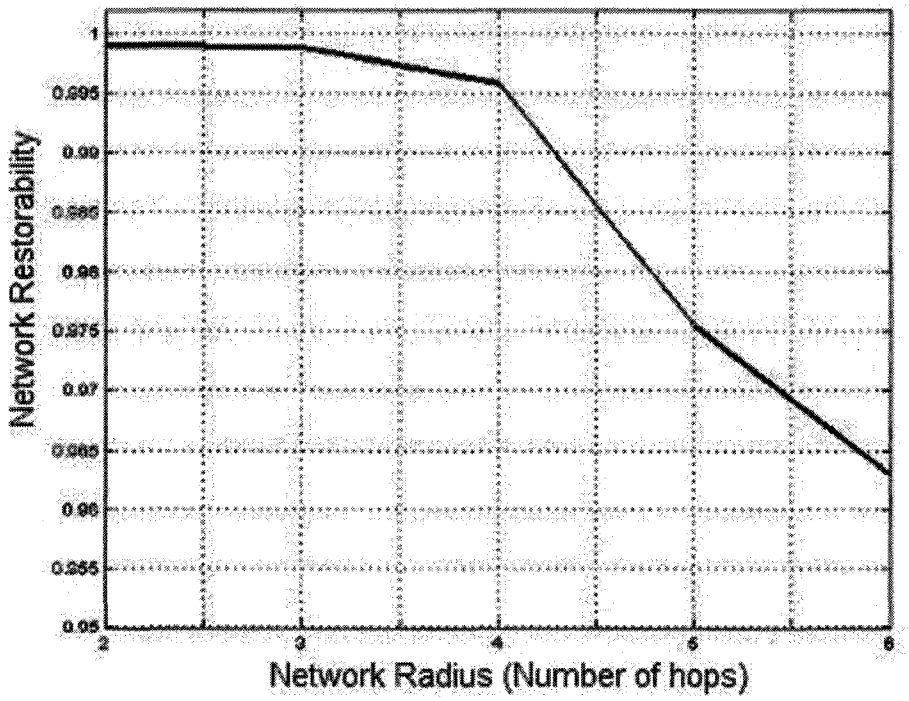


Figure 3-11: Restorability versus Network Radius

Figure 3-11 show the restorability versus network radius which provides an indication of the mesh-ness of the network, as the more mesh like the network is, the smaller number of hops must be traversed on average between the nodes. Our results showed that the HP-tree could achieve restorability of 99.5% on average in networks with radius of four hops or less. The 95% confidence interval was ±1.1% of the mean value at *rad*=6, and less than ±1% of the mean value at other points.

Based on the results of this section, it is clear that the topological design of the network plays an important role in maximum restorability that the HP-tree could achieve in that network. The impact of 3-node chains on the restorability was significant, as networks without chains could achieve restorability over 99%. Making the network more

mesh-like by reducing the network radius and average eccentricity also improves the performance of the algorithm.

3.5.4. Network Redundancy Analysis

In this section the total redundancy results of the spare capacity assignment will be presented. In studying redundancy results, one must note that statistical deductions or comparison of the efficiency between different methods are valid only if the results are considered for the same restorability level. Otherwise, a network with redundancy of zero (and restorability of zero) would be taken mistakenly as the most efficient. Therefore for the purpose of the redundancy analysis in this section we used results for which the HP-tree algorithm achieved full (100%) restorability. In cases where statistical results are presented, we use the average and standard deviation values of redundancy (ratio of total spare capacity to working capacity) among networks with the same parameter. The results are presented in form of Capacity Sharing Factor (inverse of redundancy).

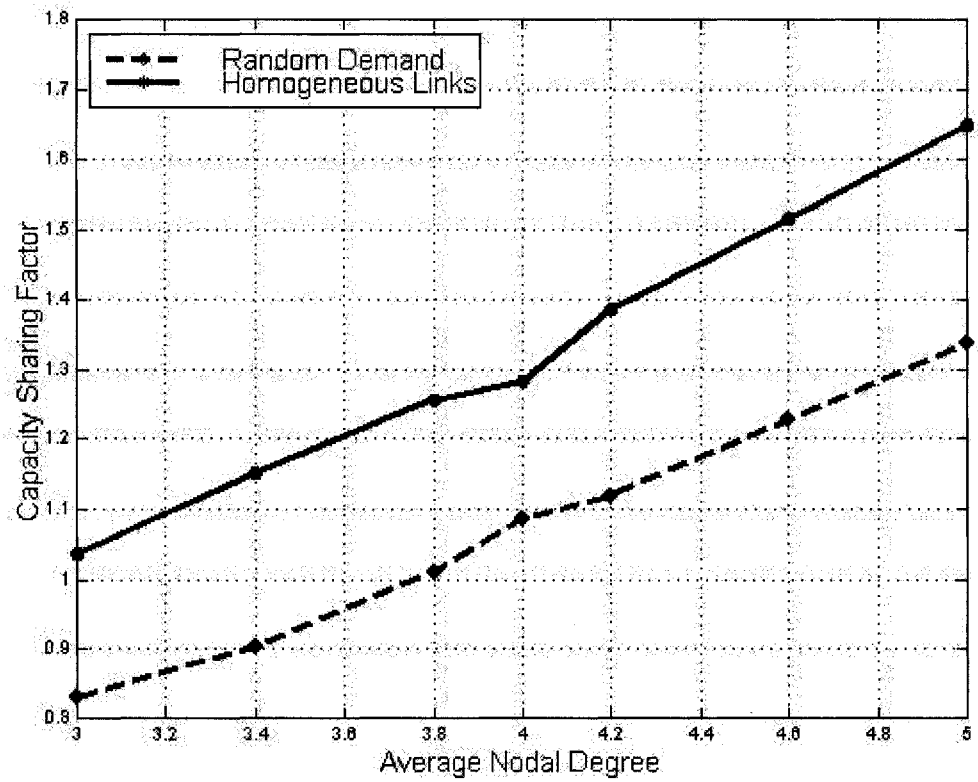


Figure 3-12: CSF versus Average Nodal Degree

Let us first start with the curve of CSF with respect to the change in the average nodal degree of the network as shown in Figure 3-12. Two cases have been presented: the random demand scenario with uniform distribution in random demand between zero and 10 units between each pair of nodes, and the homogeneous working link scenario where all links would have the same working capacity. Each point on the curve is the average value of all networks with the same average nodal degree. As expected, the homogeneous case provides better sharing efficiency of the spare capacity. The capacity sharing of the HP-tree increases linearly with average nodal degree. The result in Figure 3-12 is based on the node placement that achieved the best restorability. However, the results for different node placement methods were close, within 3% of each other. The performance results are close, indicating that the efficiency improvement resulting from shorter backup paths of the DIST and ND methods provided nearly the same efficiency improvement from arrangement of spare capacities on the trees of the MAC and AAC methods. The 95% confidence interval was within ±1% of the mean value at each point.

Our simulation results did not show any specific change in capacity sharing efficiency when the network size changed. Overall, it appeared that the nodal degree of the network played a much more significant role than the size of the network. An increase in nodal degree directly increases the number of those non-tree links that benefit from tree protection without having to contribute to protection paths. More links will share the backup paths provided through the tree. As a result, the capacity sharing factor increases with nodal degree.

Average values for CSF was about 1.20 for each individual node placement method that achieved full restorability, and as mentioned, the results for the four methods were very close. However, the best and worst recorded results varied. Among cases where full restorability was achieved, the best recorded CSF for the ND and DIST methods was about 1.57 in both cases, and about 1.50 for the AAC and MAC methods. The worst recorded CSF was about 0.71-0.73 for all four methods. As mentioned at the beginning of this section, comparison of CSF for different methods is meaningful only when the same level of restorability (full restorability for results in this section) is achieved.

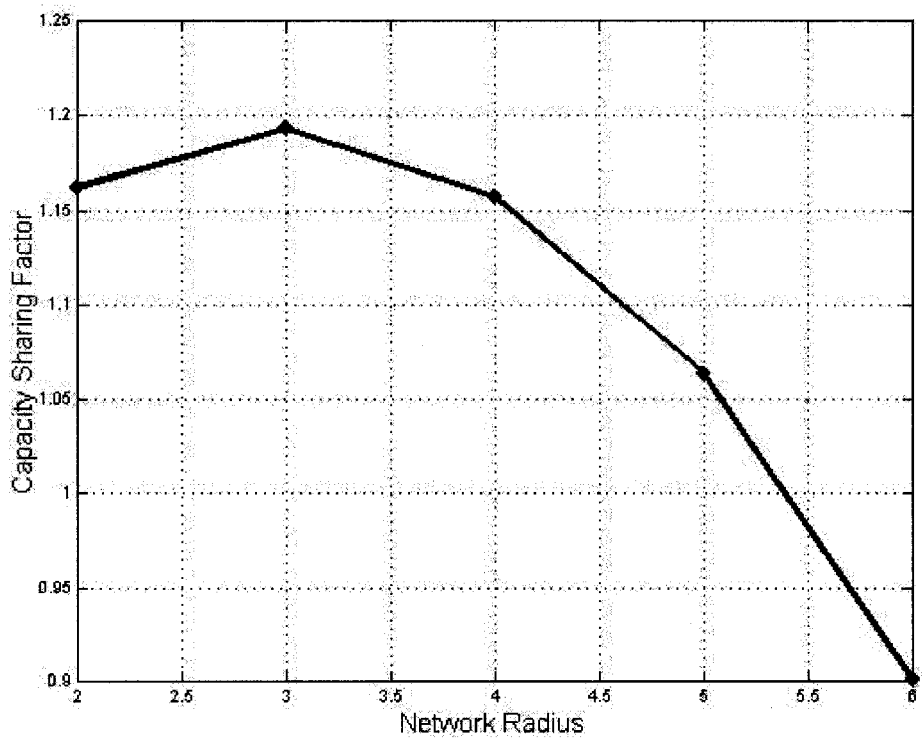


Figure 3-13: CSF versus Network Radius

Figure 3-13 illustrates the change in capacity sharing efficiency versus network radius *rad*. A similar curve was observed for the change with average node eccentricity. Both parameters indicate longer network paths and less mesh-ness, so the capacity sharing decreases with an increase in each parameter. As with restorability results, here we also observe that a topological design that incorporates more mesh-ness increases both the restorability and the capacity efficiency achieved by the HP-tree algorithm. The Capacity Sharing Factor increased slightly at *rad*=3, but the reason could be that the average nodal degree of the chosen samples happened to be higher at *rad*=3 (*D*=4.5) than *rad*=2 (*D*=4.3). As indicated in Figure 3-12, such increase in average nodal degree improves the CSF. The same factor is also partly responsible for the decrease in the CSF for larger network radius, as network radius increases when average nodal degree decreases (in networks with the same number of nodes).

The 95% confidence interval in Figure 3-13 was within ±1.2% of the mean value at each point, except at *rad*=6 for which we only had one sample and thus no confidence

interval could be specified at that point. At every other point the number of samples was larger than 100.

It would be interesting to compare our results with HP-tree with the results reported on protection cycles in [Douc05], in particular because that reference also used random graph simulation for performance analysis. Direct comparison is not possible as only six families of arbitrary chosen graphs were used in [Douc05]. Also, the method of generating various average nodal degrees by partially modifying a given graph (as done in [Douc05]) does not create totally independent graphs as we did in this study. However a comparison of trends might be somehow valid. The p-cycle ILP SCA and span ILP SCA results are used here for comparison with HP-tree heuristic SCA scenario. For the 20n40s1 family in [Douc05], the cost redundancy for p-cycle and span SCA were respectively about 115% and 130% at D=3, 95% and 115% at D=3.4, and about 80-85% at D=3.8. Respective average results for our HP-tree design for random demand case with 0-10 units (same demand scenario as in [Douc05]) were 120% at D=3, 109% at D=3.4 and 97% at D=3.8. For the 40n80s1 family the results were nearly identical for p-cycle and span SCA, at about 90% at D=3.4 and 3.8. For our approach the respective results were about 108% at D=3.4 and 97.5% at D=3.8. There were no simulation results in [Douc05] for larger networks or higher average nodal degrees, but the above examples indicated that our average HP-tree heuristic results were within 10-15% of the results reported by ILP SCA results in [Douc05]. However note that in [Douc05] the protection cycle and protection paths were designed with no hop limits. As [KoSa04] shows, it might be possible to expect that if hop limits are applied on cycle and paths to bring their average backup paths lengths within the same range as the HP-tree, the redundancy requirements of protection cycle and optimized mesh design might increase to the same levels. It also appears that the p-cycle and span ILP SCA designs increase their capacity efficiency advantage over tree-based design when the average nodal degrees increase; though a more accurate comparison is only possible if simulation results with random networks were available in those cases too.

3.5.5. Efficiency Comparison with Optimal Tree Design

In this section we will provide comparative results for the performance of the HP-tree versus the results of the generic shared backup tree optimization problem formulation

reviewed in Section 3.1. Note that the formulation by [TaRu05] designs a protection tree in which tree links could be protected by *any* non-tree link, not necessarily a non-tree link adjacent to failure. As a result, that formulation has a higher degree of freedom in choosing backup path and could achieve higher redundancy efficiency. On the other hand because the HP-tree uses adjacent non-tree links, it could achieve better restoration times, could repair itself with minimum management, and could handle multiple failures in real time. Here we examine the redundancy performance to observe how much the advantages of the HP-tree would cost us in terms of spare capacity requirements.

Table 3-4: Parameters of Sample Graphs in [TaRu05]

	Number of Nodes	Number of Edges	Average Nodal Degree	Max-to-Min Working Capacity
Tang_net1	10	22	4.40	6
Tang_net2	15	28	3.73	21
Tang_net3	11	22	4.00	14

We use performance results from [TaRu05] in which the results were presented for three arbitrarily designed sample networks. Those network topologies are presented in Appendix A.2. Here we refer to them as *Tang_net1*, *Tang_net2* and *Tang_net3*. The topological parameters of these networks were presented in Table 3-4. In order to be able to make a meaning full comparison, we requested the working link capacity information for each network from the authors of [TaRu05] and used that information directly here. Spare capacities were allocated as needed for full network restorability. The HP-tree was able to achieve 100% restorability for all three networks.

Table 3-5: Comparison of HP-tree Redundancy with Optimal Tree Design

	Optimal Tree CSF	HP-tree CSF	Node Placement Method
Tang_net1	1.4792	1.1639	DIST
Tang_net2	1.0755	0.8507	MAC
Tang_net3	1.0674	0.8879	MAC

Table 3-5 shows the results for the optimal tree from [TaRu05] and the results we achieved with our HP-tree design algorithm. The results were presented as Capacity Sharing Factors (CSF). For *Tang_net1*, we achieved the best redundancy with the DIST node placement methods, while for *Tang_net2* and *Tang_net3* the MAC method achieved slightly better results. We also observe the negative impact of increased max-to-min working capacities in the network on the capacity efficiency. Overall the capacity efficiency of the HP-tree was between 20% to 27% less than the optimal tree design.

Table 3-6: Comparison of HP-tree backup path lengths with Optimal Tree Design

	Optimal Tree mean backup path length	HP-tree mean backup path length
Tang_net1	4.4550	2.6364
Tang_net2	8.1070	2.8214
Tang_net3	5.2270	2.2273

Table 3-6 shows the mean backup path length results for the optimal tree and the HP-tree design in the tree sample networks. The HP-tree backup path length results in Table 3-6 were based on the same method that achieved the best redundancy; i.e. DIST for *Tang_net1* and MAC for *Tang_net2* and *Tang_net3*, even though those were not the best backup path length results (e.g. we achieved mean backup path length of 2.1818 hops in *tang_net1* with the ND and MAC methods); however we present the backup path lengths for the same methods in Table 3-5 to make valid comparison for cost versus backup path lengths. As Table 3-6 shows, the backup path lengths were significantly shorter in the HP-tree than the design based on the optimal tree formulation in Section 3.1

3.5.6. Mean backup path length

As defined in Chapter 2, the mean backup path length is measured as the average length of the valid backup paths (in number of hops) provided to network links by the HP-tree scheme. We also present in here the results for maximum backup path length in the sample networks. The statistical results were collected from those graphs in which at least

one node placement method could provide full restorability. There was no significant difference in mean backup path length results for different demand scenarios; i.e. random demand versus homogeneous links.

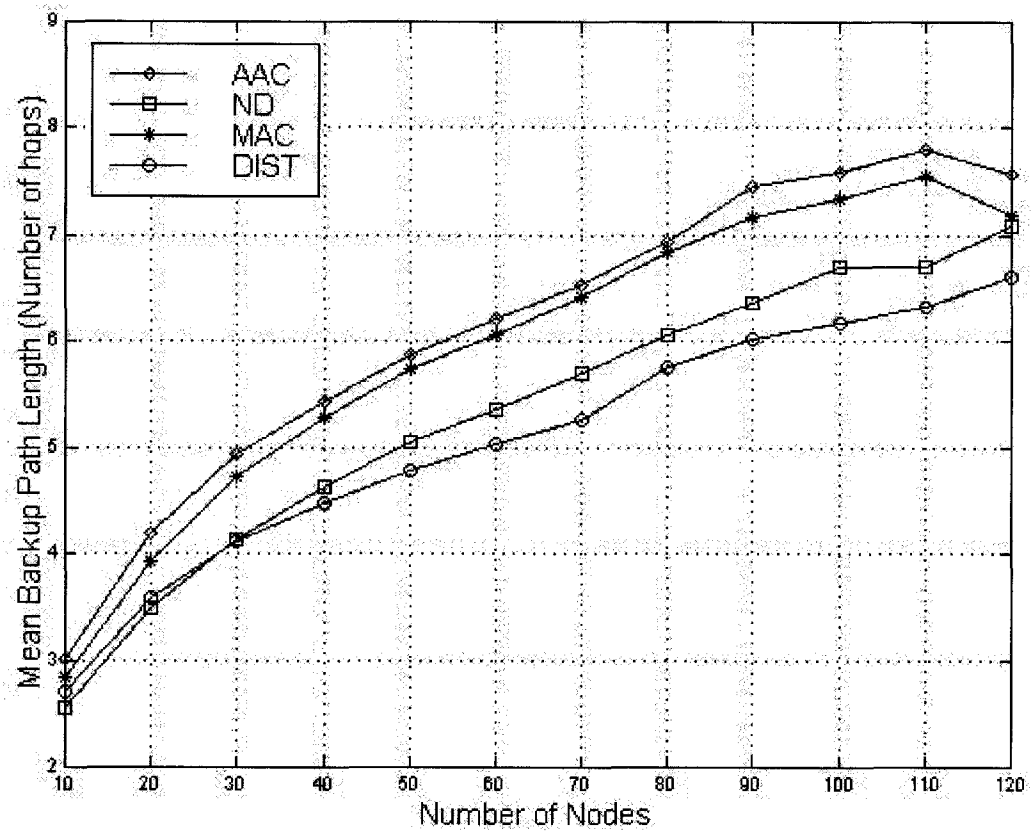


Figure 3-14: Mean Backup Path Length versus Number of Nodes

As Figure 3-14 shows, the ND and DIST node placement methods provided the shortest backup paths. Our results indicated no specific trend with regard to the average nodal degree. For networks as large as 120 nodes, the mean backup path length is about seven hops. Note that $log_2(N)$, in which N is the number of nodes, appear to provide a good approximation for mean backup path lengths. However we should remind the reader that the reason for this relationship is that in our random mesh graphs, the radius of the network increased logarithmically with network size. The shape of the curve of the mean backup path length is more related to the network radius (as our next figure shows). For instance in ring topologies where network radius changes linearly with network size, the

mean backup path length of the HP-tree also follows the same trend. However the logarithmic behaviour appears to be valid for mesh networks.

Two anomalies could be noticed in the graph. First, in the range between 90 and 110 nodes, the curves of the mean backup path increased at a higher rate than the normal logarithmic trend of the curve up to that point. The reason is related to the characteristics of the sample set; for N in the [90, 120] range the average nodal degree of the networks in the sample set used in this figure was about 4.0, while for smaller values of N the average nodal degree was about 4.5. We expect that smaller nodal degrees would result in networks with larger radius and thus, longer backup paths (as the next figure will indicate). At N=120 the mean backup path slightly decreased, but the reason could be attributed to the small number of samples at that network size (5 samples at N=120) for this study. As mentioned at the beginning of this section, we selected a subset of networks for which at least one HP-tree design could achieve full restorability. With that criteria, for N in the range of [90, 110] we had 50 samples or more for each point on the curve. For $N \leq 70$, each point is the mean value of 800 samples or more. The 95% confidence interval was within ±0.8% (0.008) of the mean value at each point for $N \leq 70$, and within ±3.5% of the mean values for $90 \leq N \leq 110$. It increased to about ±9% for N = 120. Therefore the result at N=120 is much less accurate than the rest of the curve points.

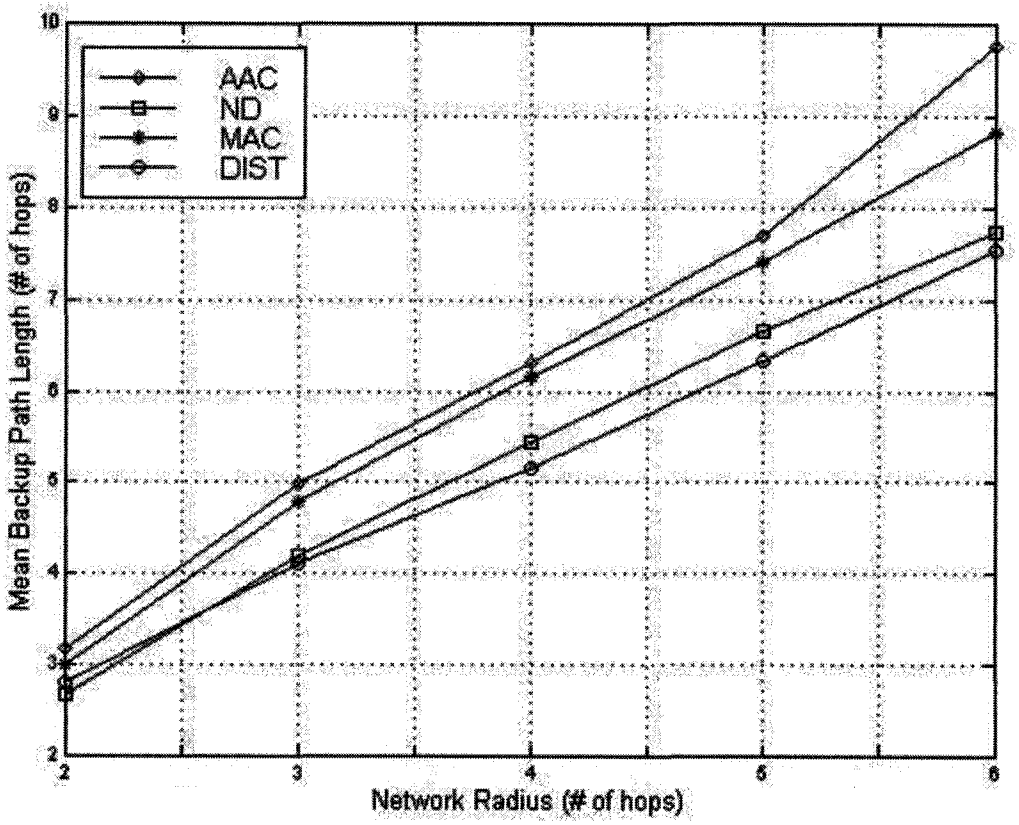


Figure 3-15: Mean Backup Path Length versus Network Radius

Figure 3-15 compares the effect on mean backup path length with respect to network radius among the four node placement methods. The curves indicate a linear trend for the expected value of mean backup path with respect to network radius. The 95% confidence interval is within ±1% of the mean value at each point for $rad \leq 4$, and ±2% of the mean value at rad=5. For rad=6 there was only one sample, so there was no confidence interval.

In addition to mean backup path length, we also collected statistics on the maximum backup path length in each graph. This is an important parameter for comparison with other schemes that specify a hop limit for the backup path length.

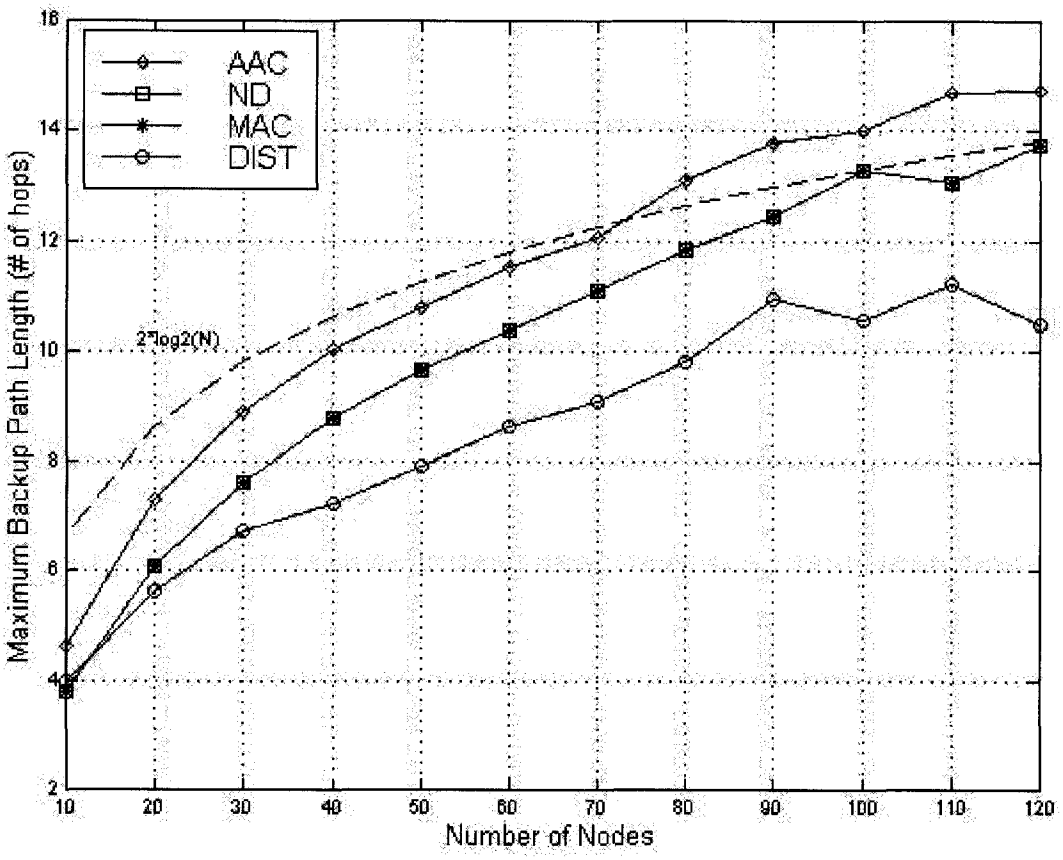


Figure 3-16: Maximum Backup Path Length versus Total Number of Nodes

In our simulation of random mesh networks, the maximum backup path length on the HP-tree was typically within less than twice of the mean backup path length values. The maximum backup path length ranged from four hops on average for a network of ten nodes, to 14 hops on average for a network of 120 nodes. The slope of the curves in Figure 3-16 could be approximated reasonably with $2*log_2(N)$, which is shown in dashed line.

3.6. Concluding Remarks

Our novel contributions in this chapter, to the best of our knowledge, included heuristic design algorithms for construction of hierarchical-protection tree in a given mesh network, as well as extensive simulation results to study the performance of our proposed protection tree in terms of required redundancy in the network and backup path length distribution. We showed that the HP-tree was able to provide backup paths with

reasonable length even in large mesh networks. Our study on the topological constraints of the protection tree showed that if 3-node chains did not exist in the topology, full or near full (in the range of 99.5%) restorability could be achieved in most mesh networks. Simulation results were presented to link the performance and restorability of the hierarchical protection tree to demand characteristics and various network parameters such as network size, number of nodes, average nodal degree, average node eccentricity and network radius.

Chapter 4. Implementing Protection Trees in Pre-designed Networks

In this chapter we study how an HP-tree could be implemented in a pre-designed network to achieve maximum restorability. In this scenario the network under study might be an operating network that may not have any specific protection scheme in place or may have been using a different scheme from an HP-tree, We then have to determine how we could organize the available spare capacity of the network into an HP-tree.

The rest of this chapter is organized as following: In Section 4-2 we consider some modifications to our HP-tree design algorithm that suit the pre-designed network scenario. In Section 4-3 simulation models using real networks will be proposed, and performance results of the algorithm will be studied. In Section 4.4 we present an approximate random analysis of restorability that could be used for estimation of the trends and changes in restorability with network size and capacity in special cases. Section 4.5 provides concluding remarks for this chapter.

4.1. Problem Statement

The scenario under study in this chapter involves the design of a protection tree in pre-designed networks with fixed working and protection capacities on each link. Obviously the objective of the HP-tree design algorithm would be different from the capacity assignment scenario of Chapter 3. While in that case we were trying to minimize the required spare capacity that had to be assigned in the network, in this case we would like to maximize the restorability that we could achieve using the available spare capacity of the network.

Total network restorability could be limited by several factors, for instance the network load, availability and distribution of spare capacity, and the effectiveness of the algorithm in utilizing the spare capacity to provide backup paths. For example, total network load might be moderate enough that would leave sufficient spare capacity in the network, but that spare capacity might not be present at locations where we need it. Alternatively, the network may have enough spare capacity for protection should an overall optimization problem is solved, but a heuristic algorithm that focuses on local failure recovery might be less efficient in utilizing the available spare capacity and thus

might fail to provide full restorability. All these factors would have an impact on the restorability of the network.

4.2. Tree Design Algorithm in a Pre-designed Network

4.2.1. Design Objectives

In this scenario we would like to have a heuristic HP-tree design algorithm that as an input would accept a network topology (usually in form of a topology vector or adjacency matrix), working capacities on network links, and spare capacity on each network link, and would put out an HP-tree in form of a vector of primary and backup parent nodes, that provides backup paths for network links and achieving high restorability and efficient use of the available spare capacity of the network. We assume that the working and spare capacities on each link have been determined prior to the running of the algorithm, and based on various factors such as fixed total capacity of each network link, demand between each pair of nodes, limitations on the size of network equipment etc. Models for determining the working and spare capacities on network links will be presented in Section 4.3.

Note from Equation (2-4) that restorability increases with the amount of protected working capacity in the network. This implies that it is important to maximize the available protection capacity on the protection path for each network edge. This fact leads us to the following principles to formulate our tree construction strategy in the pre-designed network scenario:

1. It is preferable to include edges with larger spare capacities in the HP-tree.
2. It is preferable to place higher in the HP-tree (closer to the root node) the edges with larger spare capacities.

The first condition directly increases the amount of protected working capacity and thus provides a better chance of maximizing R.

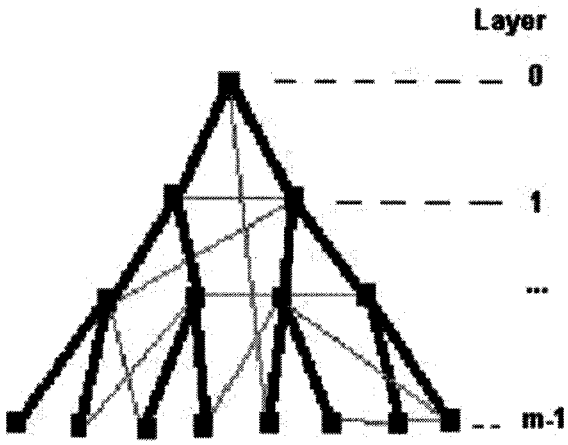


Figure 4-1: Binary HP-tree

As for the second condition, note that each edge is responsible for protection of those non-tree links that connect the child subtree of its downstream node to the other parts of the network. Consider a random mesh network with an average nodal degree of k, protected by a binary HP-tree as depicted in Figure 4-1. Let us number the tree layers from the root node (layer 0) to the leaf nodes (layer $m-1$); hence there are 2^m-1 nodes in the binary tree. Consider a tree edge that connects a node in layer i to its parent node in layer $i+1$. The child subtree of the downstream node of this link (a layer-i node) has 2^i-1 nodes (including the downstream node), each on average may have $k*2^{i-m}$ links to other parts of the network (excluding this subtree). The average number of links protected by an edge (defined as *sharability* in Chapter 2) in layer i is therefore in the order of $k*2^{2i-m}$, which increases exponentially with layer number i. Thus, the sharability of the edges in the network increases exponentially when they are put in higher levels of the hierarchical protection tree.

The above argument was meant to justify the statement that the edges with larger spare capacities must be put in higher levels of the HP-tree, because they can handle the associated sharability of the higher layers in a more efficient manner. Recall the node placement criteria in Chapter 3. In that scenario we used the placement algorithm to reduce the length of backup paths on the tree or to increase the sharing of capacity, in order to increase the Capacity Sharing Factor of the network. In the present scenario in here we use capacity-based placement methods such as AAC and MAC to push up in the

tree those nodes with more spare capacity and therefore, better utilization of the spare capacity to achieve higher restorability.

Now we proceed with a more detailed description of our HP-tree design algorithm for pre-designed networks.

4.2.2. HP-tree design algorithm for pre-designed network

Based on the discussions in Section 4.2.1, the modified HP-tree design algorithm for the pre-designed network is presented in the following:

HP-tree construction algorithm for the pre-designed case

#Choosing the primary parent nodes

1. Create Vector of Nodes V, empty tree node vector U and empty tree edge vector T.
2. Sort V by HP-tree node placement algorithm (Section 3.2)
3. $U(1) = V(1)$ (*Add the root node to the tree vector*)
4. Add all neighbors of the root node to the tree node vector U. Add corresponding links between those nodes and the root node to the tree edge vector T.
5. Repeat the following loop of steps until all nodes are added to the tree vector U:

5.1. Choose the highest node v_i in V that:
- is not a member of tree vector U, and
- is connected to a member of tree vector U

5.2. Create vector L of all members of the sorted vector V that are connected to v_i. L is thus a subset of V. Sort L based on the protection capacity of the links between v_i and nodes in L.

5.3. Going from top to bottom in L, choose the first node l_k that has at least two non-tree links adjacent to it, unless:
- l_k is the root node of the tree, or
- no node with the above condition exist in the tree, AND all other remaining (non-tree) nodes have a nodal degree of two.

In each of the above cases, choose the first entry l_k in L.

5.4. If an l_k was found in the previous step:
- add v_i to the tree node vector U.
- designate l_k as primary parent of v_i.
- add (l_j, v_i) to T.

5.5. If reached the bottom of the node vector, reset the pointer to the top of the node vector.

5.6. Return to Step 4.

#Choosing the backup parent nodes

6. For each node v_i in vector V:

6.1. Create a vector of adjacent edges E_a.

6.2. Exclude edges to neighbours that are in the subtree of v_i.
6.3. Sort E_a based on protection capacity of edges in E_a.
6.4. Choose the highest edge e_j in E_a that is not a member of T.
6.5. Determine $e_j = (v_i, v_k)$. Assign v_k as the backup parent of v_i.
6.6. End of the loop.

Note that here in capacity-based node placement methods (AAC and MAC), we use the amount of *protection* capacity on network links for node sorting, not the working capacity as used in Chapter 3. The purpose here is to maximize the use of spare capacity through highly shared nodes. Also in selection of primary parent node, instead of connecting each node to its highest neighbour in the sorted vector of nodes (as we did in Chapter 3), here we choose the upstream of the adjacent link with highest spare capacity as the primary parent of this node (Step 5.2). This arrangement increases the possibility that those links with larger spare capacities end up on higher branches of the HP-tree, as we intended.

Furthermore in the spare capacity design case of Chapter 3, we showed that triangles would provide better capacity efficiency for that case. However in the pre-designed case the spare capacity is already determined and the objective is to increase restorability, which the triangular subtree approach would not necessarily provide as it does not take the available spare capacity of the links into account. Thus we should also design a new backup parent assignment algorithm that attempts to include links with more spare capacities in the backup paths in order to increase restorability. Recall that the choice of backup parent nodes determines those non-tree links that contribute to the protection of tree links (based on theorem 2-3). Also recall that only those links that do not connect to the child subtree of a node are eligible for this purpose. Based on the above argument, our algorithm chooses as backup parent of each node the upstream node of the link with largest spare capacity from the set of eligible adjacent links of this node (Steps 6.2 to 6.5).

4.2.3. Algorithm correctness and complexity

The correctness of the sub-algorithm for selection of primary parent nodes was shown in Section 3.2.3. The modifications applied to the previous Algorithm in 3.2 do not affect any part of the argument presented in Section 3.2.3. So the algorithm will always be able to construct a spanning tree in any connected network.

The complexities of the primary-parent selection sub-algorithm and node placement criteria remain the same as spare capacity assignment algorithm in Section 3.2. As for backup parent selection the algorithm here is simpler than the one presented in Section 3.4, as no tree distance matrix has to be computed here. The complexity of the backup parent selection sub-algorithm is determined by:

- Exclusion of backup nodes from the child subtree, which can be done on average in $O(D)$ for each node and $O(M)$ in total.
- Sorting of the adjacent edge at each node, which can be done in $O(D \log D)$ for each node, and $O(M \log D)$ in total.

Based on the above, we expect the complexity of the backup parent sub-algorithm in the pre-designed case to be $O(M \log D)$.

4.3. Performance Evaluation

We studied the performance of the HP-tree in pre-designed networks using both random and real network scenarios. The random networks were generated based on the topology model described in Section 2.2. Capacity models for real and random networks were discussed in Section 2.4. We used the homogeneous capacity and homogeneous load models in various cases in our simulations here. We did not use the random link capacity model in our simulations, but considered it in our approximate statistical method in Section 4.4.

For each graph, we applied our HP-tree design algorithm to construct the HP-tree in the network using the four suggested node placement methods. In design of the HP-tree and performance computation for each network we used the placement method that provided the best result for that specific network, unless comparative performance results of the node placement methods were discussed (such as Figures 3-5, 3-14, 3-15, 3-16 and Tables 3-2 and 3-5) in which case the results were collected separately for each placement method in each network. The RP matrix for each case was computed and Restorability was calculated based on Equation (2-4).

4.3.1. Random Mesh Network Simulations

In this scenario we used a set of about 1000 random mesh graphs from our network database for simulations. The graphs were generated based on the algorithms described in Section 2.2. The homogeneous total link capacity model was used in this case and the total capacity of each link was set to the OC-1/STS-1 size. For populating the working capacity of the links we generated DS-1 demands between randomly selected pairs of nodes in the network, and routed them based on two parameters: *maxLinkLoad*, the maximum allowed load of a link, and *networkLoad*, the ratio of total working capacity to total link capacity of the whole network. In order to get a realistic model of a pre-designed network under the conditions set in Section 4.1, the routing algorithm was implemented to route individual demands on the minimum weight shortest available path as long as the ratio of working to total capacity on each segment of the path remained below *maxLinkLoad*. In order to achieve load balancing among network paths, the weight of each segment was incremented with each new demand routed through it. Once a segment was capacitated to the *maxLinkLoad*, that link would be removed from routing algorithm. The demand generation algorithm would continue to generate DS-1 demands between randomly selected pair of nodes until the total working capacity of the network reached the *networkLoad* level, at which point the total number of working channels on each link would be computed and the remaining channels (from a total of 28 available DS-1 channels on each link) would be assigned as spare capacities. Then the topology vector of the working and spare capacities on network links for each graph was fed to the HP-tree design algorithm of Section 4.2 for computation of restorability.

The above model allows us to consider pre-designed fixed-capacity networks with specific load parameters; both per-link and per-network and to compare performance of HP-tree under various load conditions.

Simulations were conducted for four node placement algorithms (AAC, MAC, ND, DIST which have been discussed in Section 3.3) for each network. Various scenarios with network loads of 40%, 50% and 60%; and Maximum link loads of 60% to 90% were simulated. Note that the network load cannot be higher than maximum link load. Also in cases where the network load and maximum link loads were close (which would result in near-homogeneous working capacities in the network), some demands could not be

routed because many links were already at their maximum loads. In such cases, new demand generation was stopped when either the network reached the desired load or 20% of total demands were blocked. We verified that in such cases the achieved network load was greater than 90% of the desired network load.

Comparison of Node Placements Performance

Our simulations results were obtained for each of the four node placement methods (AAC, MAC, ND and DIST) that we described in Chapter 3.

Table 4-1: Fixed-capacity Restorability of different node placement algorithms

Approximate Network Load	Maximum Link Load	Average Network Restorability	R(AAC)	R(ND)	R(MAC)	R(DIST)
40%	60%	All Graphs	0.9094	0.9124	0.9084	0.9030
		w/o Topo. constraints	0.9721	0.9675	0.9692	0.9627
40%	80%	All Graphs	0.8854	0.8879	0.8843	0.8770
		w/o Topo. constraints	0.9697	0.9641	0.9656	0.9604
50%	50%	All Graphs	0.9613	0.9754	0.9651	0.9733
		w/o Topo. constraints	1.0000	1.0000	1.0000	1.0000
50%	70%	All Graphs	0.7356	0.7292	0.7311	0.7059
		w/o Topo. constraints	0.8324	0.8130	0.8193	0.7947
60%	60%	All Graphs	0.6423	0.6502	0.6448	0.6471
		w/o Topo. constraints	0.6671	0.6655	0.6673	0.6652
60%	70%	All Graphs	0.5353	0.5275	0.5332	0.5133
		w/o Topo. constraints	0.6115	0.5886	0.6010	0.5753
60%	90%	All Graphs	0.5066	0.4950	0.5011	0.4664
		w/o Topo. constraints	0.6007	0.5726	0.5898	0.5515

Table 4-1 shows the restorability of our method under various network load and maximum link loads. Results were separately reported for all graphs and for those graphs where full restorability with HP-tree was feasible if enough spare capacity existed (referred in the table as w/o topological constraints). The reason for this separation is to evaluate the performance of the HP-tree in making the best use of the spare capacity without influencing it with the topological constraints of the HP-tree. The network sizes ranged from 10 nodes to 80 nodes and from 13 edges to 175 edges, average nodal degrees from 2.6 to 5, and minimum nodal degrees from 2 to 4.

Note that while a network load of, for instance, 50% in this scenario meant that an equal amount of working and spare capacities existed in the network, but spare capacities were not necessarily presented at the locations they were needed for backup paths. This is in contrast to the results presented in Chapter 3 where the spare capacities were assigned as needed on each link. As a result while in the simulations in Chapter 3 we were able to achieve full restorability with about 85% redundancy on average, here full restorability in some networks with network load of 40% was not possible. Also, it is obvious that if the network load is equal to maximum link load, the routing model that we described will result in near-homogeneous working link capacities. In such case in the homogeneous total link capacity model, the network restorability would simply become

$$R=\frac{\min\{c_p, c_w\}}{c_w}=\min\{1, \frac{1-\frac{c_w}{c_p+c_w}}{\frac{c_w}{c_p+c_w}}\}=\min\{1, \frac{1-U}{U}\}$$, with U being the maximum link utilization, as table 4-1 indicates for the 50%-50% and 60%-60% cases (restorability of 1.0 and 0.66, respectively). The results for these two obvious cases were reported for studying the impact of topological constraints (identified in the 'All Graphs' rows.)

When the impact of HP-tree topological constraint (which the ND and DIST method can avoid easier than capacity-based AAC and MAC methods) was eliminated, the AAC method was able to produce better restorability performance in comparison with other methods because this method is designed specifically to have higher priority for protecting links with largest working capacities. The DIST placement method fared poorly, confirming our previous statement that in a fixed-capacity pre-designed scenario,

the arrangement of tree links based on protection capacities has a more significant impact on restorability than the length of the backup paths.

Table 4-2: Comparison of the capability of different methods to achieve the highest Restorability (All graphs)

Total Network Load	Maximum Link Load	AAC	ND	MAC	DIST
40%	60%	37.77% of cases	39.55% of cases	31.35% of cases	22.21% of cases
40%	80%	36.89% of cases	36.89% of cases	31.46% of cases	18.49% of cases
50%	50%	27.97% of cases	62.65% of cases	36.61% of cases	55.38% of cases
50%	70%	41.86% of cases	26.31% of cases	32.57% of cases	9.38% of cases
60%	70%	38.91% of cases	26.59% of cases	35.51% of cases	10.95% of cases
60%	90%	43.88% of cases	23.37% of cases	32.57% of cases	8.28% of cases

Overall we found that, as shown in Table 4-2, the capacity-based AAC and MAC node placement methods were able to provide superior restorability among the four methods as the link utilization increased. DIST method often achieved the lowest restorability, in contrast to its superior performance in the SCA scenarios in Chapter 3. Note that the summation of the percentages in each row in Table 4-2 is larger than 100%, as often two or more methods were both able to achieve the best restorability and as a result such cases were counted twice or more (one for each method). Also note that when the total network load and maximum link load were identical (the 50%-50% case in Table 4-2), thus forcing near-homogeneous working capacities on links, the advantage of capacity-based methods disappeared because in the homogeneous working link capacity case, the capacity-based node placement methods cannot distinguish between nodes.

As results in Table 4-1 also indicate, the difference in restorability performance is higher under heavier network loads where the scarcity of the spare capacity in the right location improves the performance of capacity-based node placement methods.

Impact of Network Topological Characteristics

In this section we study the trend of restorability effectiveness of the HP-tree algorithm versus various topological parameters of the pre-designed fixed-capacity network. Note that the impact on restorability in this case comes from two separate sources: the impact of topological constraint of the HP-tree, and the effectiveness of the HP-tree to manage backup paths through the available spare capacity of the network. These two effects were demonstrated in the results presented in the previous section in Table 4-1. The general impact of topological constraints has already been studied under the SCA scenario in Section 3.5.3 where we showed how the restorability of the HP-tree would change with various topological parameters of the network. So in this section we limit our study to the second issue; i.e. the trend of restorability of the HP-tree algorithm in a pre-designed network that is free from the topological constraints. For this reason we use a subset of graphs for which the HP-tree could have had achieved full restorability if enough spare capacity had been available on the network links. We call this set the *SCA Fully Restorable* graphs in our database. The rest of the results in this section have been compiled from this set of random graphs.

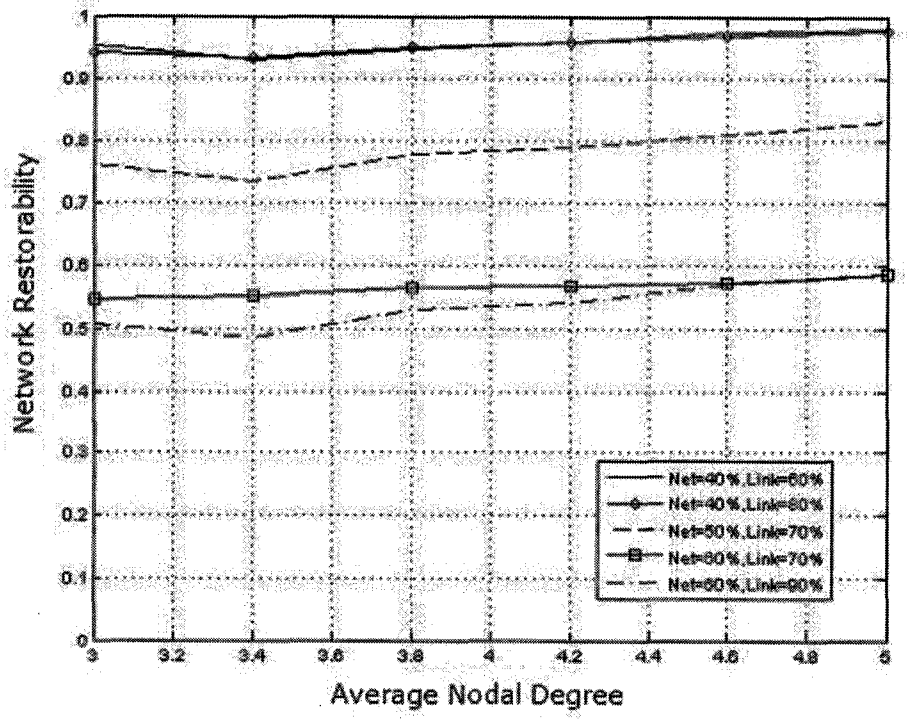


Figure 4-2: Restorability of HP-tree in pre-designed network versus nodal degree

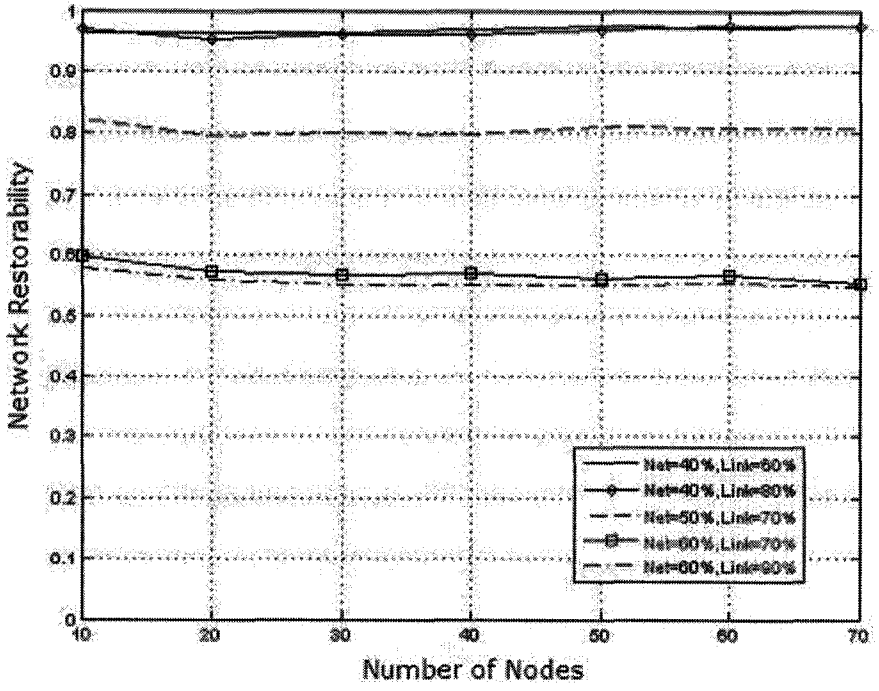


Figure 4-3: Restorability of HP-tree in pre-designed network vs. number of nodes

Figure 4-2 indicates the impact of average nodal degree on the restorability in a pre-designed fixed-capacity network for graphs that are SCA fully restorable. Results for various network load (indicated as *Net=%*) and maximum link load (indicated as *Link=%*) have been shown. Each point on the curve is the average value of networks with the same average nodal degree in the same load scenario. Points on the each curve were connected by straight lines. We recall from Section 3.5.2 that an increase in average nodal degree sharply increased the achieved restorability in the SCA scenario; reaching close to full restorability for all networks of average nodal degrees of 3.8 and higher with no 3-node chains. For SCA fully restorable graphs (those achieve full restorability with HP-tree under SCA scenario) the average nodal degree only slightly improves the restorability in a pre-designed fixed-capacity network scenario. This result further validates our expectation that in pre-designed scenario the arrangement of spare capacities on the HP-tree plays a more important role than topological parameters of the network. The 95% confidence intervals for the results in Figure 4-2 varied based on the nodal degrees: from ±6-14% of the mean values at each point at D=3, to ±1.5% of the mean values at each point at D=3.8 and higher.

As observed in Figure 4-3 the impact of the number of nodes in the network on the achieved restorability in SCA fully restorable networks in the pre-designed fixed-capacity scenario is minimal. Each point on the curve is the average value of networks with the same average nodal degree in the same load scenario.

Table 4-3: Restorability versus minimum nodal degree

Total Network Load	Maximum Link Load	D_{min}=2	D_{min}=3	D_{min}=4
40%	60%	0.9552	0.9687	0.9791
40%	80%	0.9455	0.9690	0.9820
50%	70%	0.7960	0.8079	0.8169
60%	70%	0.5740	0.5759	0.5757
60%	90%	0.5588	0.5559	0.5679

Table 4-3 shows the achieved restorability in SCA-fully restorable graphs where again a small increase in network restorability versus minimum nodal degree was noticed. However when the network load increased, the effect of minimum nodal degree was not noticeable. The confidence interval for the results in Table 4-3 was within ±1.7% of the average value at each point.

Impact of Network Load and Maximum Link Load

Tables 4-1 and 4-2, and Figures 4-2 and 4-3 also provide insights into the achieved restorability with HP-tree when network load is changed. The utilization of the network directly impacts the availability of spare capacity and thus, the restorability of the network. In our results the restorability went down from about 90% (97% for SCA fully restorable graphs) for network load of 40%, to about 73% (82% for SCA fully restorable graphs) for network load of 50%, and 65% (67% for SCA fully restorable graphs) when the average network load is increased to about 60%. Recalling from the results of the SCA scenario in Chapter 3, full restorability at network load of 60% and up (Capacity Sharing factor of 1.5) was achievable if spare capacities were assigned based on the SCA design; i.e. were they were needed. However in this scenario of leftover spare capacity from a fixed total link capacity, only half to two-third of working demands could be restored through HP-tree if the network load was 60%.

The maximum link load also plays a role here. As the results indicate, an increase in maximum link load, which results in unbalanced working capacities on network links, decreases the restorability slightly. This result could be related to the concept of forcers [ShGr03] that indicated the impact of the existence of links with large working capacities compared to others. Our own results in Section 3.5.2 also indicate that balanced or homogeneous working capacities across network reduces the redundancy requirements, and thus a higher restorability level could be achieved with less spare capacity. Our result here quantifies the trend in the specific case of HP-trees in a pre-designed network.

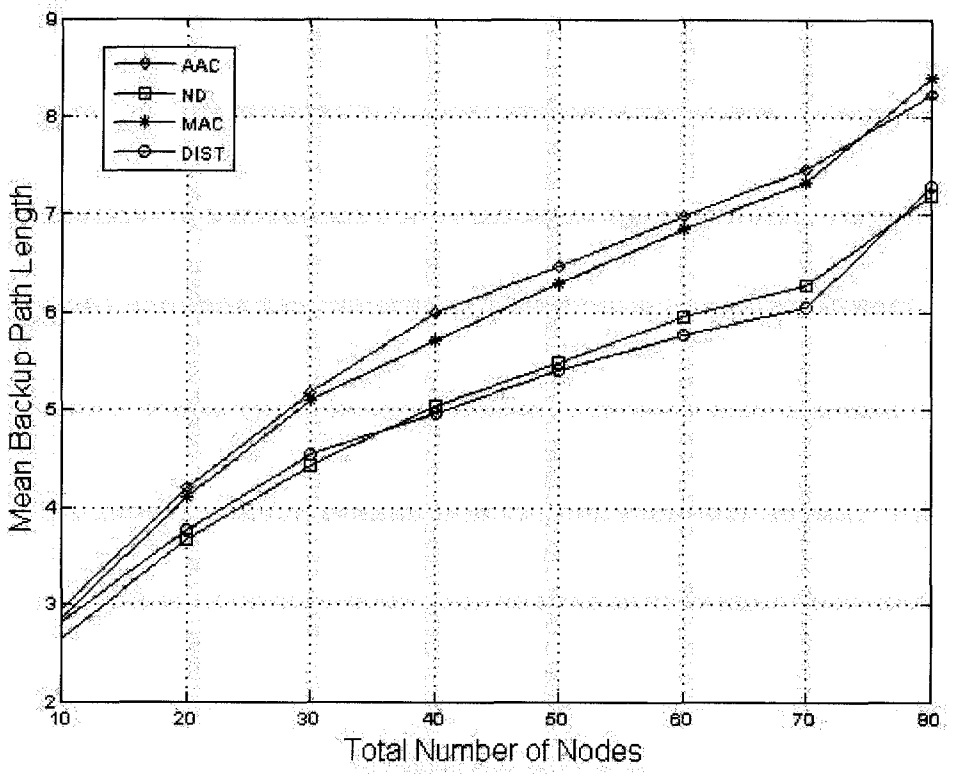


Figure 4-4: Mean Backup Path length (hops) for network load = 50%, max link load=70%

Average Backup Path Length Results

Figure 4-4 shows the average backup path length in number of hops for the four node placement methods. Our results did not indicate a significant change of mean backup path length values with network load or maximum link load. Therefore, the results here are shown for one scenario only. The results are consistent with those achieved in the SCA scenario in Section 3.5. The two topology-based methods (ND and DIST) achieve backup path lengths about 10% shorter compared to capacity-based methods (AAC and MAC). The algorithm achieves an average backup path length of about 7 hops for a network of 80 nodes.

4.3.2. Real network Scenarios

In the real network scenario we used a number of networks based on available commercial network topologies from [Mapnet]. The complete topology vectors for these

graphs have been provided in Appendix A. Each topology file contains a list of all network edges (identified by their two end nodes) and their total capacity.

Table 4-4:Characteristics of Real Network Topologies

Name	M	N	D_{min}	D_{max}	Max-to-Min Link Capacity	CSF
ATT2	66	31	2	12	242	0.9126
BBN	19	12	2	5	8	0.9091
Brazil RNP	19	11	2	10	8.33	0.9138
Data xchg	24	8	3	7	3	2.1212
Goodnet	54	23	2	17	6	1.3094
Gridnet	25	10	4	6	2	1.6000
Qwest	32	14	2	8	16	1.1835
Sprint	34	18	2	8	96	0.9158
UUnet	119	48	2	16	208	1.0816
XOnet	14	9	2	4	18	1.0109

Table 4-4 lists important parameters of the simulated networks. Each vector includes the total capacity of each link in the network. The networks range in size from 14 to 119 edges, with average nodal degrees between 3 and 6. All graphs in Appendix A are two-edge connected and none include any 3-node chain. The HP-tree SCA algorithm of Chapter 3 can provide full restorability to graphs in Table 4-4 if spare capacity could be assigned as needed. The achievable CSF using HP-tree is also provided in Table 4-4.

In this scenario we will use the homogeneous link load model (the homogeneous link capacity model was used in Section 4.3.1). In this model the link utilization U_{link}, defined as the ratio of working capacity on each link to the total link capacity, is homogeneous across the network. This model is realistic for those pre-designed scenarios where network target load is close to individual link utilization limits; for instance a scenario with target network load of 50% and maximum individual link utilization limits of 50%-60%. The routing algorithm would find and capacitate potential paths until the utilization limit on a link is reached, and because the link limit is close to the overall

network target load, the resulting working capacities tend to be all near their maximum utilization limits in this model.

Performance Evaluation

We performed simulations on the networks in Table 4-4 by varying the link utilization parameter from about 30% to about 70%. The remaining capacity on the link was left for protection. We calculated the achievable restorability of HP-tree for the given fixed amount of link spare capacity in the network at each load level. We also computed the average backup path length in each case.

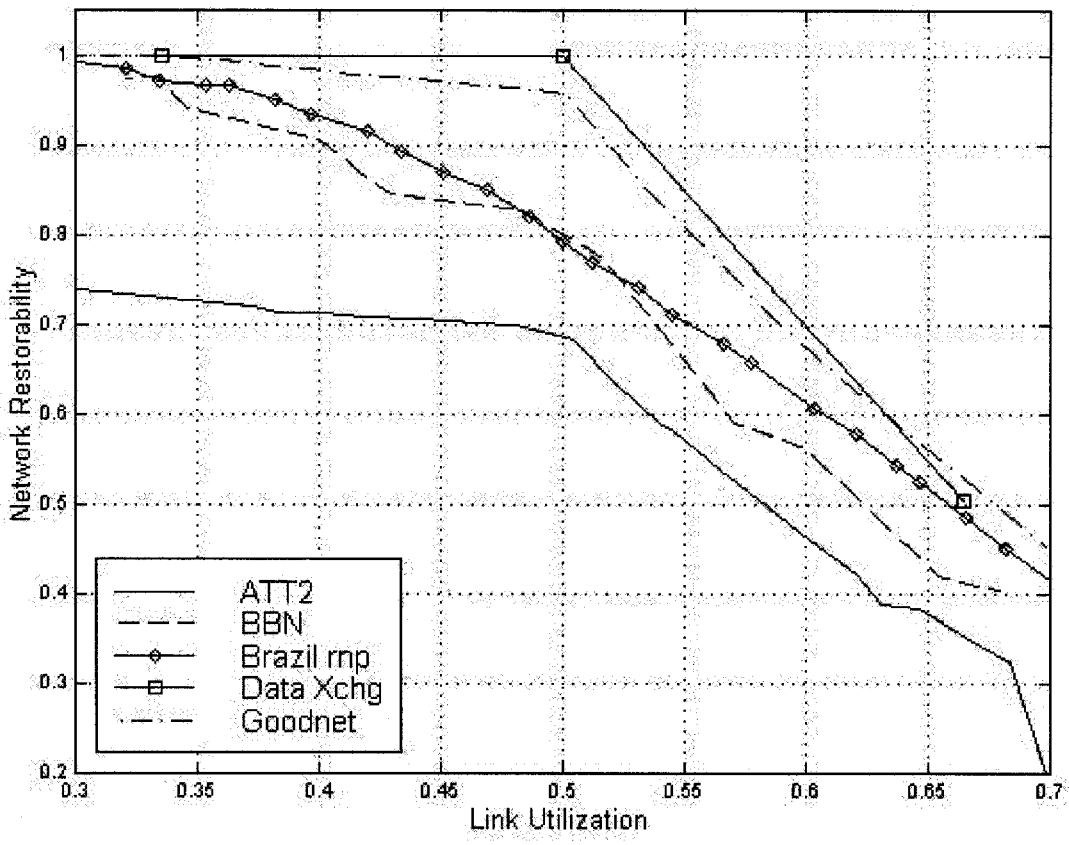


Figure 4-5: HP-tree restorability in fixed capacity real network scenario

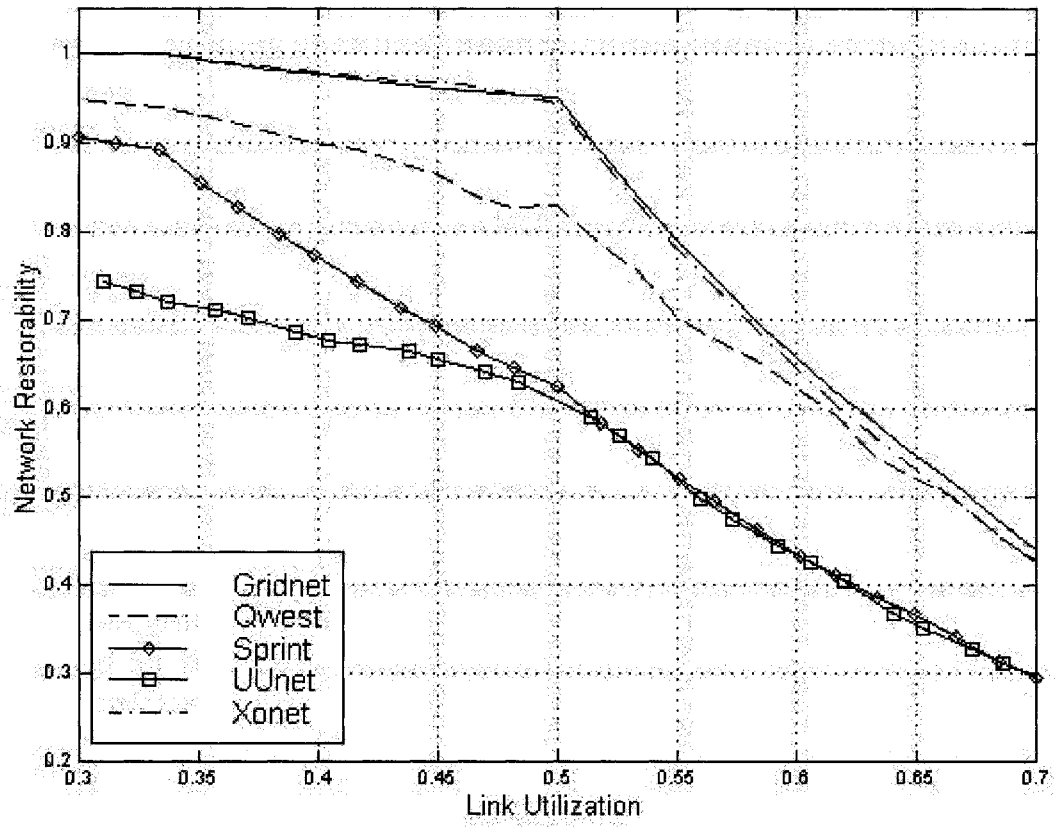


Figure 4-6: HP-tree restorability in fixed capacity real network scenario (continued)

Our results showed that the restorability in pre-designed networks depended heavily on the distribution of working capacities in the network, which could be characterized by the max-to-min working capacity ratio; i.e. the ratio of the maximum link capacity to the minimum link capacity in the network indicated in Table 4-4. As shown in Chapter 3 of this thesis, non-homogeneous networks require more redundancy for full restoration. In pre-designed network where the required redundancy is restricted, network restorability decreases.

Figures 4-5 and 4-6 shows the restorability results for the networks in Table 4-4. It is obvious that those networks with largest max-to-min link capacity ratios (respectively ATT2, UUnet and Sprint) had the lowest restorability. On the other hand near-homogeneous networks (Gridnet and Data Xchg) achieved the highest restorability. The max-to-min link capacity ratio appeared to have the most impact on the network utilizations of 50% and lower where the restorability of simulated networks with HP-tree

varied significantly, depending on the link capacity ratio. The HP-trees in those networks with lower max-to-min ratios were able to maintain a high degree of network restorability for larger network loads. However as network utilization was increased to 70% levels, the performance of HP-tree for different networks was closer to each other; often within the restorability range of 40% to 50%.

In terms of node placement methods, the AAC method achieved the best restorability in about 68% of 250 scenarios (10 networks under 25 different network utilization scenarios), followed by the MAC method that achieved the best result in almost 22% of the cases. The ND method achieved the best results in just about 10% of the cases, and the DIST method (which was the most efficient in the SCA scenario in Chapter 3) was never able to achieve a better restorability than the other methods in these fixed capacity cases.

Table 4-5: Mean Backup Path Length for Real Network scenarios

Name	*M*	*N*	Radius	Mean Backup Path Length			
				AAC	ND	MAC	DIST
ATT2	66	31	3	3.5346	2.8903	2.9727	3.0636
BBN	19	12	3	3.3333	2.8947	2.8947	3.0526
Brazil RNP	19	11	1	2.0000	2.0000	2.0000	2.000
Data xchg	24	8	1	2.9583	2.0000	2.0000	2.0000
Goodnet	54	23	2	3.3889	2.3333	2.8889	2.4074
Gridnet	25	10	2	3.0417	2.5417	2.8400	3.3200
Qwest	32	14	2	2.9375	2.5313	2.5938	2.5313
Sprint	34	18	3	3.7333	2.9118	3.4118	3.9394
Uunet	119	48	3	3.6555	2.7395	3.0672	3.2185
Xonet	14	9	2	2.6429	2.6429	3.4615	3.5385

Table 4-5 shows the average backup path length (in number of hops) achieved by each node placement method. In general, the average backup path length for each network did not change significantly with link utilization or restorability. As expected, the ND method achieved the best backup path length and the AAC method resulted in the longest backup

paths. However, overall the average backup paths for real networks were within an acceptable range. Even for large networks with 119 edges, the average backup path length remained around 3 hops. This result could be attributed to relatively high average nodal degree in those backbone networks.

4.4. Approximate Analysis of Restorability in Special cases

Before we close this chapter, we present an approximate random analysis of restorability in a special case of pre-designed random mesh networks where the link working and spare capacities could be represented by identical uniform distributions. This analysis is intended to provide an approximate estimate of how restorability of the HP-tree changes with various network parameters if the random model that we described in Section 4.2 was used. We first present our methodology, and then details of our analysis, followed by simulations to validate the results.

The following are the basic assumptions in this scenario:

1. The network is a simple undirected, 2-edge connected graph.
2. Each link in the network is protected by a single protection path between its two end nodes provided by the HP-tree.
3. We consider here the SCA-fully restorable case (see Section 4.3.1), assuming that topological constraints of the network have been eliminated by proper topological design.
4. The network has a predetermined amount of available protection capacity on each network link
5. We use the random capacity model for our approximation; i.e. assume that the amount of working and protection capacities on network links can be represented by independent random variables with known probability density functions. This approximation is used to allow statistical representation and analysis of restorability.

4.4.1. Analysis

Let us first develop a general framework for the random capacity model described in the above. From Equation 2-4, the restorability can be calculated as following:

$$R = \frac{\sum_{1}^{M} c_{wpi}}{\sum_{1}^{M} c_{wi}} = \frac{M \times \overline{C}_{wp}}{M \times \overline{C}_{w}} = \frac{\overline{C}_{wp}}{\overline{C}_{w}} \tag{4-1}$$

In an HP-tree link protection structure, a protection path is assigned for each link in the network. Let random variable $C_w(h)$ denote the amount of protected working capacity of a link for which the backup path is h hops long. If F_{wp}, F_w and F_p are the cumulative distribution functions for random variables that respectively represent protected working capacity, working capacity and protection capacity on a link, and f_{wp}, f_w and f_p are the respective probability density functions for the same random variables, then the following equation can be derived for the random model we described at the beginning of Section 4.4:

$$\Pr(C_{wp}(h) > x) = \Pr(C_w > x) \times \left[\Pr(C_p > x)\right]^h \quad \Rightarrow$$

$$(1 - F_{wp}(x \mid h)) = (1 - F_w(x)) \times (1 - F_p(x))^h \quad \Rightarrow$$

$$f_{wp}(x \mid h) = f_w(x)(1 - F_p(x))^h + hf_p(x)(1 - F_w(x))(1 - F_p(x))^{h-1} \tag{4-2}$$

By using the above expression in random capacity model, we can compute an average value for protected working capacity conditioned on backup path length distribution, and by approximating the backup path length with a random variable with uniform distribution, the restorability could be computed from Equation (4-1). Here we will approximate it for a simplified case in which the random variables representing the working and protection capacities are identically distributed between c_{min} and c_{max} uniformly (thus also assuming a 50% network load), then $f_w = f_p = \frac{1}{c_{\max} - c_{\min}}$, $c_{\min} \le x \le c_{\max}$. By substituting in Equation (4-2) we will have:

$$f_{wp}(x \mid h) = \frac{h+1}{(c_{\max} - c_{\min})^{h+1}}(c_{\max} - x)^h \quad , \quad c_{\min} \le x \le c_{\max} \tag{4-3}$$

$$\overline{C}_{wp}(h) = \int_{c_{\min}}^{c_{\max}} x f_{wp}(x \mid h)dx = \frac{c_{\max} + (h+1)c_{\min}}{h+2} \tag{4-4}$$

In this model the backup path length could further be approximated by a random variable with uniform distribution, as non-tree links in a protection tree might be present between any two nodes on the tree with equal probability. We also further approximate the integer variable h with a continuous variable so that we could compute an integral instead of series. With this assumption the average protected working capacity could be computed as following:

$$\overline{C}_{wp} = \int_{h_{\min}}^{h_{\max}} \frac{c_{\max} + (h+1)c_{\min}}{h+2} \frac{1}{h_{\max} - h_{\min}} dh = \frac{c_{\min}}{(h_{\max} - h_{\min})} \int_{h_{\min}}^{h_{\max}} \frac{F + h + 1}{h+2} dh$$

$$= \frac{c_{\min}}{(h_{\max} - h_{\min})} \int_{h_{\min}}^{h_{\max}} (1 + \frac{F-1}{h+2}) dh = c_{\min} \left[1 + \frac{F-1}{h_{\max} - h_{\min}} \ln \frac{h_{\max} + 2}{h_{\min} + 2} \right] \tag{4-5}$$

In which $F = \frac{c_{\max}}{c_{\min}}$ is the max-to-min capacity ratio. The model in this section used identical distributions for working and spare capacities, so F is the same for both working and spare capacities. Furthermore, for a protection tree in a simple graph with no parallel edges, the value of h_{min} is always two hops. In Chapter 3 we showed that the maximum backup path length in the HP-tree had a logarithmic relationship with the number of nodes close to $2*(log_2 N)$. Then we could write:

$$R = \frac{\overline{C}_{wp}}{\overline{C}_w} = \frac{c_{\min}}{\frac{c_{\max} + c_{\min}}{2}} \left[1 + \frac{F-1}{h_{\max} - h_{\min}} \ln \frac{h_{\max} + 2}{h_{\min} + 2} \right]$$

$$\Rightarrow R = \frac{2}{F+1} \left[1 + \frac{F-1}{2\log_2 N - 2} \ln \frac{\log_2 N + 1}{2} \right], \quad \text{N>2} \tag{4-6}$$

We note the limitations of this approximation, primarily the fact that it has been computed for a special case of identical uniform distribution for working and spare capacity and 50% load, as well as simplifications with regard to backup path length. The result in (4-6) ignores the important impact of the average nodal degree of the network. However, the approach could be extended for other distributions or more generalized assumptions. If the working and protection capacities in a network resemble a random variable with uniform distribution, then (4-6) could provide us with a useful

approximation for network restorability in such pre-designed network with respect to min-to-max capacity ratio, F, and network size, N, allowing us to estimate the restorability in large networks for which accurate computational methods may not be feasible.

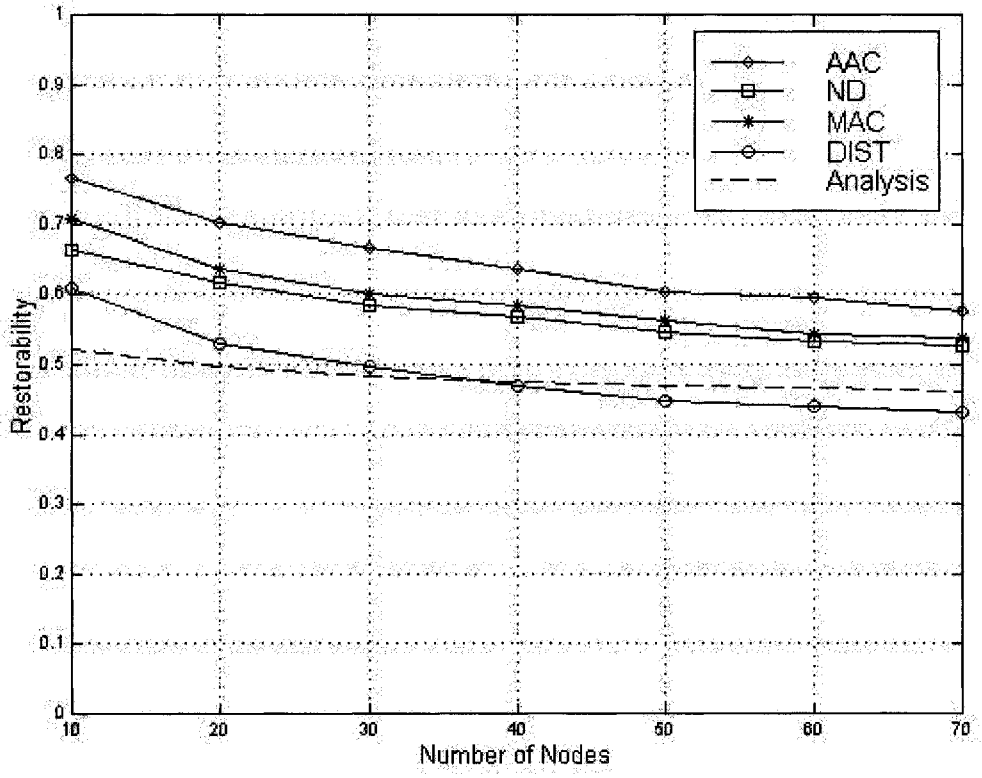


Figure 4-7: Approximate random capacity results for HP-tree restorability vs. Number of nodes (F=6)

In order to verify the applicability of the approximate solution of (4-6), we applied it to the HP-tree results for a range of networks built with the random capacity model. Those results along with the approximate analytical results are shown in Figure 4-7. The results in this figure were obtained for F=6. The approximate curve underestimates the restorability, but follows the trend of the simulation results. The simulation results and the approximate results in Equation 4-6 indicate that in a mesh network where link capacities can be approximated with a random variable with uniform distribution, the restorability decreases with network size, approaching $2/(F-1)$.

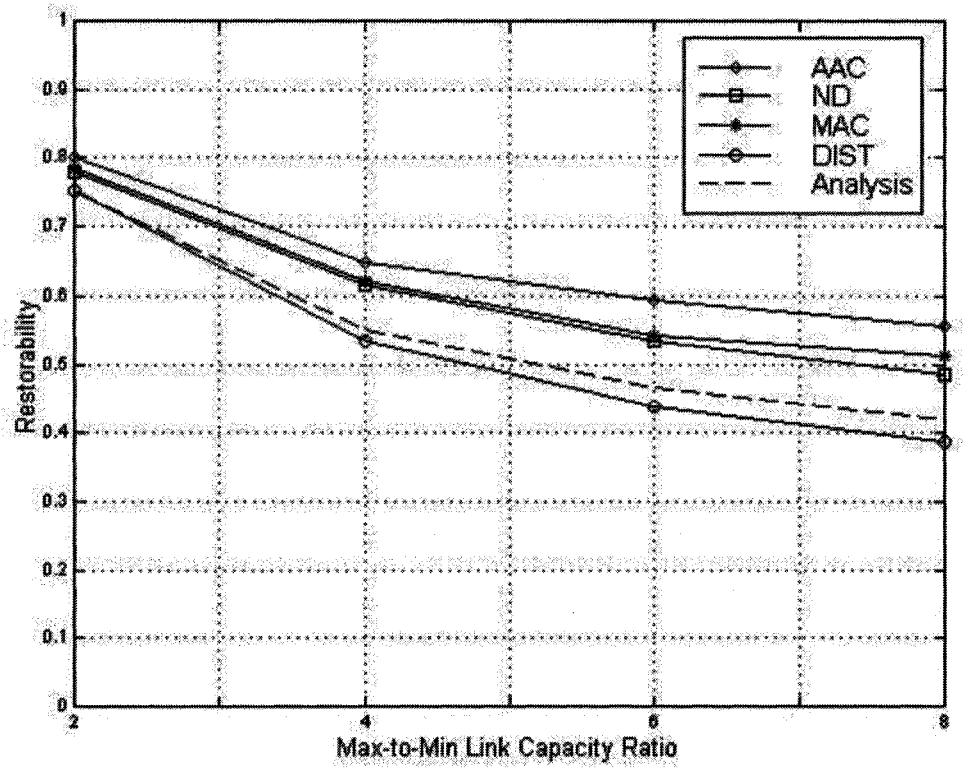


Figure 4-8: Approximate random capacity results for HP-tree restorability vs. Max-to-Min link capacity ratio (N=60)

Figure 4-8 provides the curve of restorability with respect to the max-to-min link capacity ratio *F*. Here also the restorability estimates using the approximate analysis provided an accurate trend of the change in restorability.

4.5. Concluding Remarks

In this Section we presented a design scenario in which the network topology and the amount of available working and spare capacity on each link were limited and pre-determined. Our novel contributions are, to the best of our knowledge, (1) Suggesting different network models for the pre-designed fixed-capacity scenario; (2) Modifying our HP-tree design algorithm for this purpose, and (3) presenting simulation results and analytical approximations for various network models that were suggested in this chapter.

Chapter 5. Multiple Link Failure Restorability of HP-Tree

Having considered and analyzed the tree-based restoration of single failure scenarios, we now move onto the multiple link failure scenarios in which the performance of the protection scheme under a sequence of independent network link failures is analyzed. We present the problem statement in Section 5.1, discuss various multiple failure models in Section 5.2, then proceed with a description of our algorithm for computing multiple failure restorability and redundancy requirement in Section 5.3. Performance results are presented in Sections 5.4 and 5.5.

5.1. Background and Problem Statement

In discussing multiple-failure scenarios we use the following common assumptions that have been used widely in similar studies:

1. That concurrent multiple failures occur when a subsequent failure hits before the previous failure has been repaired [ZhZh04].
2. That the working traffic has been completely re-routed to its backup path before the second link failure occurs [ScGr04]. Even though considering the self-repairing mechanism of the HP-tree (as discussed in Section 2.6), this assumption may not be required, however it simplifies our multiple failure models.

In order to specify the boundary of the problem we will be addressing in this chapter, we will first discuss various multiple failure categories and the placement of our problem within each category.

Multiple link failures for backbone networks can be categorized according to the root cause and impact of failure. Failures can be categorized as fundamental or algorithmic failures [LuMe01a]. Fundamental failures are closely related to limitations of the topology of the network, and essentially no algorithm can fully recover from them. Algorithmic failures are related to the rerouting technique that each specific algorithm employs for recovering from failure. Fundamental failures can be further categorized into following [LuMe01a]:

- *Disconnection* failures, where a set of consecutive cuts partitions the graph.

- *Network Capacity* failures where a second failure independent path for restoration from a subsequent failure could not be found.

Disconnection failures are directly related to edge-connectivity of graph as discussed in Chapter 2. In order for a network to be able to recover from every possible k-link failure scenario, the network must be at least (k+1)-edge connected. A necessary condition, but not sufficient, is for the minimum nodal degree of the network to be at least k+1. In case of recovery from double link failures, the network will have to be at least three-edge connected.

We could also add *Path length* failures to the above category, which had been listed by [LuMe01a] under algorithmic failures. This type of failures is related to the case where the length of the backup paths is too long to be feasible. Note that while some protection schemes can be designed with path length limits [HeBy94, KoSa04], it is never possible to design a protection structure for an *arbitrary* desired path length limit because a backup path with the desired length may not exist. An example is a link restoration mechanism in a ring of N nodes where backup paths could be as long as N-1 hops. Therefore, the path length limits must be *feasible*. Also note that even if a network might be able to provide single failure backup paths for a feasible path length limit, such backup paths might still be infeasible for double- or multiple-failure scenarios in that same network. The path length failure categorization would cover all these scenarios.

In studying the performance of the HP-tree, we focus on the algorithmic failures because fundamental failures are common to all algorithms and cannot be handled by mere algorithm improvements. Algorithmic failures are related to the ability of the algorithm to pick edge-disjoint backup paths for handling consecutive link failures in a pre-planned fashion. Various sub-categories of algorithmic failures are as following [LuMe01a, KiLu03]:

- *Path hit* failures where combined impact of the hits would prevent restoration of the traffic through pre-planned backup paths, by failing both a link and its backup path.
- *Blocked path* failures where the protection capacity for recovery of the second failure has already been used up for restoration of the first failure.
- *Topological Constraint* failures where certain topological characteristics of the network might prohibit creation of the protection structure for double- or multiple-

failure restoration. Examples include ring cover design [LuMe01a], and our HP-tree topological constraints as discussed in Section 2.6. For instance, a failure that creates a 3-node chain in an HP-tree protected network would leave at least one link vulnerable to a second failure, as discussed before.

In studying multiple failure restorability, typically every combination of link failure scenarios must be considered. Also note that in dynamically reconfigurable restoration schemes the order of failures is important here too; i.e. failure scenario of (e_1, e_2, .., e_k) is different from (e_2, e_1, .., e_k), because depending on which failure is being restored first, the state of the available protection capacity on the network links would change. For a network size of M edges in a study of k subsequent failures, a total of $\frac{M!}{(M-K)!}$ different scenarios must be analyzed.

It is our objective to provide models and algorithms for performance study of HP-trees in multiple failure scenarios. In the next section we proceed with methodology and failure recovery models that will be studied here. Section 5.3 will provide a definition of multiple failure restorability and an algorithm to compute it. Performance results will be presented and discussed in Section 5.4.

5.2. Methodology and Models

5.2.1. Working-only and All-capacity Restoration Models

In general a restoration scheme may restore either only the working channels of the network, or both working and protection channels when they are already in use [ScGr04]. The difference could be explained in the context of guaranteed quality of service in the following manner. In the case of working-only protection, only the working capacity of a link is restored if hit with failure. Therefore if a working channel is re-routed to the protection capacity of the network and then a second failure hits it, the channel will not have the privilege of a second restoration. In other words in this scenario the network could survive multiple failures if they do not affect the same working traffic more than once.

On the other hand in the all-capacity protection case, those working capacity units that have been re-routed to spare channels will also be restored in case of a second failure. We

shall present results for both scenarios in this chapter. In order to model an all-capacity protection scenario, every time the working capacity of a link is restored through a backup path, we add it to the working capacity of every segment of the backup path while subtracting an equal amount from the spare capacity of each segment; in other words, we shall re-assign those spare channels as working channels on each segment. In the working-only protection scenario, the re-routed working capacity will be subtracted from the spare capacity on each segment, but will not be added to the working capacity.

5.2.2. Static and Dynamically Reconfigurable Protection Schemes

Restoration schemes could be static or dynamic (also called reconfigurable). In the static case, the backup paths are usually pre-determined and the network does not reconfigure its protection architecture to create new backup paths. There could be several scenarios where this model would apply [Schu03]:

- Where the network administrators would like to avoid reconfigurations because of signaling and administrative overhead, prohibitive cross connect switching times or other reasons.
- Where specific reconfiguration of a backup path is not possible because of lack of spare capacity or other constraints.
- Where the reconfiguration after a first failure has not been completed yet.

In the static case, recovery from multiple failures is only possible in cases where the backup paths for consecutive failures are edge-disjoint (or *spatially-independent* [ClGr02]). Obviously, a fully static working-only restoration scheme can never achieve full multiple failure restorability because it cannot recover from a scenario where both a link and its backup paths have been hit consecutively. In a static case, the multiple failure restorability depends on *sharability* of network links, which is defined as the number of links that share the same link in their backup paths [ZhZh04]. As shown in Chapter 2, sharability can be directly computed from the Restoration Paths Matrix. Maximum allowed sharability *(MAS)* could be included in the objectives of the protection design, it was shown in those studies that while an increase in sharability would reduce the required network redundancy for protection against single failures, it also reduces the double-failure restorability of the network. As such, there is a tradeoff between redundancy and

double failure restorability for static cycle-based designs, which could be controlled by number and length of the cycles employed in the design [Schu03a]. The MAS-based design for the HP-tree is a future research objective.

Reconfigurable protection schemes are based on the assumption that there will be sufficient time between two consecutive failures to collect information about the current state of the network, compute new backup paths for the reconfigured protection architecture, and communicate information those paths to network nodes, all before the second failure hits the network. Fully adaptive schemes would require a complete view of the state of network links and nodes. The amount of signalling for collecting those data and for communicating the whole new set of backup paths to all network nodes might become prohibitive. However such approach would provide the most capacity efficient solution to multiple failure restorability.

Partial reconfiguration scenarios could be considered too where only part of the protection architecture would be modified. Examples of such reconfigurations include:

- Selection of a new backup path that has not been affected by the previous failures from a set of pre-determined backup paths.
- Addition of new protection cycles to the network after a failure recovery, while keeping the previous cycles unchanged.

The HP-tree should be classified as a partial reconfigurable scheme. The HP-tree can recover from multiple path hits as long as enough spare capacity exists and topological constraints do not block the backup paths. A detailed description of the self-repairing mechanism of the HP-tree was presented in Chapter 2. Note that the post-failure reconfiguration of HP-tree is minimal, because only one node of the network must switch its parent node after each failure. This simplicity of operation reduces the administrative overhead of HP-tree significantly, as all the signalling and switching of the backup paths are done just adjacent to the failure. On the other hand we expect to pay a price in terms of redundancy requirements, as repairing the protection tree with a non-tree link adjacent to the failure may not always be the most efficient solution.

5.3. Multiple Failure Restorability and Redundancy Computation Methodology

5.3.1. Definition of Multiple Failure Restorability

In Section 2.5.1 we discussed the difference between average link restorability and total network restorability, and explained why we believed the latter would provide better representation of restorability performance of protection schemes. The same argument can be applied to multiple failure restorability as well. Several prior works on double failure link restoration [Schu03, Schu03a, DoGr02, ScPr04] use the following definition for Restorability of a given double failure (i, j) [ClGr02]:

$$R(i,j) = 1 - \frac{L(i,j)}{c_{wi} + c_{wj}} \tag{5-1}$$

in which $L(i, j)$ denotes the total amount of unprotected working capacity of the two links e_i and e_j subject to double failures. They then proceed to compute average double failure restorability by averaging (5-1) over all possible $M(M-1)$ pairs of (i, j) links.

This method has the same advantages and drawbacks that we discussed for average link restorability in Section 2.5.1, namely the fact that it takes average between link pairs of different sizes, and thus cannot truly represent the amount of traffic restoration in the network. Although in the multiple failure case, the impact of the capacity variation is lower than single failure computation because here the link capacities are summed in pair. In order to have a more accurate measure of traffic restorability, in our work here we use the average network multiple failure restorability, which we compute by calculating total network restorability in the resulting graph post-failure of each network link, and then average the results. While this measure still averages results between networks of different topologies (result of removing one link at each stage), the approximation is still better than average link restorability as those resulting graphs only differ from the original graph in one link at a time. For double failure case, it is defined as following:

$$\overline{R}_2 = \frac{1}{M}\sum_{i=1}^{M} R(e_i) = \frac{1}{M}\sum_{i=1}^{M} \frac{\sum_{j=1, j\neq i}^{M} c_{pwj}}{\sum_{j=1, j\neq i}^{M} c_{wj}} \tag{5-2}$$

in which M is the total number of links and c_{pwj} denotes the restorable working capacity of link e_j after the failure of (e_i, e_j). The above formula could be easily extended for a multiple (k>2) failure case.

An alternative measure was also proposed in [ClGr02] as following:

$$\overline{R}_2 = 1 - \frac{\sum_{\forall(i,j),i\neq j} L(i,j)}{\sum_{\forall(i,j),i\neq j} c_{wi} + c_{wj}} \tag{5-3}$$

in which the ratio of the summation of all pair-wise losses to summation of all pair-wise working capacities is computed. In this work we use Equation 5-2 (and its generalization for k-failure case) to compute multiple failure restorability, because it is easier to implement using the Restoration Paths matrix and more scalable for generalization to k-failure case, while at the same time provides an acceptable measure of traffic restorability in a network.

5.3.2. Multiple Failure Restorability Computation for Static Schemes

In [ShYa04c] we presented a recursive method to compute multiple failure restorability using the restoration paths matrix for the case of static working-only restoration schemes. The algorithm was based on simple matrix operations such as removal of a column or row, and based on Equation 2-4. A description of the computation method follows:

```
Function Multiple_Failure_Static (Rp, Cw, Cp, Fc)
{
1. If Fc>1
      {
1.1   For i=1 to Sizeof(Rp)
          {
            // Assume link i is failed
1.2         Copy Rp to A, Copy Cp to B, Copy Cw to C
            //Check for links that are protected by link i
1.3         M = Sizeof(Rp);
1.4         For j=1 to M
            {
1.4.1             If A(j,i)>0
1.4.2                   □k0{1..M}, A(j,k)=0;
            }
            // Update the available protection capacity
1.5         For j=1 to M
            {
```

```
1.5.1              If A(i,j)>0
1.5.2              B(j)=max(B(j)-C(i), 0)
              }
1.6           {Remove Row i and column i from A}
1.7           {Remove vector element i from B}
1.8           {Remove vector element i from C}
1.9           Call Multiple_Failure_Static (A, C, B, Fc-1)
           }
    }
2. if Fc==1
       {
2.1    {Calculate Restorability R for this scenario from Equation (2-4)}
       }
}
```

The input parameters include Restoration Paths Matrix *Rp*, Working capacity vector *Cw*, Protection capacity vector *Cp*, and the number of failures, *Fc*. The function *Sizeof*( ) gives the number of row or columns of a matrix, whichever is greater. For $FC = k$, the algorithm examines each combination of failures by removing $k-1$ links recursively one by one. An amount equal to the working capacity of the failed link in that stage is subtracted from the remaining protection capacity of its backup path at each recursion. Then we compute network restorability for the last failure in each combination from (2-4). The computed restorabilities are kept in an array or written to a file, and the average and standard deviation of the values can be computed at the end once the algorithm is completed.

As mentioned in Section 5.1, computation of average multiple failure restorability would require examination of $\frac{M!}{(M-K)!}$ different failure scenarios, which is the number of times Step 3 for restorability computation would be repeated in our algorithm. As explained in Chapter 2, the computation of single failure network restorability would have a worst case computational complexity of $O(MN)$. Therefore for double failure restorability the computational complexity could be as high as $O(M^2N)$, and for triple failure restorability computation as high as $O(M^3N)$ in the worst case. For this reason, computation of average multiple failure restorability for large networks could take a significant amount of time and processing power.

The *Multiple-Failure-Static* function in the above is applicable to static working-only protection scheme for which the restoration paths matrix can be computed. As mentioned in Chapter 2 while the basic assumption for restoration paths matrix is that each link is protected by a single backup path, the idea could be extended to schemes that provide backup paths for individual bandwidth units on each link (such as [StGr00]). In such case the restoration paths matrix could be built by assigning one row/column for bandwidth units with the same backup path; i.e. dividing such links into parallel links. The restoration paths matrix has no requirement or pre-assumption about whether a graph is simple or includes parallel links. However if the matrix is constructed in this way, then the static algorithm in the above must be adjusted slightly because then each failure requires removal of several bandwidth units; i.e. more than one row and column from the restoration paths matrix. For such case one could use a conversion matrix to represent the relationship between each span and the parallel links associated with it in the restoration paths matrix. If the network has M edges and the resulting restoration paths matrix for this scenario is PHP, then the size of the conversion matrix would be MHP.

5.3.3. Multiple Failure Restorability Computation for HP-tree

As mentioned earlier in this chapter and explained in more detail in Section 2.6, the HP-tree is a self-repairing protection scheme that partially reconfigure itself upon a failure. At this point we only consider reconfigurations aimed at maintaining the connectivity of the HP-tree. Therefore a failure on a non-tree link would not require reconfiguration. If a tree link fails, the downstream node of the link would switch from its primary parent node to its backup parent node to preserve the connectivity of the HP-tree (for definition of a downstream node see Section 2.6.2). In case of multiple failures, this operation would be repeated again; i.e. the downstream node would have to keep trying to preserve HP-tree connectivity each time by switching to another adjacent parent node as long as such there is a non-tree link to an upstream node exists.

Because of this mechanism, the restoration path matrix for the HP-tree will change for each link failure as the backup paths for the links connected to the downstream node and its child subtree will change. Therefore, the method in Section 5.3.2 must include a call to update the restoration paths matrix. Such update would require the topology vector (or

alternatively, the adjacency matrix) as an input to the function call along with the restoration paths matrix; because the restoration paths matrix by itself does not have sufficient information for computing the changes in the HP-tree. To facilitate the update of backup paths, we use an HP-tree node table, *Vtable*, which includes the relationship between nodes and their primary and backup parents and essentially defines the HP-tree. Each row in the table includes the following information:

Vtable(i) = { *Node i,* *Primary Parent Node ID,* *Backup Parent Node ID*}

For each node that does not have a primary or backup parent node, -1 would be entered in the table. Therefore the root node row would look like {Root ID , -1, -1}. Also if the HP-tree algorithm cannot assign backup parent node to a node, -1 would be entered in its place. The *Vtable* is a direct output of our HP-tree design algorithm in Chapters 3 and 4. The restoration paths matrix for the HP-tree could be computed from *Vtable* and the topology vector or adjacency matrix. Now updating the *Vtable* for each tree link failure is trivial as following:

For failure on tree link i:

1. Determine from the topology vector the two end nodes of link *i*, call them v_1 and v_2.
2. From *Vtable* if v_2 is primary parent of v_1, mark v_1 as the downstream node v_{down}. Otherwise mark v_2 as v_{down}.
3. In the *Vtable*, replace primary parent node of v_{down} with backup parent node of v_{down}.
4. Compute new backup parent node for v_{down} using the HP-tree backup parent sub-algorithm in Section 4.2.2.
5. Remove link *i* from the topology vector.
6. Compute new restoration paths matrix from the updated *Vtable* and topology vector.

As discussed in Section 2.6, the HP-tree algorithm is able to repair the tree upon subsequent failure as long as the downstream node of the failed link has at least one non-tree link to an upstream node in the HP-tree (i.e. a node which is not located in the subtree of the downstream node). If no more backup parent node can be found in Step 4, the respective entry in *Vtable* will be replaced with –1, meaning that another failure on this last remaining tree link would be left un-restored. If in Step 3 in the above the backup parent of v_{down} was already –1, then upon failure of link *i* the node v_{down} is left without any

primary parent node and thus the HP-tree would be partitioned into two disconnected subgraphs. In such case the HP-tree could only provide partial restoration to network, restricted to those links whose end nodes are within the same partition.

In our computations while the input HP-tree to the multiple-failure algorithm is calculated using the spare-capacity assignment design of Chapter 3, the modification of the HP-tree upon subsequent failures are treated as fixed-capacity design because the total capacity of network links cannot be changed in between failures. Therefore in the sub-function for updating the restoration path matrix, we use the method in Chapter 4 for assigning backup parent nodes; i.e. based on the available spare capacity of the link that connects the current node to each of the adjacent nodes. This approach has been covered in Section 4.2.2.

5.3.4. All-Capacity Restoration Scenario

The all-capacity restoration model assumes that upon subsequent failures, the working traffic (that had already been transferred to spare capacity upon the first failure) would still be restored. In other words, the portion of the spare capacity that is now carrying the working traffic will be handled as working capacity. Therefore upon each failure the amount of working traffic on the failed link should be subtracted from the available spare capacity on each segment of the backup path, and it should also be added to the working capacity on each of those segments too. The following line (1.5.3) could be added in the corresponding place in the algorithm in Section 5.3.2:

```
...
1.5.3 C(j)=C(j)+C(i)
...
```

In Section 5.4 we will present and compare results of multiple failure restorability for both working-only and all-capacity restoration models described in the above.

5.3.5. Spare Capacity Requirement Computation

For a shared capacity link protection scheme the spare capacity requirement on link e_i for all single failures in the network can be calculated from the following equation:

$$c_{pr}(i) = \max_{j=1..M}\{c_w(j) \times r(j,i)\} \qquad (5\text{-}4)$$

in which c_{pr} is the minimum spare capacity required on link i for maximum achievable restorability, $c_w(j)$ is the working capacity of failed link j, and $r(j, i)$ is the (i, j) element of the restoration paths matrix. Note that $r(i, i)=0$.

For multiple failure scenarios each time a link is taken off the network to represent a previous failure, its working capacity must also be added to the minimum spare capacity requirement of the segments of its backup path. The modified parts of the algorithm would be as following:

```
1. If Fc>1
      {
...           ...
1.5           For j=1 to M
              {
1.5.1               If A(i,j)>0
                    {
...                       ...
1.5.4                     Cpr(j,Fc)= Cpr(j,Fc)+C(i)
                    }
              }
...           ...
1.9           B(j).req=3Cpr(j,Fc) over all Fc
1.10          Call Multiple_Failure_Static (A, C, B, FC-1)
1.11          Cp(j).req=max{Cp(j).req, B(j).req}
         }
   }
2. if Fc==1
      {
...   ...
2.2   {Calculate Cp(j).req from 5-4 for all j}
      }
}
```

The unchanged part of the code has been skipped in the above except around the changed area for clarity. A subfield called *req* was added to the spare capacity vector elements to hold the mount of required redundancy to achieve maximum restorability. Whenever the recursive algorithm reaches the restorability computation stage for a given scenario (Fc=1), the *req* subfield of the protection capacity vector Cp is also computed. During each of the Fc recursions the vector C_{pr} is used to hold the amount of working capacity that is being routed to backup paths and thus must be added to the minimum redundancy requirement of the segments of backup paths. The total amount from

rerouting of previous consecutive failures in each multiple failure sequence is computed in line 1.9. Then in line 1.11 the *Cp.req* is updated to reflect the largest spare capacity required in scenarios examined so far in the algorithm.

The redundancy requirement for full double- or multiple-failure restorability is much larger than the redundancy requirement for single failure restoration. The ratio of theoretical lower bounds of double failure redundancy to single failure redundancy is estimated at $\frac{2(d-1)}{(d-2)}$, with d being the average nodal degree of the network [Schu05]. Thus the lower bound for double failure redundancy is four times the single failure case if d=3, and twice if d=4.

Based on the above argument, it might be of interest to see if a fairly high partial restorability with HP-tree could be achieved by increasing the redundancy partially but not to the full double failure restorability requirement level. This scenario is called here the *partial double-failure restorability* case. In order to compute the double failure restorability versus network redundancy, we increment the redundancy of the network in steps from the minimum required for single failure restorability toward the values in the *Cp.req* vector that were computed previously. The increments are computed in the following manner:

```
1. Compute Cp = Minimum redundancy for single failure restoration
2. Compute Cpp=Cp.req for full double failure restoration
3. Compute Cn = (Rp')*Cw
4. Sort Cn in descending order; let I = index vector
5. For k= 1 to M:
  5.1.  Cp(I(k))=Cpp(I(k))
  5.2.  Compute R2(k) for the new Cp
```

In the above, redundancy requirements for maximum single and double failure restorabilities are computed in Steps 1 and 2. These two vectors would serve as initial and target values for link spare capacities. In our simulation model in each step of incrementing the spare capacity of the network, we upgrade the spare capacity of one link of the network from its initial value in *Cp* to the target value in *Cpp*. Therefore in a network of M links, M steps were used.

We determined the order of choosing links for each incrementing step based on the sharability of each link, which was computed in Step 3 of the above algorithm from the restoration paths matrix *Rp* and working capacity vector *Cw*. Recall from the definition of restoration paths matrix in Chapter 2 that the true elements (=1) on each column *j* in *Rp* indicate the links that are using link *j* in their backup paths. Therefore multiplying the transpose of *Rp* by working capacity vector *Cw* gives the amount of working capacity in the network that is being protected by each link, a measure that we refer to as the sharability of the link. In Step 4 the vector of sharabilities is sorted, and the index vector of the sort operation is used for incrementing of the network redundancy and the computation of partial restorability in Step 5.

The above approach provided a simple way to calculate the improvement in restorability when the redundancy is partially increased with a granularity of one link at a time. Note that the capacity increase could have also been allocated in other ways; for instance by solving an optimization problem to assign additional capacity on network links for maximum restorability, or by distributing the increments evenly on network links etc. However, we believe our approach of upgrading the link with the highest sharability offers a simple and practical method that, as we show later, provides significant improvement in restorability.

5.4. Performance Results

In this section, computational results of multiple failure restorability for HP-tree are presented. For double failure analysis of random topologies we used a subset of our database of random mesh graphs that were three-edge connected, with a total of over 3000 graphs. The graphs ranged in order from 10 to 70 nodes, 17 to 175 edges, with an average nodal degree of 3.4 to 5. Note that in a three-edge connected graph, the minimum nodal degree of a node will be three of larger. For our study of triple failures, we used a subset of more than 700 four-edge connected graphs.

For demand routing, a random demand model was used except where stated otherwise. This model generated a random demand of between 1 and 10 channels between each pair of nodes independently. The demands were routed based on the weighted shortest path method described in Chapter 2.

For each graph, we applied our HP-tree design algorithm to construct the HP-tree in the network using the four suggested node placement methods. In design of the HP-tree and performance computation for each network we used the placement method that provided the best result for that specific network, unless comparative performance results of the node placement methods were discussed in which case the results were collected separately for each placement method in each network.

The confidence intervals for the mean results in each figure in this chapter were computed in the same way as Section 3.5.1. The 95% confidence interval for the results in this chapter was ±3% of the mean value at each point, unless stated otherwise.

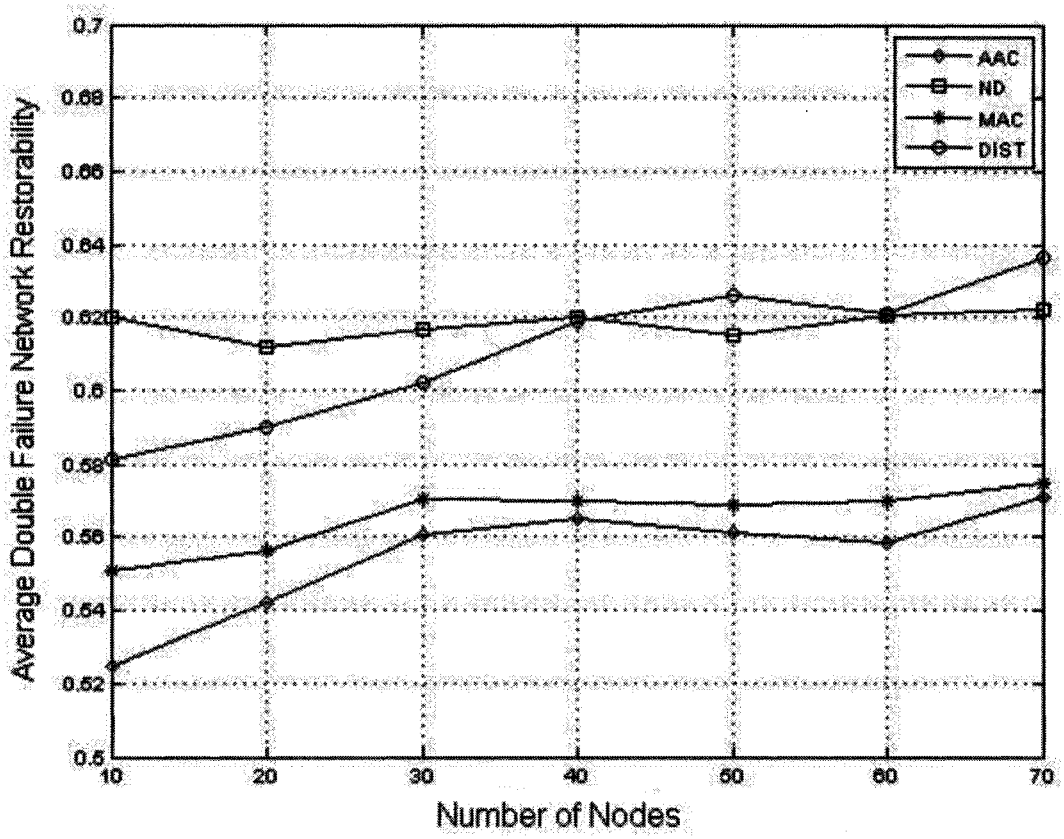


Figure 5-1: Average Double Failure working-only Restorability vs Number of Nodes

5.4.1. Double Failure Restorability Results

In this section we study the double failure restorability performance of HP-tree in networks that their allocated spare capacity is set to the minimum redundancy required to achieve full single failure restorability. For HP-trees, the spare capacity requirements

were calculated using the HP-tree design algorithm in Chapter 3. The HP-tree was reconfigured after the first failure based on the algorithms in Section 5.3.3.

Figure 5-1 shows the average double failure working-only restorability versus the number of nodes. In computing each point on the curves, the results for individual networks of the same order (but different nodal degrees and number of edges) were computed and averaged. The results showed that the topology-based ND node placement criteria provided the highest double failure restorability. The likely reason is that by assigning more protection responsibilities to links whose adjacent nodes have larger nodal degrees, more network traffic could be re-routed upon subsequent failures. This result is also related to the way we defined the double failure restorability in Section 5.3.1. For the ND method, the restorability was about 62% on average and fairly constant with the number of nodes. Small variations in the average values were the result of the impact of difference in nodal degrees of the graphs in addition to number of nodes. For the other three node placement schemes, the double failure restorability increased slightly with the number of nodes in the network.

In general, there are several factors related to network size that affect the restorability of the HP-tree in different ways. Our results in Chapter 3 showed that the single-failure restorability slightly decreased with network size if the average nodal degree was low. The likely reason was that as network grew, there was a higher possibility of not being able to find a backup parent for a node while constructing or repairing the tree post failure. As confirmation, when the average nodal degree of the network increased (thus increasing the possibility that a non-tree link would be found in most circumstances), the curve of restorability versus network size became almost flat. On the other hand when networks grow in size, there is a better chance that the impact of consecutive failures would be independent, thus increasing the double failure restorability. The overall result would depend on the size of the impact of each individual factor for a specific network.

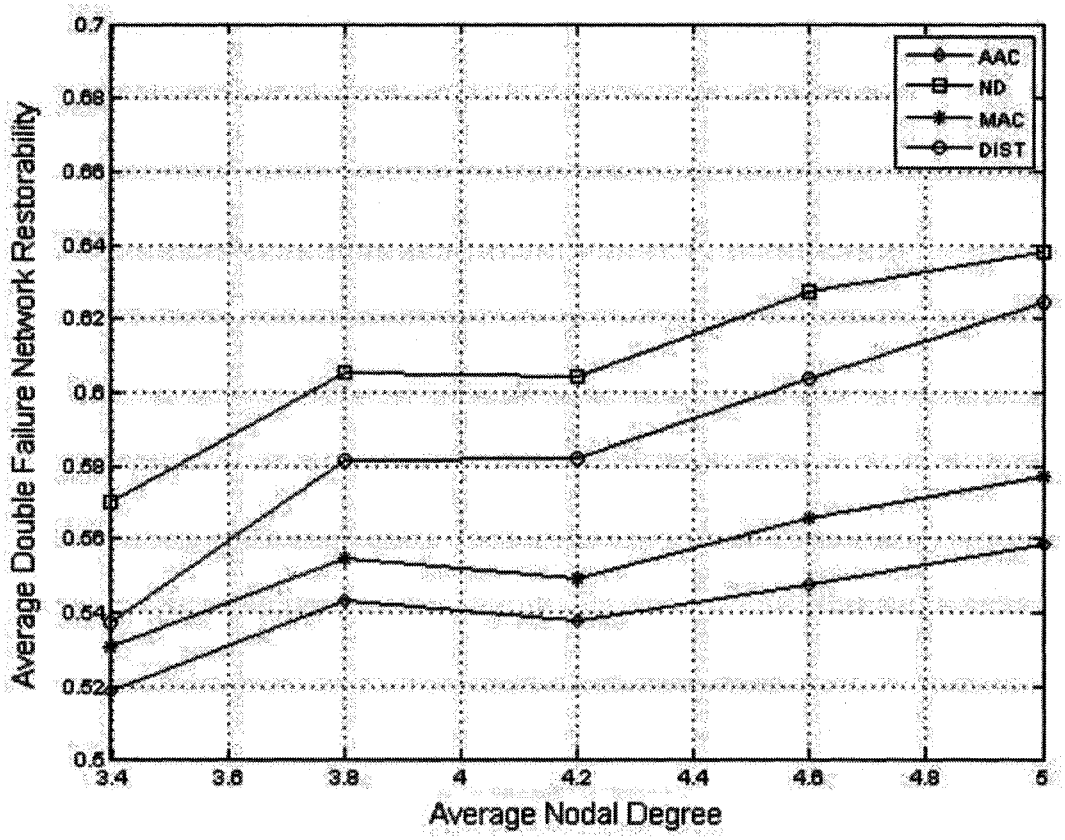


Figure 5-2: Average Double Failure working-only Restorability vs Average Nodal Degree

Figure 5-2 shows the results for the four node placement criteria versus the average nodal degree of the network. In computing each point on the curves, the results for individual networks of the same average nodal degrees (but different network sizes) were computed and averaged. The points on each curve were connected with straight lines. The double failure restorability increased with average nodal degree, as more non-tree links were available for redirection of network traffic upon the second failure. Small variations in the slope of the curve could be attributed to the fact that networks of different sizes were being used for computing the average at each point. So those statistical variations should diminish if a larger set of networks is used. Overall, the ND method achieves the best result with double failure restorability increasing from an average value of about 57% for average nodal degree of 3.4, to about 64% for average nodal degree of 5.0. The DIST, MAC and AAC methods followed respectively.

We also studied the average results versus the number of edges in the network, as well as topological characteristics of the graph such as network radius and average eccentricity of network nodes. Our results did not indicate any significant change in double failure restorability with respect to the number of edges in the network, nor the average network eccentricity. As our results in Chapter 3 indicated, the network size did not have a significant impact on restorability in networks with average nodal degrees higher than a threshold. In our computations here all networks were three-edge connected, which required a minimum nodal degree of three or higher. As a result, the graphs in the subset used in this section had an average single failure restorability of 0.999. Thus the impact of network size and eccentricity on single failure restorability had already been diminished, and as our computations showed, it had a similar impact on double failure restorability as well.

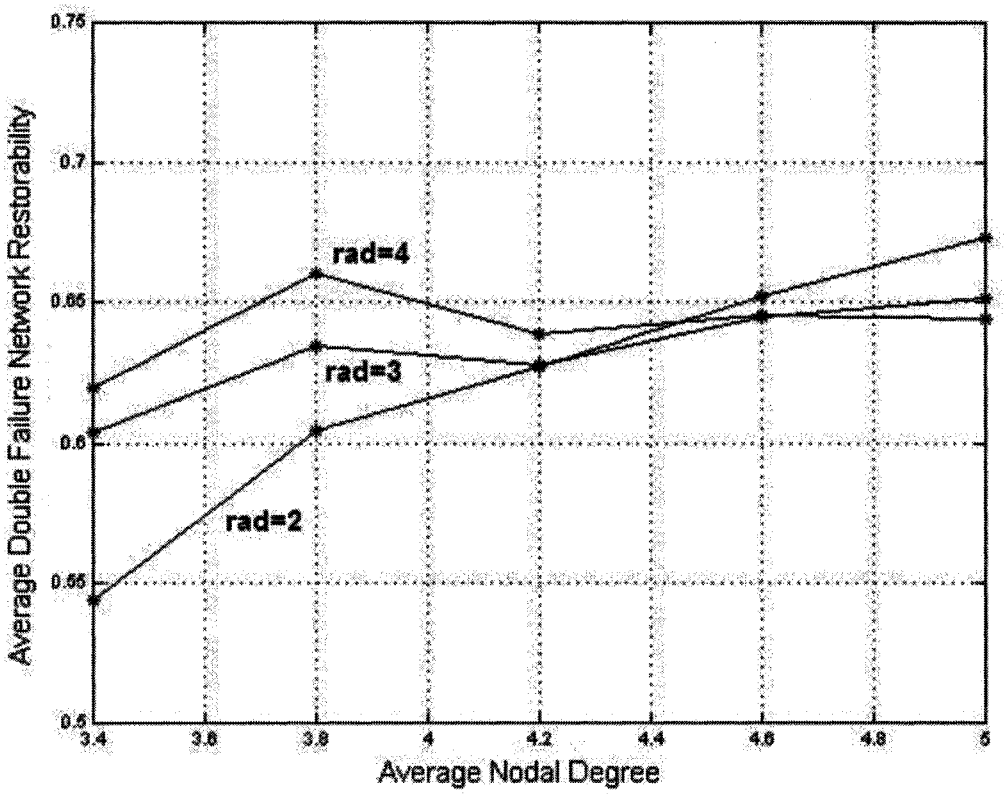


Figure 5-3: Average double failure working-only restorability for network radius (hops)

Figure 5-3 shows the double failure restorability results versus average nodal degree, with the curves separated based on the network radius (indicated as *rad*= number of hops). Each point on a curve is the average HP-tree double failure restorability value for all those graphs with the same network radius and average nodal degree (but different network sizes). In order to be able to study the impact of network radius and average nodal degree independently from the impact of 3-node chains, only graphs with 100% single failure HP-tree restorability were included in this graph. The points on each curve were connected with straight lines.

The results are interesting in the fact that it indicates a difference in the trend of restorability versus network radius. While at high nodal degrees, the double failure restorability decreases with network radius from about 0.68 for *rad*=2 to 0.64 for *rad*=4, as the average nodal degree decreases the impact of network radius on double failure restorability is reversed, this time from 0.54 for *rad*=2 to about 0.62 for *rad*=4. The curve for *rad*=4 also indicated the possibility that at higher network radius, there might be an optimal value for average nodal degree at which the double failure restorability of HP-tree is maximized, although more simulation results with a larger number of graphs are required to rule out statistical inaccuracies.

Table 5-1: Average network size for respective graphs in Figure 5-3

Network Radius	Average Number of Nodes
2 hops	12
3 hops	28
4 hops	47

However the impact of network size might have also played a role here too. Table 5-1 shows the average number of nodes for each curve in Figure 5-3. We already know from Figure 5-1 that double failure restorability of HP-tree increased with number of nodes if the average nodal degree was low, which is also the impact we observe here for average nodal degree under 4: the double failure restorability increased with network radius (and thus, number of nodes in the network). The justification here is the same: the isolation of the impact of consecutive failures in larger networks. In particular when the average nodal degrees are low, there is a higher chance that consecutive failures might jeopardize

the ability of the HP-tree to repair itself, as fewer paths would be available to each node. Therefore, increasing the average nodal degree would reduce the impact of the network size by providing more options for backup parents at each node.

5.4.2. Restorability Efficiency Results for Full Double Failure Restorability

After a failure if the self-repairing mechanism of the HP-tree were successful (being able to choose another backup parent node), then the HP-tree would be able to fully recover from subsequent failures if enough restoration capacity has been allocated in the network beforehand. The amount of spare capacity requirement increases substantially with number of consecutive failures that the network should recover from. It is also much higher for all-capacity restoration compared to working-only restoration. The results for double-failure working-only scenario are presented here. All-capacity and Triple failure results will be presented in next sections.

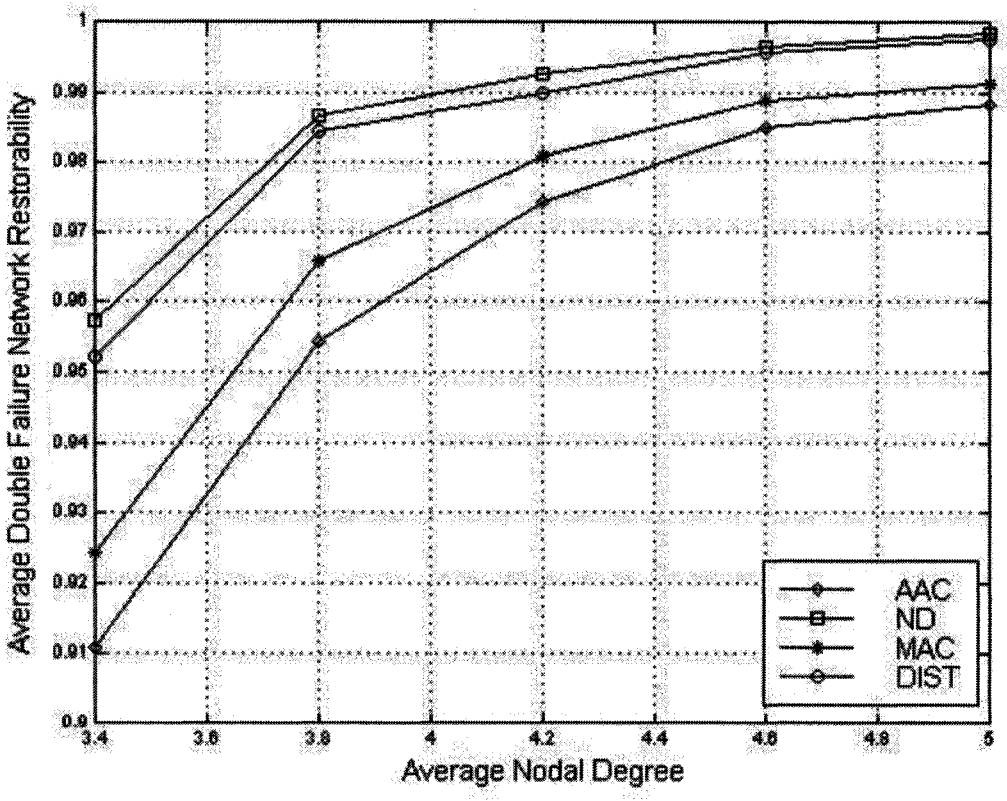


Figure 5-4: Average double failure restorability limitation of the HP-tree

As mentioned in Chapter 2, comparison of redundancy requirements is valid only when the same restorability levels are considered. When using the HP-tree scheme in networks where single failures can be fully restored, the double failure restorability may be less than 100% if the first link failure creates a topological constraint for the HP-tree in restoring the subsequent failures. Figure 5-4 shows the average value of the maximum achievable double failure restorability when spare capacity for maximum double failure restorability was assigned beforehand. All networks in this sample can restore single failures 100% under each node placement criteria. So the figure provides a fair comparison of achievable double failure restorability with each method. Each point on a curve was the average value of restorability for all networks in our sample set with the same average nodal degree. The points on the curves were connected with straight lines. The ND and DIST method achieved the highest double failure restorability between about 0.96 for d=3.6 to 0.998 for d=5. The AAC method had the lowest restorability of 0.91 at d=3.4 and 0.988 at d=5, with the MAC method performing slightly better.

As a result, the maximum achievable restorability must be taken into account in comparing the redundancy requirements in different scenario. In order to be able to make a fair and objective comparison, we use *Network Restorability Efficiency* as the measure of the redundancy in the network to indicate the amount of protected working capacity for each unit spare capacity in the network. Admittedly, the usefulness of this definition depends on the objectives of the restoration; i.e. if only full restorability is desired, the restorability efficiency parameter would not provide a good performance measure. Based on Equations 2.4 and 2.6, we define the double failure network restorability efficiency E_N as:

$$E_N = \frac{R_2}{N_r} = R_2 \times CSF_2 \tag{5-5}$$

in which R_2 is the maximum double failure network restorability and CSF_2 denotes the double failure capacity sharing factor for R_2.

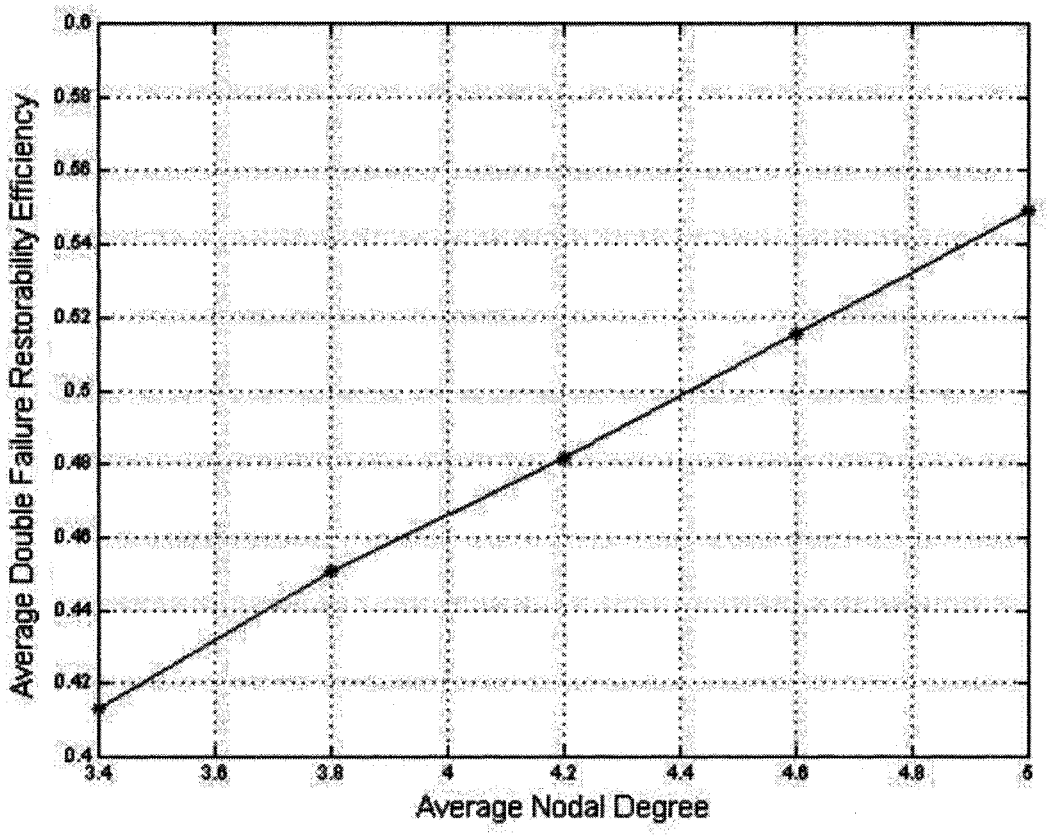


Figure 5-5: Average Double Failure Restorability Efficiency En vs. Nodal Degree

Figure 5-5 shows the average double failure restorability Efficiency E_N versus average nodal degree for the ND method. All networks in this sample set could achieve full single failure restorability with all four node placement criteria, and while the ND method performed slightly better, the E_N results were very close for all four methods (within a ±2% range). Each point on the curve represents the average of the results for all networks with the same average nodal degree. The double failure redundancy efficiency increased from about 0.413 for d=3.4 to 0.55 for d=5.

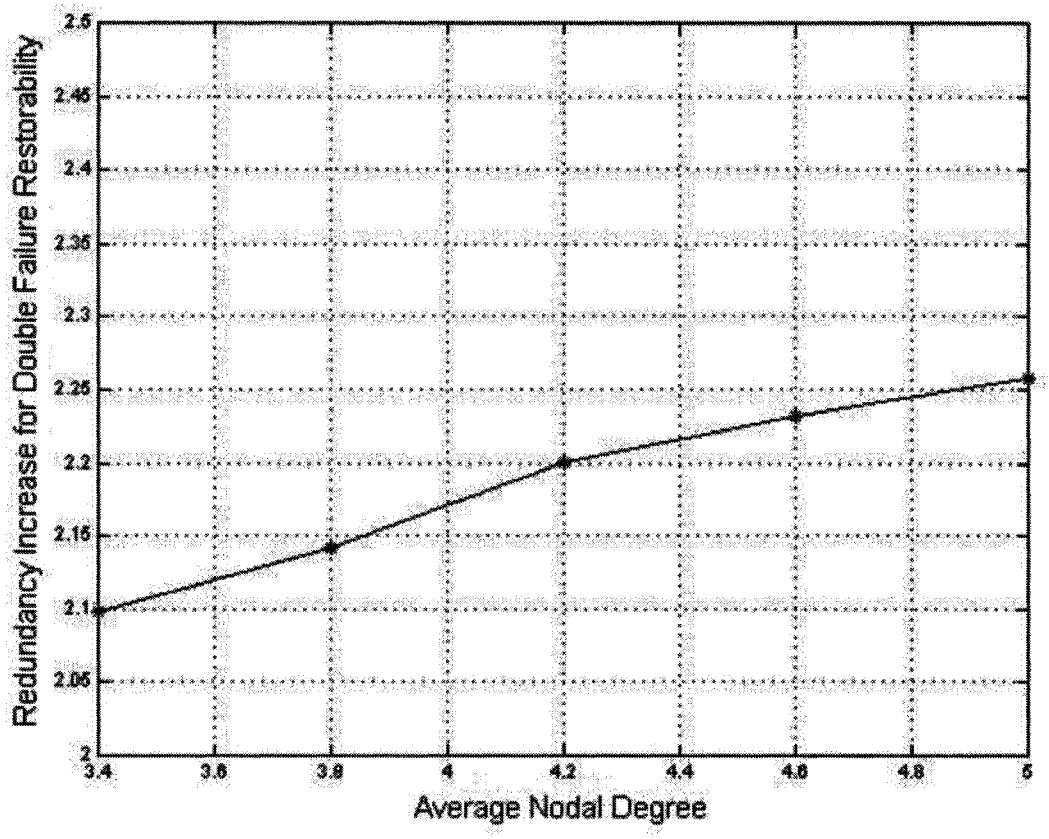


Figure 5-6: Normalized redundancy for Maximum double failure restorability

In order to measure the redundancy requirement for recovering the second failure, we use *S*, the ratio of the required redundancy for maximum double failure restorability to the required redundancy for maximum single failure restorability. Figure 5-6 shows this ratio as a function of the average nodal degree. Each point on the curve is the statistical average of results for all networks that had the same average nodal degree but different sizes. The points on each curve were connected with straight lines. The results are shown for the ND node placement criteria; however, there was no noticeable difference in the computed redundancy results for other node placement methods. Note that the redundancy requirement increased with average nodal degrees here because the achieved double failure restorability, as shown in Figure 5-4, was lower for smaller nodal degrees. As the curve shows, depending on the average nodal degree of the network, additional spare capacities between 109% and 126% of the amount of original spare capacities (for

single failure case) should be added to the network to achieve maximum double failure restorability with HP-tree.

5.4.3. All-Capacity Double Failure Restorability and Redundancy Results

The presented results so far were for the working-only restoration scenario. We expected that if all traffic were to be restored based on the all-capacity restoration model described in Section 5.3.4 in a network that has been designed for single failure restorability, the double failure restorability would be much lower because of insufficient capacity; i.e. the blocked path failure model, described in Section 5.1. Similarly, the redundancy requirement for achieving maximum double failure restorability would be higher too because the amount of working capacity that must be restored is larger.

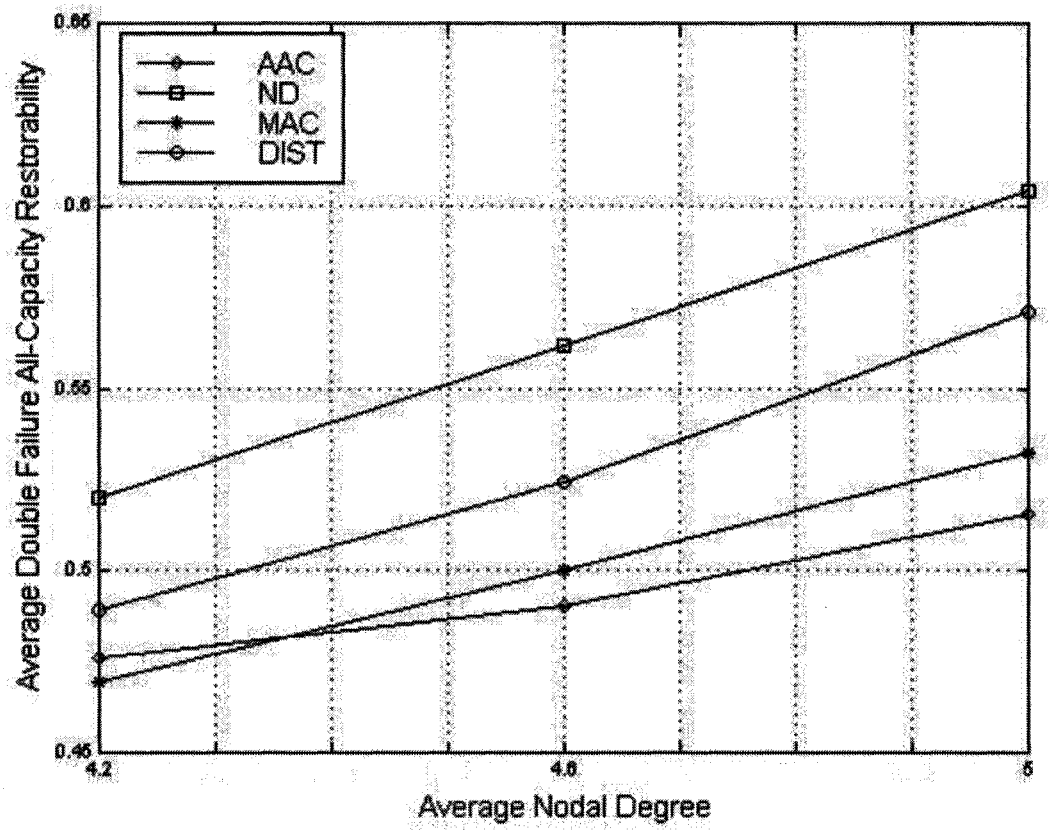


Figure 5-7:Average Double Failure All-Capacity Restorability versus Average Nodal Degree

Figure 5-7 shows the results for different average nodal degrees. Each point on a curve is the statistical average of the results for all networks that had the same average nodal

degree but different sizes. The points on each curve were connected with straight lines. As the working-only restoration scenario (Figure 5-2), the ND and DIST node placement criteria achieve the highest double failure restorabilities. Average restorability for the all-capacity scenarios using the ND method is about 0.52 at d=4.2, 0.56 at d=4.6 and 0.605 at d=5.0, comparing to the respective restorability values of 0.60, 0.62 and 0.64 for working-only scenario.

Table 5-2: Redundancy increase for all-capacity double failure restorability

	AAC	ND	MAC	DIST
Average Redundancy increase from Single failure case	199.16%	196.66%	198.88%	199.52%

Table 5-2 indicates the average redundancy increase for the all-capacity cases. Comparing to the working-only double failure restoration where on average the spare capacity would have been scaled up by 120%, here the spare capacity must be almost tripled (200% increase) to achieve all-capacity double failure restoration. The 95% confidence intervals for the mean results in Table 5-2 were within ±1% of the average values.

5.4.4. Partial Double Failure Restorability Results

We also studied the relationship between the amount of additional redundancy in the network and the level of double failure restorability that can be achieved, based on the model presented in Section 5.3.4. About 450 networks were randomly selected from our random mesh network database, and for each network, the amount of redundancy was increased step by step from the minimum redundancy required for single failure restorability. The double failure working-only restorability was computed at each step.

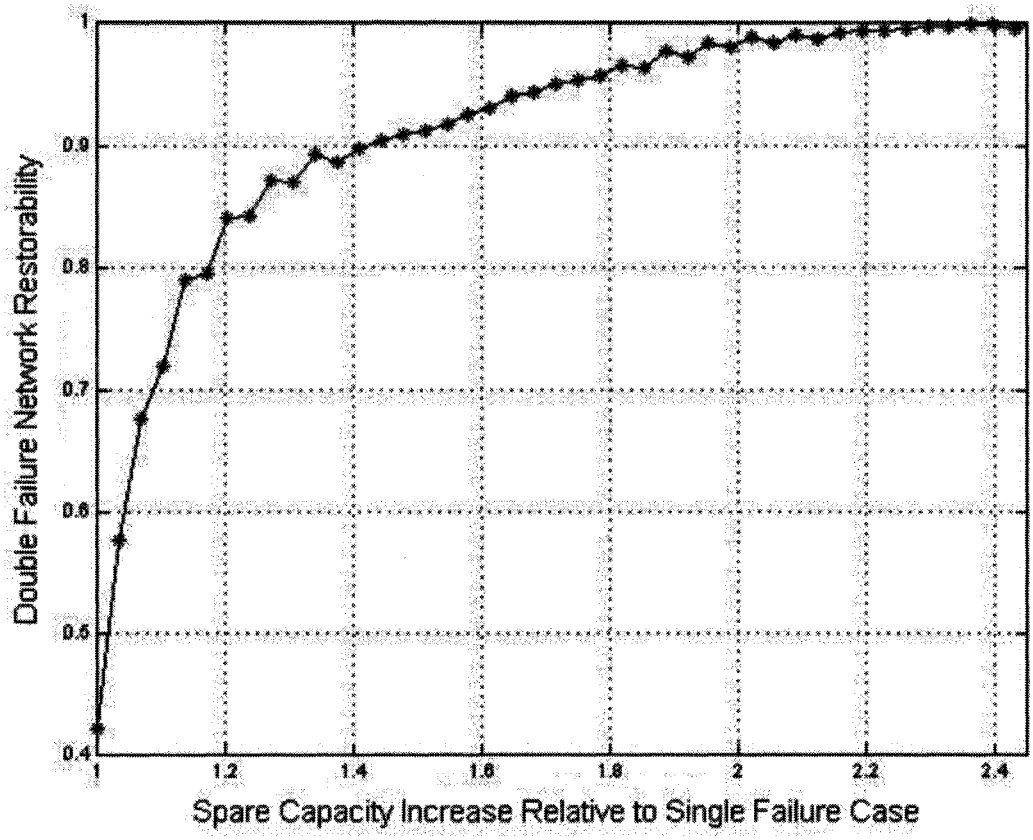


Figure 5-8: Partial Double Restorability Increase vs. Redundancy Increase

Figure 5-8 shows the double failure restorability as a function of the ratio S between the allocated spare capacity in the double failure case and that required for single failure restorability. Each point on the curve represents the average value of double failure restorability among all networks in the subset for the given spare capacity increase in the network. The double-failure restorability curve is an increasing function of S as expected. The curve levels off around S=2.2 when it is very close to 100% restorability. Although the redundancy requirement here for 100% double failure restorability is very high (~220%), most of the increase can be achieved with some modest increase in the beginning. For example, an increase of 20% (from S=1 to 1.2) can sharply improve double failure restorability of the HP-tree from 40-50% (for S=1) to 80-85% range (for S=1.2) Further note that in the model here a very simple and scalable mechanism was used for redundancy increment steps, where spare capacity was increased based on the sharability of each link using the RP matrix information that could be made easily

available to network nodes. If we were to use centralized optimization methods to assign spare capacity increments in optimal locations, we would expect to achieve even higher double failure restorabilities at the cost of more complex computations and deployment difficulties in a distributed environment.

5.4.5. Triple Failure Restorability and Redundancy Results

The algorithm in Section 5.3 allows us to extend computation of k-failure scenarios from double to triple failure restorability easily. As mentioned before, the complexity of the computation increases by a factor of network size M from k-failure to $(k+1)$-failure analysis. Furthermore, for study of triple failure scenarios the networks must be four-edge connected. Testing the network for four-edge connectivity would also be M times more complex than testing it for three-edge connectivity. For this reason the amount of time and processing power could become prohibitive for triple, quadruple and higher failure scenarios. In our simulations we selected a subset of just over 700 four-edge connected networks from our database of random graphs. The networks ranged in network size from 21 to 125 edges, with total number of nodes from 10 to 50 nodes and average nodal degrees of 4.2, 4.6 and 5. Note again that in a four-edge connected network, no node can have a nodal degree less than four.

In studying the HP-tree performance under triple failure scenarios, our intention was to discover trends in restorability and redundancy requirement for networks as we move from single-failure scenarios to double-failures and further to triple-failure cases; thus hoping that further extrapolations can be made for larger failure scenarios based on the trends discovered here. We studied the restorability of the HP-tree in all-capacity and working-only restoration scenarios, and computed the redundancy requirements for the maximum restorability achieved by the HP-tree in each case.

Furthermore, in analyzing triple-failure restorability cases, we assumed that the network had already been capacitated to achieve maximum double failure restorability. The purpose was to discover how much additional capacity was required in the network to move from k-failure restorability to $k+1$-failure restorability in an HP-tree protected network.

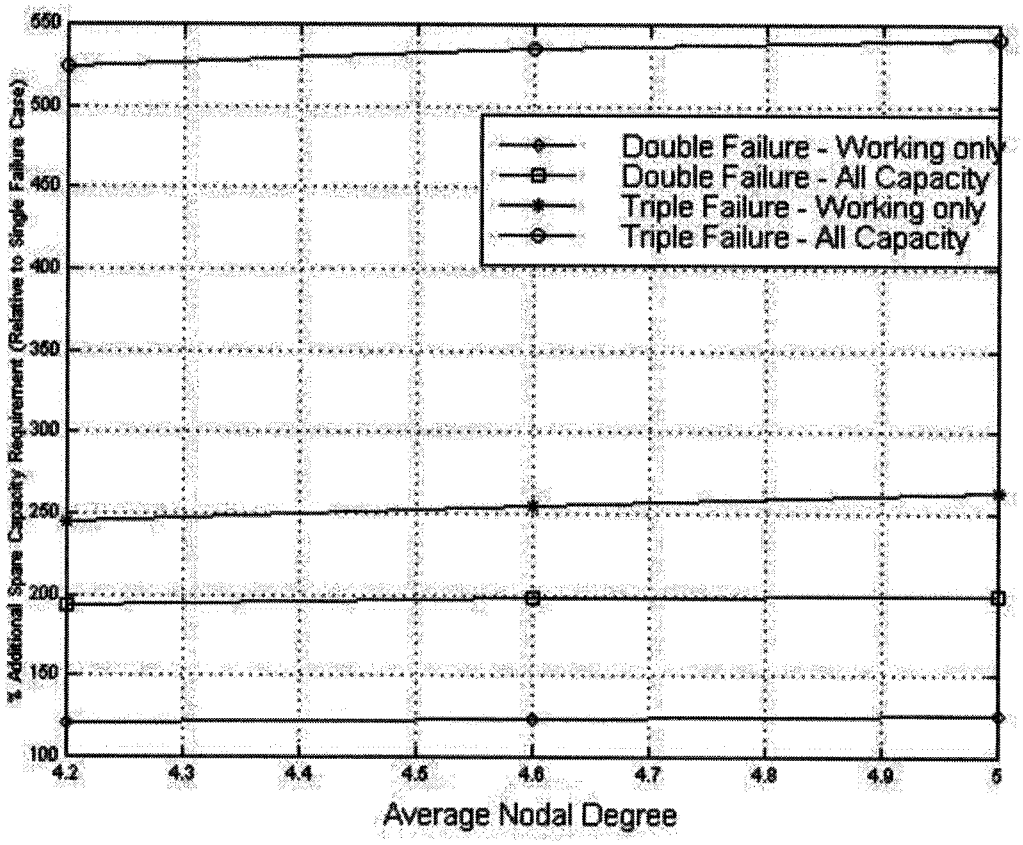


Figure 5-9: Additional Redundancy Requirements for Double and Triple Failure Scenarios

Figure 5-9 shows the additional spare capacity requirement in percentage for each scenario, relative to single failure redundancy requirements. Each point on a curve was the statistical average of the results using the ND method for all networks with the same average nodal degrees. Double failure results have been included for comparison. In the working-only restoration scenario the single failure spare capacity should be increased by 250% (more than tripled) to achieve triple failure restorability, while for all-capacity scenario this increase was more than six-fold (over 500%).

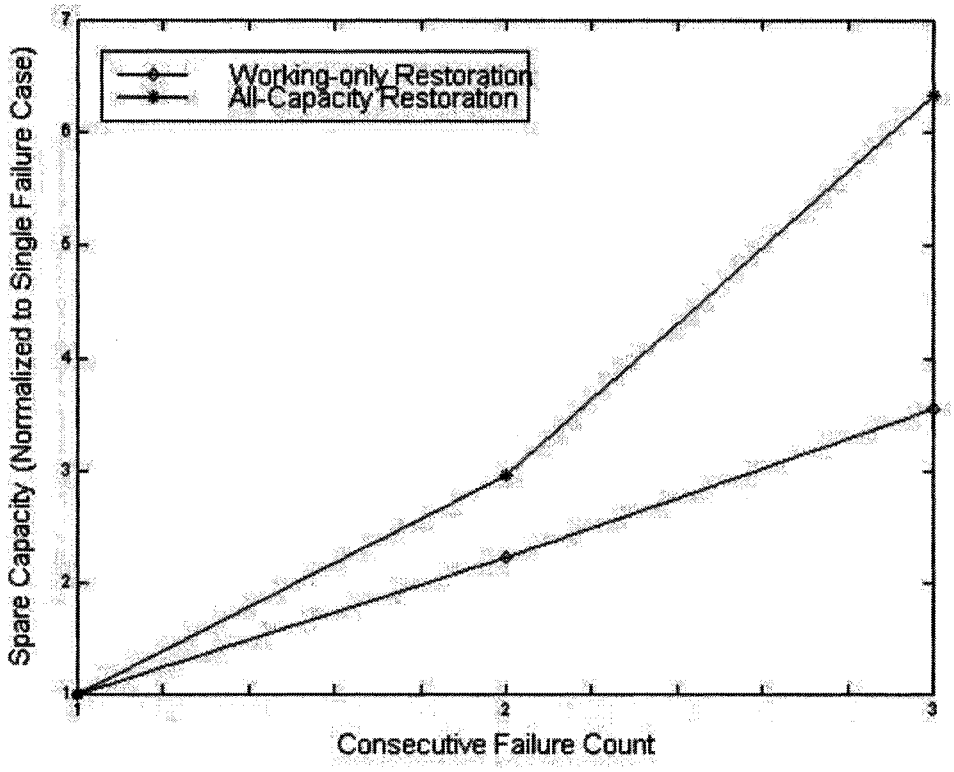


Figure 5-10: Redundancy increase for k-failure restoration

Figure 5-10 shows the trend in spare capacity requirement increase for k-failure restoration scenarios. The redundancy for working-only restoration scenario had a near-linear curve (increasing with $k^{1.15}$ as determined by curve fitting), while the redundancy for all-capacity restoration increased in a more exponential trend as the inclusion of protection capacity for k^{th}-failure in the working capacity of the $(k+1)^{th}$ failure would have an accumulating impact on the total redundancy requirements. This factor should be taken into account when deciding the cost of guaranteeing multiple failure restoration to both working and spare capacity of the network, or only the working capacity.

Table 5-3: Triple Failure Restorability for Double Failure Redundancy Levels

	AAC	ND	MAC	DIST
Working-only Restoration	0.8706	0.9093	0.8752	0.8934
All-Capacity Restoration	0.8828	0.9150	0.8822	0.9007

While the redundancy requirements for achieving full triple failure restorability are much higher than double failure restorability case, our HP-tree can still provide a fairly high triple failure restorability level in a network designed with capacity for double failure protection. Table 5-3 shows the results for each node placement methods. Note that in the working-only restoration scenarios the network spare capacity allocation was based on working-only restoration design, while in the all-capacity restoration scenario it was based on all-capacity restoration design (and thus had higher redundancy according to Figure 5-9). The results indicate that on average we were able to achieve up to 90% triple failure restorability in a network that was designed for full double failure restorability. Therefore from a network planning perspective, the cost of doubling the spare capacity of a network to increase the restorability from 90% to near 100% would be an important factor to take into account.

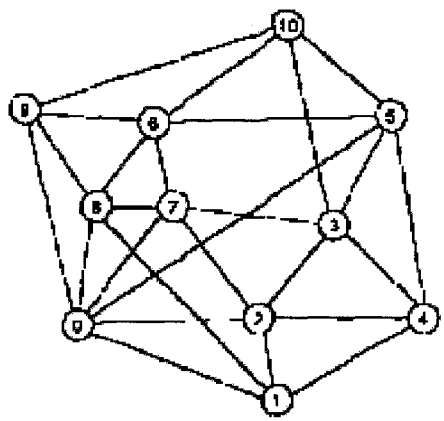


Figure 5-11: Sample Graph for Triple Failure Restorability Analysis [Schu03]

We also applied our multiple failure algorithm to compute double and triple-failure restorability of HP-trees for a sample graph from [Schu03], which is presented in Figure 5-11. We use the demand matrix provided in [Mauz02] for this network. The demand is routed using the weighted shortest path algorithm that increased the cost of each link based on the number of working capacities already routed on that link, thus achieving load balancing across the network nodes. Protection capacities were assigned on each link based on our HP-tree design algorithm from Chapter 3 in order to achieve full single failure restorability. Then we computed the double- and triple-failure restorability, and the amount of spare capacity for achieving full single, double and triple failure restorability in the working-only restoration scenario.

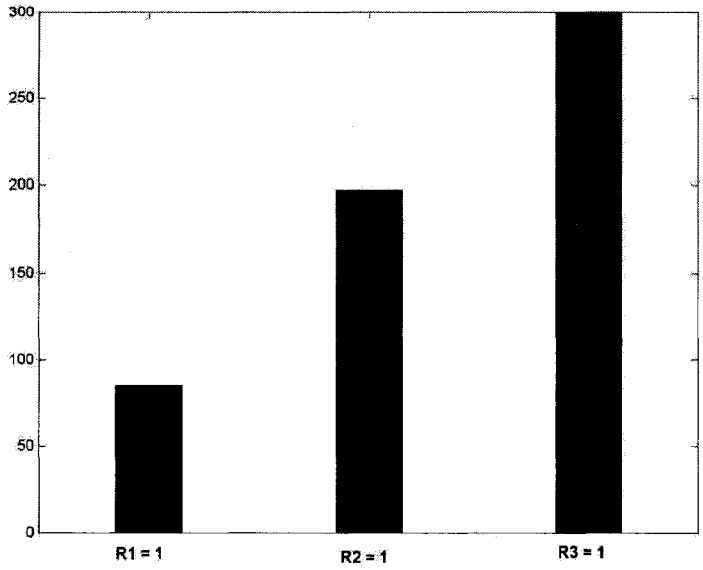


Figure 5-12: Redundancy requirements for full single, double and triple failure restorability

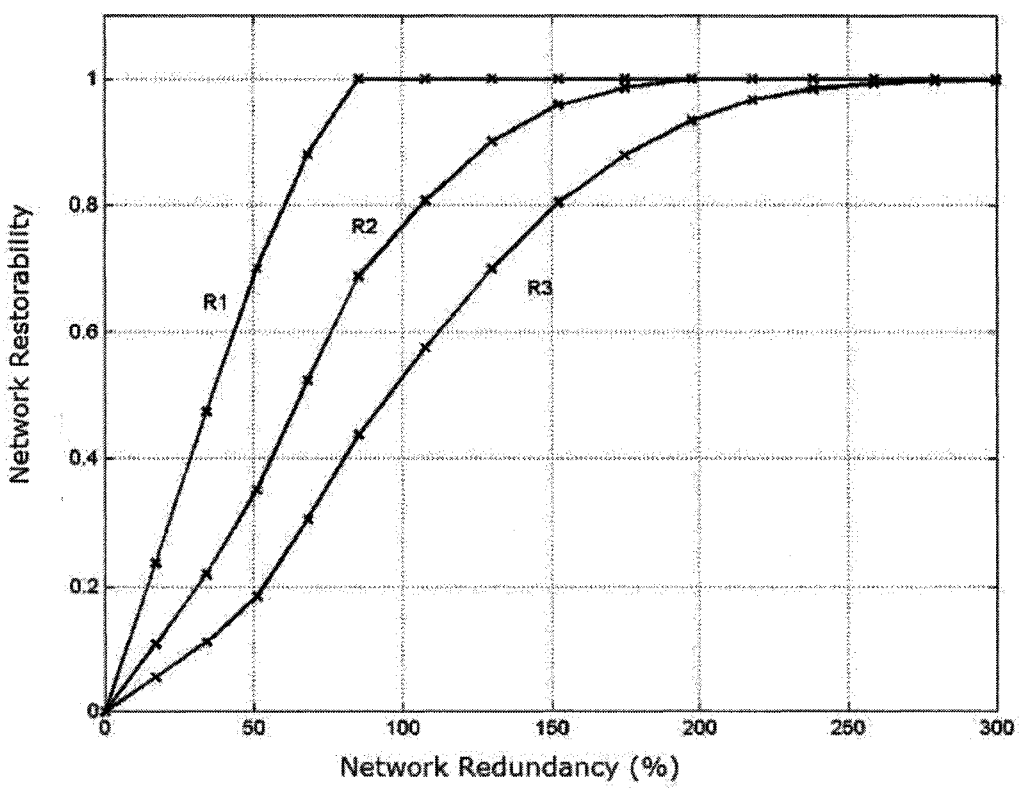


Figure 5-13: Multiple Failure Restorability versus Redundancy in the sample Graph

The amount of redundancy required to achieve full restorability in each case is presented in Figure 5-12. Restorability results are shown in Figure 5-13 for various amounts of redundancy. R1, R2 and R3 respectively represent single-, double- and triple-failure restorabilities. Full single failure restorability (R1=1) using the shared HP-tree is achieved at the network redundancy of about 85%. At this point, the double failure restorability stands at about 69%, while triple failure restorability is merely 44%. By increasing the redundancy to about 197%, full double failure restorability is also achieved, while the triple failure restorability is improved to about 93.3%. The redundancy could be even increased further to 299.5% in this case in order to achieve triple failure restorability as well.

5.5. Comparison with Other Results

In this section we examine results from some references on multiple failure performance of other schemes and compare them to our results. In [ClGr02] the double failure restorability of mesh restorable networks was studied. The spare capacity in the network had been designed for single failure case. The sample network sizes ranged from 6 to 22 nodes. The restorability results for mesh restorable scheme were in the range of 0.59 to 0.80 for fully static schemes, which would increase to 0.61 to 0.83 if the scheme were fully adaptive. The double failure restorability was defined according to Equation 5-3, which is different from what we used here. However our HP-tree results (~0.60-0.64) seems to be near the lower end of the results obtained in [ClGr02]. The authors further extended their study of mesh restoration in [ClGr05] where the redundancy requirements were examined. The redundancy results for the six networks studied in [ClGr02] showed the double failure redundancy requirement was between 265% to 340% of the single failure spare capacity requirement. Our HP-tree results in Figure 5-6 was about 220% for working-only and 295% for all-capacity restoration, though one should note that the optimal mesh restoration achieves better single failure redundancy than other schemes including HP-tree, and thus the absolute redundancy values remained lower for optimal mesh restoration. A partial restorability curve was also presented in [ClGr05] to indicate how the double failure restorability increased with redundancy. For most sample networks in that case, an increase of restorability from 60-75% to 90% required a redundancy increase to 150-200%. Our HP-tree curve in Figure 5-8 is much sharper,

where the average double failure restorability is increased from 40% to 90% only by a 37% increase in redundancy. However this effect could be related to the fact that we used network restorability in our analysis, and therefore higher priority was given to restoring high capacity links.

Double failure restorability and redundancy results were also presented in [DoGr02] where five test networks of between 16 to 46 nodes were examined. That paper used average link restorability (Equation 5-1) as performance measure. The networks all achieved relatively high double failure restorabilities between 84% and 95% when designed for single failure redundancy. However, increasing those values to full (100%) restorability required increase of network redundancy to 250% or 300% of the single failure cases. As mentioned before, the redundancy results are consistent with our results for HP-tree as well.

Schupke studied double failure scenarios for pre-connected protection cycles in [Schu03, ScGr04]. He computed the double failure restorability based on the length of the protection cycle and the susceptibility parameter, and was able to achieve double failure restorability of 90% with protection cycle by reducing susceptibility; though at an additional 90% spare capacity cost. In static protection cycles (and every other static protection scheme), 100% double failure restorability is infeasible. His results for reconfigurable cycles indicated that if global reconfiguration and capacity optimization of cycles are used, full double failure restorability could be achieved at about 160% spare capacity (relative to single failure redundancy level). If only additional cycles were added (and previously assigned cycles remained unchanged), spare capacity requirement was about 220% of single failure case for working-only scenario, and about 250% for all-capacity scenario. The relative increase in redundancy requirements with protection cycle is within the same range as our results for HP-tree, though the single failure spare capacity base would normally be lower for cycles as discussed in Chapters 1 and 3 of this thesis. When full reconfiguration of protection scheme was employed the redundancy requirements were decreased; at the cost of higher administrative and computational overhead for reconfiguring all backup paths.

5.6. Concluding Remarks

In this chapter we studied the multiple failure scenarios for the HP-tree. Our contributions included models for HP-tree multiple failure operation, computation methods for multiple failure restorability in static and dynamic scenarios, and extensive simulation results for double and triple failure restorability and redundancy requirements of the HP-tree in random mesh networks. The results indicated many important trends in how the restorability of the HP-tree would be affected by consecutive failures, and how much redundancy increase would be needed to protect the network against multiple failures. Our partial restorability results in Figure 5-8 suggests that by restricting the multiple failure protection service to a portion of the traffic (e.g. the high priority traffic), significant saving on redundancy requirements of the HP-tree for double failure and triple failure restoration could be made.

Chapter 6. Distributed Signalling for Construction of Protection Tree

In this chapter we present a distributed signalling method to illustrate how a hierarchical-protection tree can be constructed in a mesh network. Some basic mechanisms in our approach are similar to IEEE 802.1d Spanning Tree protocol (STP), which is used to create a reliable routing tree among network bridges. The idea of designated and alternate bridge ports in STP algorithm has similarities to primary and backup parent nodes in hierarchical-protection trees. We extend this algorithm to create the hierarchical tree based on the available protection capacity at each node. Our approach includes two main steps: Finding the root node of the tree, and constructing a spanning tree from the root node.

6.1. Problem Statement

Our objective in this section is to design a signalling mechanism, including message type, timing and flow diagrams, to allow constructing an HP-tree among network nodes in a distributed fashion. The relevant scenario for this objective is the pre-designed case in Chapter 4; i.e. designing of the tree in a network that is already operational and has pre-allocated amounts of working and spare capacities on each link. We would like our signalling scheme to have the following characteristics:

1. To construct the HP-tree in a distributed manner with minimal need for a central database or decision center, except for selection of the root node from which the signalling starts.
2. To handle the construction of the HP-tree through message exchanging between network nodes.
3. To require a feasible and scalable amount of network state information on each node, preferably only local information.
4. To re-use existing network protocols and algorithms (OSPF, STP, etc) as much as possible in order to facilitate deployment.
5. To be able to converge to a solution in finite time; i.e. no infinite loop.

6. To have mechanisms in place to repair or grow the HP-tree in face of dynamic changes in network topology; i.e. addition and removal of nodes and links, or link and node failures.

6.2. Choosing the root node of the HP-tree

The construction of the HP-tree starts from the root node. A comprehensive discussion of root node selection for HP-tree was presented in Chapters 3 and 4, where we presented the node placement criteria and presented results of their performance in different scenarios. Generally speaking, the construction of the HP-tree can be initiated in one of the two ways

1. Manual initiation: In this case the network operator (human) or a central network management system initiates the construction of the HP-tree by sending a start message to a pre-provisioned root node. The root node selection could be done in any of the following manners:
    a. Random selection
    b. Based on node placement criteria
    c. Based on other design or business factors, such as site reliability.
2. Distributed selection of the root node: In this case, the nodes choose the best root node among themselves based on a pre-agreed upon criteria. This process could be done in the same manner as the STP algorithm [IEEE04] where the nodes announce their node ID and the one with the lowest node ID becomes the root node. Instead of the node ID, node placement criteria such as AAC, MAC, ND or DIST could be used.

In a distributed selection scenario, a root selection period would be necessary during which each node announces its availability and the value of the root selection parameter (computed by the node placement algorithm). This root selection parameter could be the average adjacent protection capacity for the AAC method, or the maximum adjacent protection capacity vector for the MAC method, or the nodal degree for the ND method, or the average distance for the DIST method. This parameter could be computed locally at each node for the first three methods (AAC, MAC and ND). But for the DIST method each node has to create a shortest spanning tree to the rest of the network to compute its average distance, so the initial root selection would take longer for the DIST method.

In our discussion here we assume a manually-initiated process will be implemented, where the management system or operator will be given the responsibility to initiate the process from the root node of his choice based on his selection of node placement criteria or other factors as mentioned. This process will not only be much faster and much simpler, but also gives the network operator the power to take various global network parameters into account in making his selection. The network could still adjust the location of the root of the tree based on dynamic changes in the network; we will discuss procedures for that purpose in Section 6.6.

6.3. Constructing the HP-tree

Once the root node is selected, then a single shared hierarchical-protection tree must be constructed in the network to create protection paths for each link in the network. In order to take full advantage of the characteristics of network trees, we developed a distributed mechanism for construction of the hierarchical-protection tree in the network and for assigning primary and backup parent nodes for each network vertex. This mechanism is primarily based on the concepts and algorithms discussed in Chapter 4, with the difference that it is implemented in a distributed manner among the nodes. Without loss of generality, in the rest of this chapter we assume that the AAC node placement criterion is used based on the general discussions in Section 3.3. This choice could be replaced with any other node placement method of choice.

We use signalling among the network nodes so that each node would know its position on the tree and would be able to identify its primary and backup parent nodes. The nodes perform this task by distributing identification labels to their child nodes. Each label includes an identification address called the tree ID of the node, and the respective available protection capacity through the current parent node toward the root of the tree. We use a dotted decimal notation for tree IDs to indicate the current position of the node in the hierarchy. The root node will have a tree ID address of 1. The tree ID for its immediate child nodes are 1.1, 1.2, 1.3, etc. Hence a node with tree ID 1.2.1.4.3 is a child node of the node with tree ID 1.2.1.4. Each node will keep and update a table of network node tree IDs for every other network node.

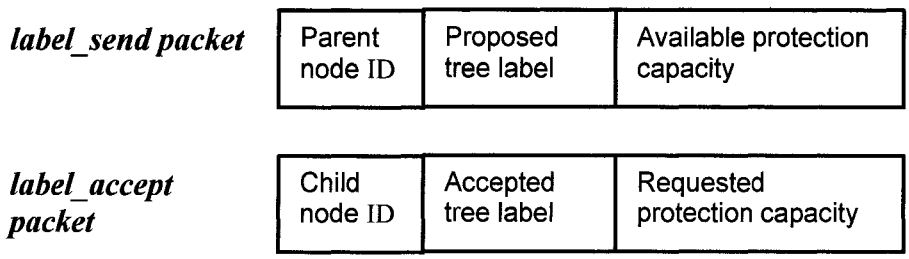


Figure 6-1: Pakcet formats for *label_send* and *label_accept*

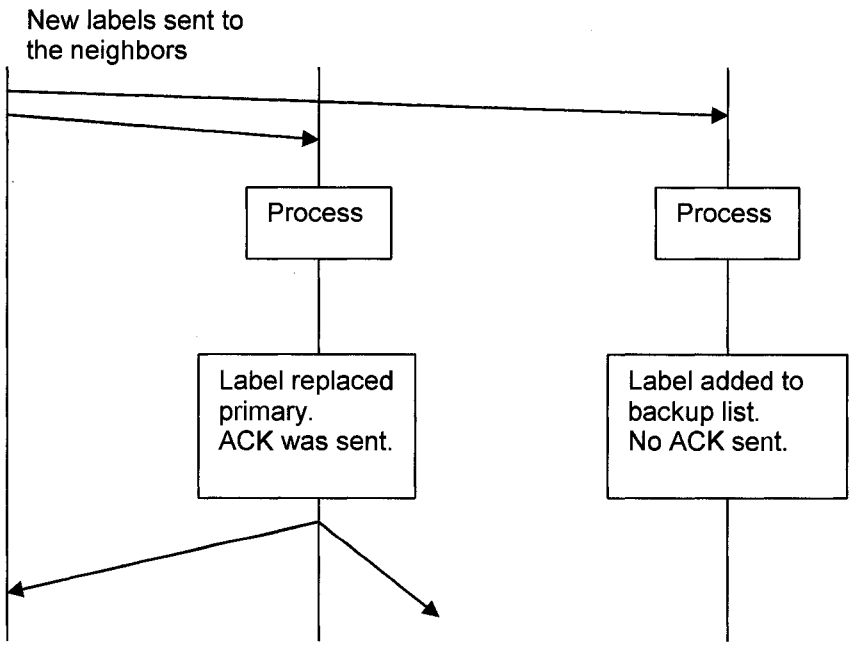


Figure 6-2: Signalling used in hierarchical-protection tree algorithm

Figure 6-1 illustrates proposed formats for signalling messages used in our scheme. Two message types are used in our distributed signaling algorithm: *label_send* and *label_accept*. A *label_send* message is sent by a node to its intended child-nodes. The *label_send* message is sent by a node to its target child node, and includes the assigned tree ID for the target node as described earlier, and the maximum capacity available through this parent node toward the root node. The *label_accept* message is an acknowledgment from child node that it has accepted this node as its parent, and confirms the same information, plus requested protection capacity from this node. If the

available capacity from the sender is less or equal to the available capacity from the current primary parent of this node, it just adds the label to its backup list and does not send any acknowledgement back.

The signalling message sequence in the process is demonstrated in Figure 6-2. Two cases are shown: when the target node does pick up the sender as its parent node, and when the target node saves it as a backup parent node. The algorithm starts with the root node sending *label_send* messages to its neighbors, using the *label_generating* algorithm below.

The *label_generating* algorithm

1. Define Vns = {Set of neighbor nodes}-{current primary parent node}
2. For each node in Vns:
    a. Copy the current label of this node into a new label.
    b. Add a new field (in dotted form) to the new label and increase the label field length by one.
    c. Insert the corresponding subindex in the new field of the new label. Sub-indexes should start from 1 for the first neighbor node and increment for next ones.
    d. Construct a *label_send* packet with the new label and send it to the corresponding neighbor node.
3. Repeat

With the exception of the root node, each node upon receiving a *label_send* message runs the *label_processing* algorithm that is described in the following.

The *label_processing* algorithm

1. Extract the parent node ID, proposed HP-tree label ID and the available protection capacity info from the packet.
2. Remove the sender from the receiver node's parent and child node lists.
3. Run the *label_loop_elimination* algorithm. If failed, discard the label.
4. Compare the available capacity of this new label with the current label of this node.
5. If new label has more available capacity:
    a. Select the sender as the new primary parent node.
    b. Move current primary parent node to sorted backup parent node list.
    c. Replace the current label of this node by the new label.

d. Set the available protection capacity to the lesser of the available protection capacity of the new label, and the protection capacity of the link that connects this node to its new primary parent.
e. Send *label_accept* acknowledgment back to the sender.
f. Update parent lists.
g. Call the *label_generating* algorithm.

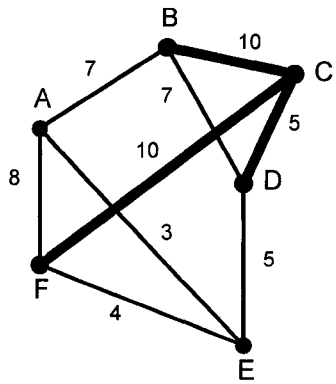
6. If the new label has less capacity than the current label:
   a. Add sender to the sorted backup parent node list(in proper place).
   b. Update parent lists.
   c. End

The *label_processing* algorithm uses the *label_loop_elimination* sub-procedure to determine if the newly received label was actually sent by a member of the child sub-tree of this node. If it is so, the label must be discarded to avoid loops.

The *label_loop_elimination* algorithm

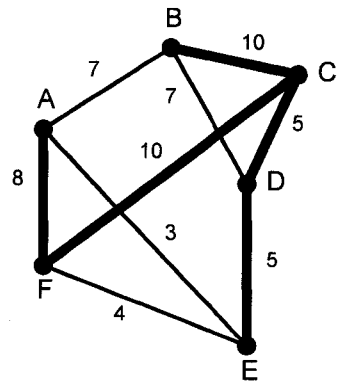
1. compare the new label with the current label of the receiver node.
2. if the new label could be generated from the receiver node's current label, return FAILED. Otherwise return PASSED.

The *label_send* message includes the protection capacity available from this node through the hierarchical-protection tree toward the root node. Note that if the path to the root node consists of multiple hops, the minimum of the capacities of the links on the path is reported. It is simple to verify that the available protection capacity of a node is equal to the minimum of the protection capacity available from its parent and the protection capacity available on the link that connects it to its parent node. For example if a node itself has a path with protection capacity of x units of bandwidth to the root node, when it sends a label to one of its child nodes which is connected to it by a link capacity of $y > x$, the label will include the designated capacity of x, not y.



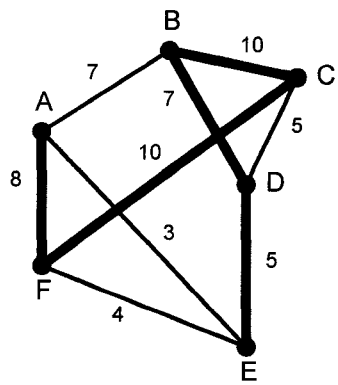
a) Step 1 Labels

C: 1
B: 1.1
F: 1.2
D: 1.3



a) Step 2 Labels

C: 1
B: 1.1
F: 1.2
D: 1.3
A: 1.2.1
E: 1.3.1



a) Step 3 Labels

C: 1
B: 1.1
F: 1.2
D: 1.1.1
A: 1.2.1
E: 1.1.1.1

Figure 6-3: Distributed algorithm for construction of an HP-tree

It must be noted that the algorithm always converges to a point where every node has selected a parent node from its neighbors that provides maximum protection capacity and no new labels will be generated after that. The process at each node will remain in a waiting or stand-by mode after this point until a topology change happens and new labels are generated. Each node would only need to maintain the tree label ID of its neighbouring nodes to determine its parent and child nodes.

6.4. Illustrative example of the HP-tree construction algorithm

Let us now examine the algorithm with an example. Figure 6-3 shows a network with six nodes, named from *A* to *F*. The numbers beside the links show the protection capacities of each link. We use the average adjacent protection capacity for determining the root node, and we include links with largest protection capacity in the tree, based on the concepts discussed in Chapter 4. Node *C* has the maximum average adjacent protection capacity and therefore is selected as the root node. It sends *label_send* messages to neighboring nodes *B*, *F* and *D*. HP-tree labels for these nodes are numbered from 1.1 to 1.3. In the second step, each of the child nodes *B*, *F* and *D* also send labels to their neighbors except their parent node. Node *A* receives two labels, a 1.2.1 ID from node *F* and a 1.1.1 ID from node *B*. Each node includes with the label the available protection capacity on the path from this node to the root node. Therefore node *B* can provide seven units of capacity for node *A* while node *F* provides eight units. Node *A* processes the incoming labels and picks label 1.2.1 from node *F*. It does not matter which label arrives first. If label from node *F* arrives first, node *A* picks it and rejects the next incoming label from node B. If label from node *B* arrives first, node *A* picks it and then changes it when the higher capacity *label_send* message from node F arrives.

Let us elaborate on how a node and its children change their labels dynamically. Originally, node *D* chooses root node *C* as its primary parent and assumes HP-tree label 1.3. Accordingly, it adds node *E* to the HP-tree with label 1.3.1. But next node *D* receives a label from node *B* with larger capacity, and switches to B as its primary parent node. Node *D* changes its own label to 1.1.1, and sends new *label_send* messages to its neighbors. Its child node *E* therefore changes its label to 1.1.1.1.

6.5. Recovery operation

Suppose that a hierarchical-protection tree has been constructed, and each node now knows its own tree ID (which indicates its position on the HP-tree) as well as that of its neighbors. If a network link fails, the downstream node can re-route the connections on the affected link toward its primary parent node. If the failed link was the tree link, the backup parent node will be used.

The routing of the traffic could be done locally at each node based on the tree ID label of the destination node. In such case, the amount of information stored at each node is on average *O(D)*. Alternatively, a source routing mechanism could be put in place where the downstream node of the affected link could specify the complete backup route to the upstream node. For the source routing case, each node of the network must have the tree ID label of other network nodes, which would require a database of *O(N)* complexity if source routing is used. Basically, such database would be similar to maintaining ARP tables on PCs on a LAN: one entry per node. It would not have to be updated often, as tree node IDs do not change with addition or removal of connections. Only topology changes (failures or add/removal of a node/link) would trigger changing of the tree node IDs. If source routing is not used, a node only needs to know the tree ID labels of its neighbors to decide where to switch the affected connections. The speed of switching would be fast, as transferring connections in bulk from one switch port to another (as done here) would take less time than transferring the connections individually.

For an example of recovery routing, suppose the link between a node with tree ID 1.3.2.1.1 and another node with tree ID 1.3.1.2 fails. Then the restoration path will be computed as follow:

(1.3.2.1.1) -> (1.3.2.1) -> (1.3.2) -> (1.3) -> (1.3.1) -> (1.3.1.2)

The following algorithm, called the *Recovery* algorithm, can be used to determine the protection path for a network link that connects node x to a node y. Note that there are two different cases: when the x-y link is a non-tree link, it is protected through the HP-tree. If it is a tree link, then the downlink node must first switch to its backup parent and then the path through the HP-tree is discovered. This algorithm could be used for source routing if required.

```
The Recovery algorithm
Suppose link x-y has failed.
1. If link x-y is a non-tree link, use p_tree_path(x,y).
2. If link x-y is a tree-link,
      a. If x is primary parent of y, use p_tree_path(x, backup_parent
         of y).
      b. If y is primary parent of x, use p_tree_path(y, backup_parent
         of x).
3. End
```

The *Recovery* algorithm uses the sub-procedure *p_tree_path*, which determines the path between two nodes of the network through the hierarchical-protection tree. The procedure returns a vector of edges that identify the tree path between the two nodes.

```
The p_tree_path(x,y) sub algorithm
1. Start from x,
2. Loop1:
      a. add Edge(x, primary_parent of x) to p_tree_path
      b. If y is not in the child tree of primary_parent of x, replace
         x with primary_parent of x
3. End of Loop1
4. Replace x with primary_parent of x;
5. Loop2:
      a. add Edge(y, primary_parent of y) to p_tree_path
      b. If x ≠ primary parent of y, replace y with primary_parent of y
6. End of Loop2
```

The path computation does not have to be performed after the occurrence of the failure. In fact, because our scheme is essentially a pre-determined shared link protection mechanism, it is possible for each node to calculate protection paths to each of its neighbors before the failure and to store them in a lookup table. Then when the failure happens, all the node needs to do is to retrieve the path information and re-route the connections accordingly. Therefore the restoration speed would be much faster than an ordinary path restoration scheme.

6.6. Addition or removal of network nodes and links

Our distributed algorithm can also adjust to dynamic changes in network topology that are not related to failure, such as addition of a link or node. Basically, the new node should introduce itself to the network and to request information on the protection scheme. We introduce a new message, *label_request*, for this purpose. Upon receiving a

label_request message, each neighboring node will send a label to the newly added node. Then the new node can choose the best parent node and respond with the corresponding *label_accept* acknowledgment.

It is important to note that the addition of a node may change the hierarchy in adjacent branches as well. Some of the neighbouring nodes may now find more protection capacity through this newly added node and hence switch to it as their primary parent node. Tree ID labels will also change as a result of this processing. Our tree construction algorithm will handle this procedure in a distributed manner.

If a new link has been added, the process is simpler. In this case, nodes on each side of the link must determine if the new link will provide them with more protection capacity. In order to do that, the nodes will send *label_send* messages to each other. The one node with less protection capacity will choose the other node as its primary parent if it provides more protection capacity than its current primary parent. If the protection capacities are less than current capacities or the same, no change happens.

Removal of a node or link can be treated in the same way as failure recovery. In such a case, the affected nodes switch to their primary or backup parent nodes (depending on whether the removed link was a tree link or non-tree link). They will then broadcast their new label information as well as *label_send* messages to their neighbours.

It is important to note that if nodes and links are deleted and added to the network frequently, there must also be a mechanism in place for recycling tree ID labels that are no longer in use. This can be done at each node by periodic monitoring of the current tree ID labels assigned to the child nodes. A list of unused IDs could be stored at each node and used for generating a new Tree ID label.

As mentioned in Section 2.6.2, the HP-tree cannot repair itself from the failure of its root node. The recovery mechanism in this case will depend on the root selection method for the HP-tree as discussed in Section 6.2. For networks in which the root node is selected centrally, it would be the role of the central Network Management System to detect the failure of the root node and select an alternative root node. The detection of root node failure could be done by periodical heartbeat signals from the root node to the NMS, though in such case there would be an additional detection delay equal to the period of the signal.

If root node selection is done in a distributed manner by the network nodes, then upon the failure of the root node, its immediate child nodes could trigger the root selection process again. As described in Section 6.2, a root selection period would be needed in this case to stabilize the decision. Either way, once the new root node is selected, it can re-initiate the HP-tree formation process.

As such, the failure of root node would have the biggest service impact on the HP-tree. Therefore, it would be wise to select a node with high site reliability and option of hot swap backup switches on site.

6.7. Performance Evaluation

The important performance parameters for our distributed signaling protocol include the complexity and restoration time. The restoration time for HP-trees was analyzed in Section 2.6, and the same analysis applies here as well. Whether the HP-tree is constructed using the centralized algorithms or distributed signaling, the post-failure restoration follows the same procedure. We present a simple complexity analysis of our distributed signaling in Section 6.7.1, and some examples in Section 6.7.2 which would also include simulation results for tree construction times.

6.7.1. Complexity Analysis

In this section, we examine the complexity of our signalling algorithm, an important factor that determines the scalability of our proposed scheme. The complexity of the algorithm must be examined at three stages:

1. Selection of root node(s): Suppose the AAC node placement method is used for root selection. The initiating node could compute its criteria parameter (Average Adjacent Capacity) in $O(D)$ time, and then broadcast its candidacy along with the criteria parameter to the network. Each node with a better criteria parameter (e.g. Average Adjacent Capacity) would broadcast a response to challenge the candidacy. In worst case, the best candidate for the root node is the last node that receives the broadcast and then broadcasts his response back to the network. Assuming a BFS broadcast mechanism, this process would take an order of $O(M+N+D)$ time.

2. Tree ID label processing: This part gives an indication of the amount of local processing at each node at the time of label arrival. Here the newly received Tree ID

must be verified for validity and loop-elimination. The process depends on the length of ID, which indicates the depth of the tree at this node. It is bounded by O(N-1), but more likely to be closer to O(log_2N) for a hierarchical binary tree. Other elements of label processing such as label capacity comparison and response generation are generally independent of network size.

3. Tree Growing: This part indicates the time it takes to form the tree based on propagation of tree ID labels across the network. The algorithm works in a similar manner to the BFS search, which has a worst-case time complexity of O(M) [BaHu89]. In our case, however, a node can be visited more than once by new tree ID labels. Because the maximum possible nodal degree in the network is N-1, the worst-case complexity for tree growing is increased to O(NM) [ZhSh02]. This is an upper bound though, similar to the worst-case scenario we described in Section 3.2.4. In practice we expect the distributed algorithm would typically converge within $O(M)$ time if a proper node placement method was used for the given network.

6.7.2. Examples

We implemented the proposed distributed algorithm using OPNET/C and used OPNET modeler to construct the HP-tree in some sample networks. In all cases the algorithm converged successfully. Figures 6-4 and 6-5 show HP-trees constructed in two graphs based on France and US long haul networks. The tree links are identified in red, while the non-tree links are in black. The tree label ID next to each node identifies its position in the HP-tree. The node with tree label ID of 1 is the root node. In both cases the HP-trees have been constructed using the AAC node placement method.

The tree construction time here depended on three factors: propagation delay, message processing time, and message transmission time. The tree building messages in Figure 6-1 are of small size, likely a few bytes only, and so transmission delay was considered negligible in our model. In our OPNET model we assumed a constant message processing time for each message. This processing time includes the time to retrieve the message from the buffer and processing its command. A second processing time element was considered for putting together a message for the next node. For simplicity and without loss of generality, we assumed both processing times to be the same.

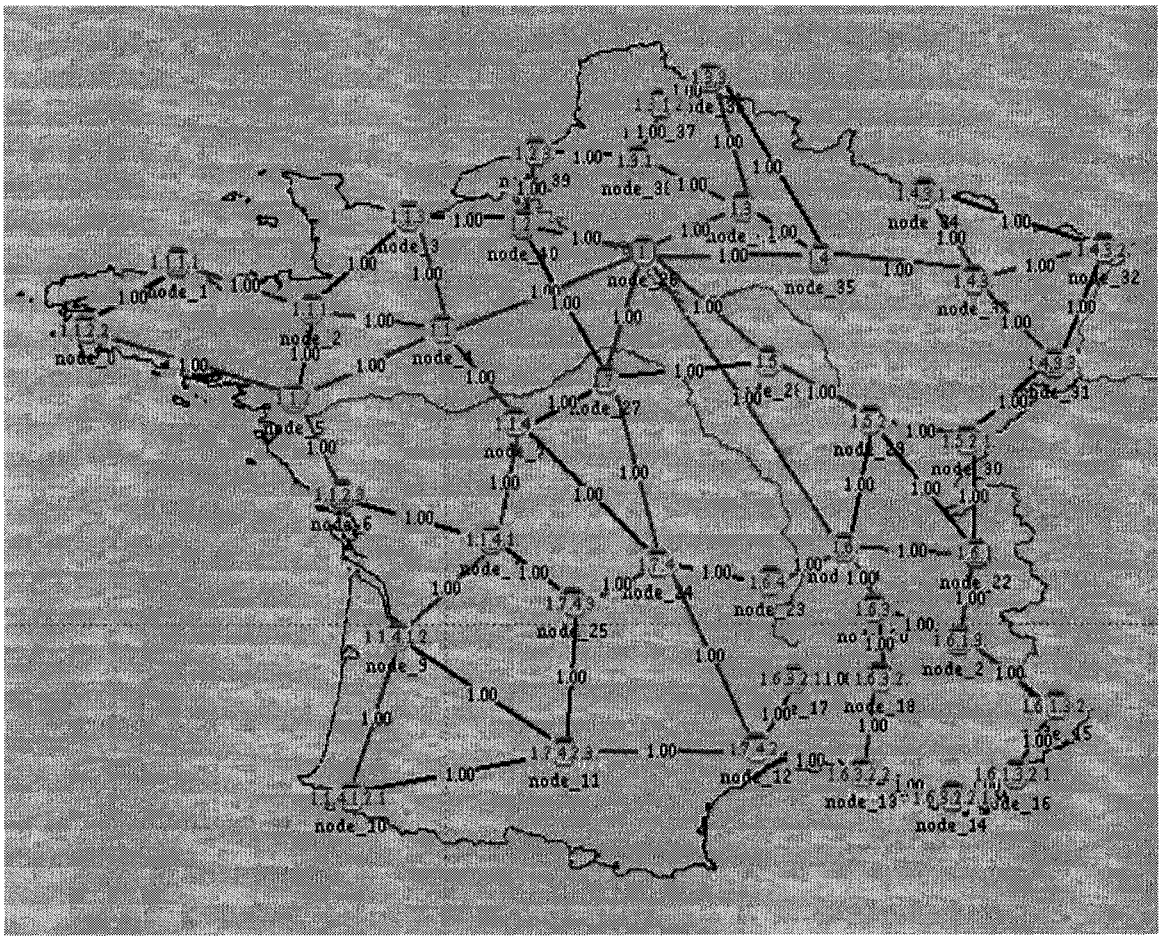


Figure 6-4: Distributed HP-tree in a France-based Network

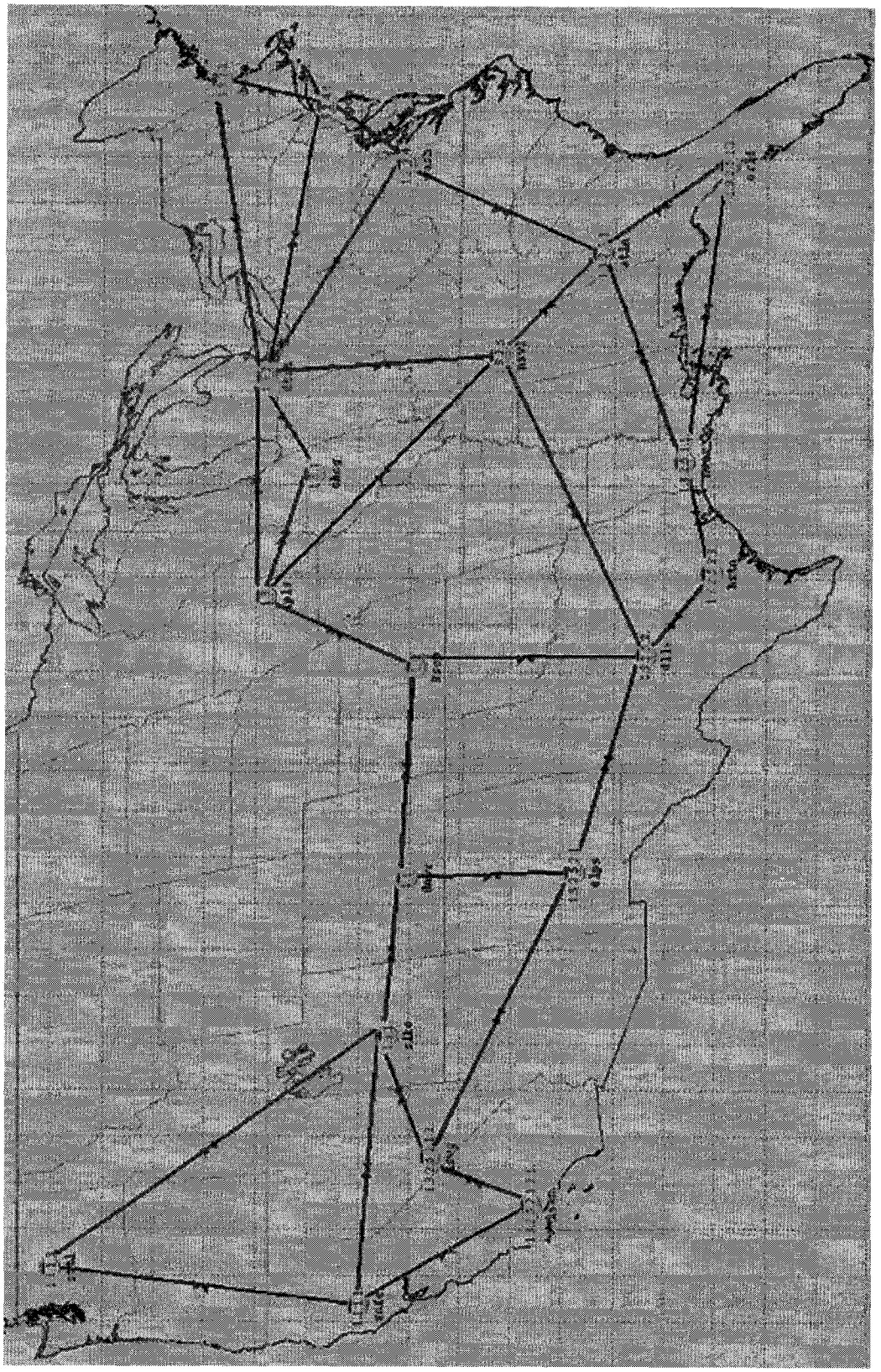


Figure 6-5: Distributed HP-tree in a US-based Network

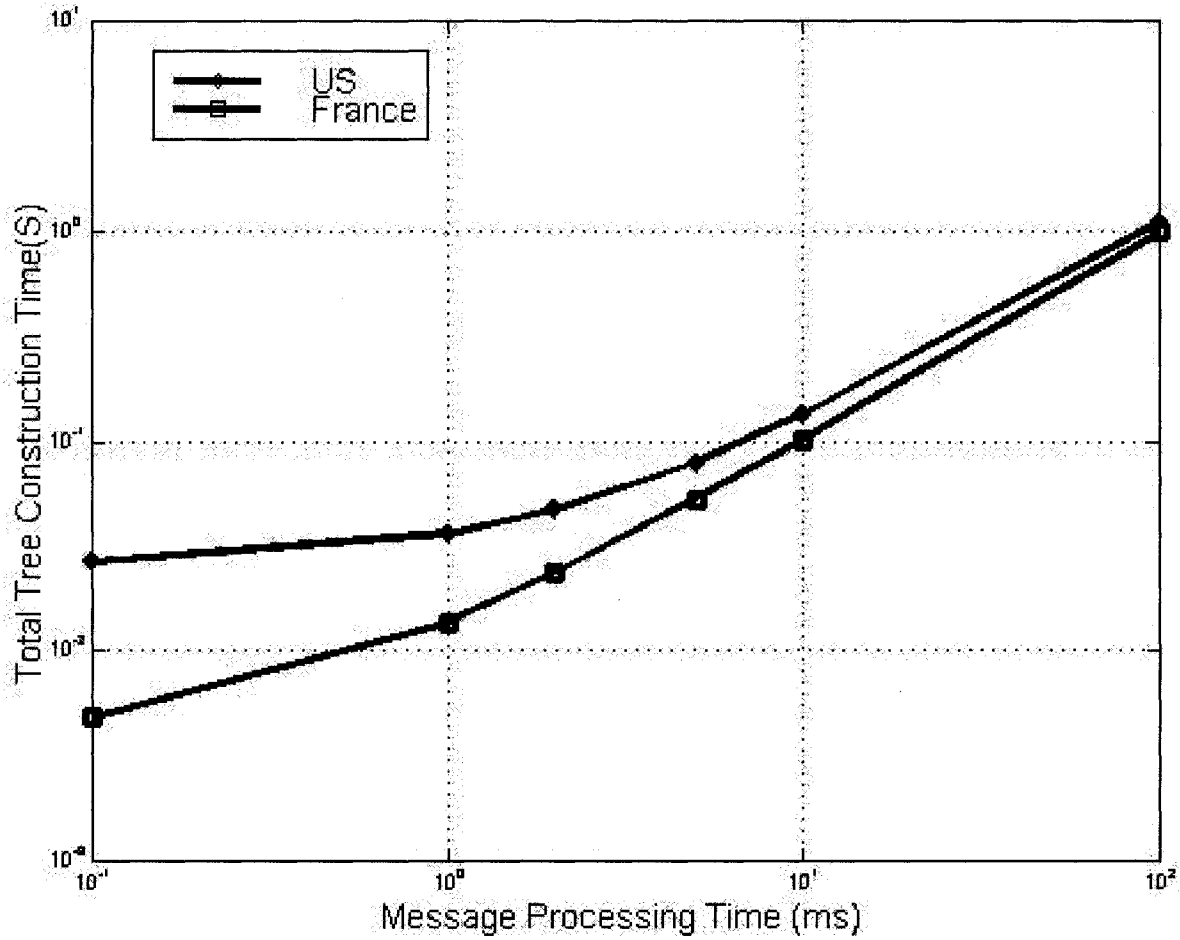


Figure 6-6: Tree construction time in the US and France networks

Figure 6-6 shows the total tree construction time for the France network (Figure 6-4) and the US network (Figure 6-5). Note that the algorithm is running continuously in order to react to any changes in the network. Therefore the total construction time was computed till the time that no more events were generated and the signaling ceased. The model uses actual propagation delays based on the distance. The overall impact of the propagation delay is to add a constant value (depending on the depth of the tree) to the total tree construction time. The message processing time has a direct impact on the total time it takes to construct the tree. In Figure 6-6 we computed the tree construction times for message processing delays from 100 microseconds to 100 milliseconds, however the currently available processing power might allow much smaller delays. It would depend on how the message-processing module is implemented in a switching node (hardware or

software) and at which layer the processing is done. As Figure 6-6 shows when the processing time is reduced, tree construction times would practically come down to the propagation times from the root to the leaves of the HP-tree. This propagation time was about 25 milliseconds for the US network, and about 3.7 milliseconds for France backbone network.

6.8. Concluding Remarks

In this chapter we provided a signalling mechanism for construction of the HP-tree in a distributed manner and update or modification of the protection tree if the topology of the network changes. Our study indicated that such implementation was feasible by extension of protocols and algorithms already in use in today's networks, and that the protocol converges to a point where every node identifies itself in the tree by a tree ID label. We discussed how the nodes can handle failure recovery and addition and removal of network links and nodes in a distributed manner. We also proposed methods for dealing with root node failure.

Chapter 7. Conclusion

In this thesis we have provided algorithms and detailed performance results for a novel unicast protection approach for mesh networks based on a class of self-repairing hierarchical-protection spanning trees. The main objective was to propose a shared protection mechanism based on link restoration. The proposal would allow sharing of the spare capacity, and more importantly, could be updated and managed based on the requirements of dynamic network operation. First the core concept of hierarchical-protection trees was introduced and the basic operations, characteristics, limitations and advantages of the scheme were presented. We showed how a self-repairing mechanism could be included in the protection tree by introducing primary and backup parent nodes for each node and by updating them using adjacent non-tree links as the state of the network topology changes. We also showed how such approach could handle multiple failures with minimal changes in the protection paths across the network.

Next we studied the spare capacity assignment (SCA) problem and pre-designed network cases for design of the HP-tree. We proposed heuristic algorithms for constructing the HP-tree in different scenarios, and introduced the idea of HP-tree node placement methods using four different node sorting criteria. For performance evaluation we constructed a large database of random mesh networks of different sizes and nodal degrees using the most relevant models in the literature. We also proposed models for working and spare link capacities in random networks in different scenarios. Using our database of random networks, we conducted extensive performance simulations of HP-trees and identified trends in performance parameters such as restorability, redundancy and average backup path length versus network size, nodal degree, radius and other topological parameters. We studied the topological constraints of the HP-tree protection scheme and linked it to specific deficiencies in the topology, confirming it using simulation results. We also showed that the HP-tree could provide feasible backup protection path lengths even in very large networks. We further studied the performance of HP-tree in several networks based on real commercial samples, and showed how the restorability of our proposed scheme was related to total network load and maximum link utilization.

We also conducted extensive performance study of the HP-tree scheme under various multiple failure scenarios. We proposed recursive computation methods for calculation of multiple failure restorability and redundancy requirements in various cases. We studied the restorability and redundancy requirements of the HP-tree under double and triple failure scenarios, based on working-only and all-capacity protection models, and showed that if the required spare capacity was allocated, the HP-tree was able to achieve full restorability in most cases with minimal self-repairing backup path changes.

Finally, we designed a comprehensive signalling mechanism, with detailed description of message formats, sequences, recovery mechanisms etc, to construct the tree in a distributed fashion. The operation and management of the HP-tree in a distributed environment under different scenarios were discussed, and we showed that our approach was feasible, scalable and manageable.

In summary, our research in this thesis provided a comprehensive package on HP-trees, from concept and operation to design formulation, heuristic tree construction algorithms, extensive performance evaluation in single and multiple failure scenarios, as well as implementation details in distributed real networks.

7.1. Future Work

This thesis establishes the new concept of HP-tree for unicast link restoration, and as such, a whole set of new venues could be pursued to further extend the results in this thesis. The signalling mechanism provided in this thesis could be implemented using commercial protocols. The spare capacity assignment problem that we studied in this thesis could be further extended in form of a joint capacity assignment problem for the HP-tree in which the working and protection capacity assignments are optimized together. This problem would basically address a static case where the pair-wise demands between nodes are given, and the designer will combine the problem of demand routing and protection assignment to achieve higher efficiencies. Results in the literature (e.g. [IrMa98]) indicate that the joint capacity assignment solution would increase the efficiency of capacity sharing in the network and thus reduces the spare capacity requirement. The joint capacity assignment formulation for HP-tree is one of our planned future research works

Another issue for further research includes study of optimized network topology design for maximizing the efficiency of the HP-tree. We have indicated some design rules in Chapter 2 and 3 of this thesis to avoid topological constraints of the HP-tree as much as possible. This approach could be expanded further into a topology design algorithm that would deliver an optimized network design for full restorability and minimum redundancy.

In this thesis we examined four different node placement criteria, and our research only provided an initial insight into how to choose the best method for a given scenario; e.g. selection of capacity-based methods for fixed capacity scenarios and ND and DIST method for spare capacity minimization. A future research goal would be to study in more detail the relationship between the topological characteristics of a network and the best method for node sorting in that network. A broader node placement algorithm that would take specific characteristics of network into account in node placement is another future research goal in the area of HP-tree restoration.

Also related to the node sorting problem is the impact of tie-breaking when, for a given node placement method, two nodes score the same. In our approach we relied on node indices (node numbers) to break the tie; i.e. the node with a lower index would be selected if all other parameters were alike. This approach is similar to the standard IEEE STP protocol for bridges. However our results indicate that the numbering of nodes (which acted as the tie-breaking rule in our algorithm) has a notable impact on the performance of the constructed tree. The study of this effect is also a future research objective.

Finally, as mentioned in Chapter 1 the PWCE concept could facilitate the control of protection paths in network under dynamic changes. The envelopes in this method could be constructed using different techniques, including HP-tree. Studying the performance of HP-trees in such environment would be another future research objective on the subject of tree-based protection.

References

[Ayan02] S. Ayandeh, "Convergence of Protection and Restoration in Telecommunication Networks", *Photonic Network Communications*, vol. 4, no. 3/4, 2002, pp. 237-250.

[BaFr93] A. J. Ballardie, P.F. Francis and J. Crowcroft, "Core-based trees", *Proceedings of ACM SIGCOMM'93*, San Francisco, CA, USA, Sept. 1993, pp. 85-95.

[BaDr01a] A. Banerjee, J. Drake, J. P. Lang, B. Turner, K. Kompella, Y. Rekhter, "Generalized Multi-protocol Label Switching: An Overview of Routing and Management Enhancements", *IEEE Communications Magazine*, Jan 2001, P144-150.

[BaDr01b] A. Banerjee, J. Drake, J. P. Lang, B. Turner, D. Awduche, L. Berger, K. Kompella, Y. Rekhter, "Generalized Multi-protocol Label Switching: An Overview of Signaling Enhancements and Recovery Techniques" IEEE Communications Magazine, July 2001, P144-151.

[BuHa89] F. Buckley and F. Harary, "Distance in Graphs", Addison-Wesley Publishing Company, 1989.

[CaDo97] K. Calvert, M. Doar and E. Zegura, "Modeling Internet Topology", *IEEE Communication Magazine*, June 1997, pp. 160-163.

[CaZe95] K. Calvert, E. Zegura and M.J. Donahoo, "Core Selection Methods for Multicast Routing", *Proc. of IEEE conference on Computer Communication and Networks*, 1995, pp. 638-642.

[ChJa05] P. Cholda and A. Jajszczyk, "Reliability assessment of rings and p-cycles in DWDM networks", *IEEE Conference on Next Generation Internet Networks*, 18-20 April 2005, pp. 364 - 371.

[ClGr02] M. Clouqueur and W.D. Grover, "Availability Analysis of Span-Restorable Mesh Networks", *IEEE Journal of Selected Areas in Communications*, vol. 20, no. 4, May 2002, pp. 810-821.

[ClGr05] M. Clouqueur and W.D. Grover, "Mesh-Restorable Networks with Enhanced Dual-Failure Restorability Properties", *Photonic Network Communications*, vol. 9, no. 1, 2005, pp. 7-18.

[DoGr01a] J. Doucette, W.D. Grover and T. Bach, "Bi-criteria studies of mesh network restoration path-length versus capacity tradeoffs", *Proceedings of OSA Optical Fiber Communication Conference and Exhibit (OFC 2001)*, March 17-22, 2001, Anaheim, California.

[DoGr02] J. Doucette, W.D. Grover, "Maintenance-Immune Design of Span-Restorable Mesh Networks," *Proceedings of the 18th Annual National Fiber Optics Engineers Conference (NFOEC 2002)*, Dallas, TX, September 2002, pp. 2049-2061.

[DoHe03] J. Doucette, D. He, W.D. Grover and O.Yang, "Algorithmic Approaches for Efficient Enumeration of Candidate p-Cycles and Capacitates p-Cycle Network Design", *Proceedings of the Fourth International Workshop on the Design of Reliable Communication Networks (DRCN'03)*, 19-22 October 2003, Banff, Alberta, Canada, pp. 212-220.

[DoMo94] R.D. Doverspike, J.A. Morgan and W. Leland, "Network Design Sensitivity Studies for Use of Digital Cross-Connect Systems in Survivable Network Architectures", *IEEE Journal on Selected Areas in Communications*, vol.12, no.1, January 1994. pp 69-78.

[Douc05] J. Doucette, "Advances on Design and Analysis of Mesh-Restorable Networks", Ph.D. Thesis, University of Alberta, Spring 2005.

[DuGr94] D.A. Dunn, W.D. Grover and M.H. MacGregor, "Comparison of k-Shortest Paths and Maximum Flow Routing for Network Facility Restoration", *IEEE Journal on Selected Areas in Communications*, vol. 12, no. 1, January 1994, pp. 88-99.

[Elli00] G. Ellinas et al, "Protection Cycles in Mesh WDM Networks", *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 10, October 2000, pp. 1924-1937.

[EsTa76] K.P. Eswaran and R.E. Tarjan, "Augmentation Problems", *SIAM J. Computing*, Vol. 5, No.4, December 1976, pp. 653-665

[FiMe97] S.G. Finn, M.M. Medard, R.A. Barry, "A Novel Approach to Automatic Protection Switching Using Trees", *Proceedings of IEEE ICC'97*, Montreal, Quebec, Canada, June 1997, pp.272-276.

[GaHe94] L. M. Gardner, M. Heydari et al, "Techniques for finding ring covers in survivable networks", *Proc. of IEEE GLOBECOM'94*, 28 Nov. – 2 Dec. 1994, pp. 1862-1866.

[GrBi91] W.D. Grover, T.D. Bilodeau and B.D. Venables, "Near Optimal Spare Capacity Planning in a Mesh Restorable Network", *Proceedings of IEEE Globecom'91*, Phoenix, Arizona, USA, Dec. 1991, vol.3, pp. 2007-2012.

[GrDo01] W.D. Grover and J. Doucette, "Increasing the Efficiency of Span-Restorable Networks on Low-connectivity Graphs," *Proceedings of the Third International Workshop on the Design of Reliable Communication Networks (DRCN'01)*, 7-10 October 2001, Budapest, Hungary, pp. 99-106.

[Grov87] W.D. Grover, "The Selfhealing Network: A fast distributed restoration technique for networks using digital cross-connect machines", *Proc. IEEE Global Conference on Communications*, Tokyo, 1987, pp. 1090-1095.

[Grov94] W.D. Grover, "Distributed Restoration of the Transport Network," in Telecommunications Network Management into the 21st Century, IEEE Press, 1994, pp. 337-417.

[Grov03] W.D. Grover, "Mesh-based Survivable Transport Networks: Options and Strategies for Optical, MPLS, SONET and ATM Networking", ISBN 013494576X, Prentice-Hall, 2003.

[GrSt98] W.D. Grover, D. Stamatelakis: Cycle-Oriented Distributed Preconfiguration: Ring-like speed with mesh-like capacity for self-planning Network Restoration", *Proceedings of IEEE ICC'98*, Atlanta, GA, USA, June 1998, vol.1, pp. 537-543.

[GrVe90] W. Grover, B. Venables et al "Performance Studies of a Selfhealing Network Protocol in Telecom Canada Long Haul Network", *Proc of IEEE GLOBECOM'90*, San Diego, CA, 1990.

[GrVe91b] W.D. Grover, B.D. Venables, "Performance of the Selfhealing Network Protocol with Random Individual Link Failures", *Proceedings of IEEE International Conference on Communications (ICC '91)*, Denver, CO, USA, June 1991, vol.2, pp.660-666.

[GrYe99] J. Gross and J. Yellen, "Graph Theory and its Applications", ISBN 0-8493-3982-0, CRC Press, New York, 1999.

[HeBy94] M. Herzberg and S. J. Bye, "An Optimal Spare Capacity Assignment Model for Survivable Networks with Hop Limits", *Proceedings of IEEE Globecom'94*, Nov. 27 - Dec. 1, 1994, San Francisco, CA, USA, pp. 1601-1606.

[HoMo04] P-H. Ho and H.T. Mouftah, "Reconfiguration of Spare Capacity for MPLS-Based Recovery in the Internet Backbone Networks", *IEEE/ACM Transactions On Networking*, Vol. 12, No. 1, February 2004, pp. 73-84.

[HuCo02] H. Huang and J.A. Copeland, "A Series of Hamiltonian Cycle-Based Solutions to provide Simple and Scalable Mesh Optical Network Resilience," *IEEE Communication Magazine*, November 2002, pp. 46-51.

[IEEE04] IEEE 802.1D Standard on MAC bridges, 2004.

[IrMa98] R.R. Iraschko, M.H. MacGregor, W.D. Grover, "Optimal Capacity Placement for Path Restoration in STM or ATM Mesh Survivable Networks", *IEEE/ACM Transactions on Networking*, Vol. 6, No. 3, June 1998, pp 325-336.

[Ishi97] K. Ishida, "Multiple Node-Disjoint Paths protocol for virtual Path in ATM Networks", *Proceedings of the joint workshop on Parallel and Distributed Real-Time Systems*, Geneva, Switzerland, April 1997, pp. 91-97.

[ItRo88] A. Itai and M. Rodeh, "The Multi-tree approach to reliability in distributed networks", *Information Computing*, vol. 79, 1988, pp 43-49.

[KiJe96] H.J. Kim, G-I. Jee and J.G. Lee, "Optimal Load Distribution for Tree Network Processors", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 32, no. 2, April 1996, pp. 607-612.

[KiLu03] Sun-il Kim and S. S. Lumetta, "Evaluation of Protection Reconfiguration for Multiple Failures in Optical Networks", *Proceedings of OFC 2003.*

[Knut97] D. Knuth, "The Art of Computer Programming, Volume 3: Sorting and Searching", 3rd Edition. Addison-Wesley, 1997. ISBN 0-201-89685-0.

[KoSa04] A. Kodian, A. Sack and W.D. Grover, "p-cycle network design with hop limits and circumference limits", *BroadNets 2004*, Page(s):244 – 253.

[LeGr02] S.K. Lee, D. Griffith and N.-O. Song, "An Analytical Approach to Shared Backup Path Provisioning in GMPLS Networks", *Proc. of IEEE MILCOM 2002*, 7-10 Oct. 2002, vol.1, pp. 75-80.

[LiTi05] Y. Liu, D. Tipper, and P. Siripongwutikorn, "Approximating Optimal Spare Capacity Allocation by Successive Survivable Routing", *IEEE/ACM Transactions On Networking*, Vol. 13, No. 1, February 2005, pp.198-211.

[LiYa03] H. Liu, O. Yang and S. Shah-Heydari, "A dual tree-based link protection algorithm", *Proc. Canadian Conference on Electrical and Computer Engineering (CCECE'2003)*, May 2003, Montreal, Canada, pp 939-942.

[LuMe00] S.S. Lumetta, M.M. Médard, and Y-C. Tseng, "Capacity Versus Robustness: A Tradeoff for Link Restoration in Mesh Networks", *Journal Of Lightwave Technology*, Vol. 18, No. 12, December 2000, pp. 1765-1775.

[LuMe01] S.S. Lumetta, M.M. Medard, "Towards a Deeper Understanding of Link Restoration Algorithms for Mesh Networks", *Proceedings of IEEE INFOCOM 2001*, Anchorage, AL, USA, April 2001, vol.1, pp. 367-375.

[LuMe01a] S. Lumetta and M.. Medard, "Classification of two-link failures for all-optical networks", *Proceedings of the Optical Fiber Communication Conference*, Anaheim, California, March 2001. TuO3

[MaGr93] M.H. MacGregor, W.D. Grover, U.M. Maydell, "Connectability: A performance metric for reconfigurable networks," *IEEE Journal on Selected Areas in Communications*, Dec. 1993, vol. 11, no. 9, pp. 1461-1469.

[Mapnet] MAPNET web site, http://www.caida.org/tools/visualization/mapnet/

[Matl70] MATLAB® version 7.0B, Mathworks, 2004.

[Mats93] T. Matsui, "An Algorithm for Finding All the Spanning Trees in Undirected Graphs", *Technical Report METR 93-08*, Department of Mathematical Engineering and Information Physics, University of Tokyo, Tokyo, Japan, 1993.

[Mauz02] C. Mauz, "Dimensioning of a Transport Network Being Robust to Changes of the Traffic Pattern", Proceedings of ICTON2002, Warsaw, Poland, April 2002.

[MeBa99] M. Medard et al, "Redundant Trees for Preplanned recovery in Arbitrary Vertex-redundant or Edge-Redundant Graphs," *IEEE/ACM Transactions on Networking*, vol. 7, no. 5, October 1999, pp. 641-651.

[MeBa02] M. Medard et al, "Generalized Loop-Back Recovery in Optical Mesh Networks", *IEEE/ACM Transactions On Networking*, Vol. 10, No. 1, February 2002, pp. 153-164.

[Mori98] T. Mori et al, "Ultra High-Speed SONET Fiber-Optic Transmission System", *Hitachi Review*, vol. 47, no. 2, April 1998, pp. 79-84.

[MuKi98] K. Murakami and H.S. Kim, "Optimal Capacity and Flow Assignment for Self-Healing ATM Networks Based on Line and End-to-End Restoration", *IEEE/ACM Transactions on Networking*, vol. 6, no. 2, April 1998, pp. 207-221.

[NaGu97] D. Naor, D. Gusfield and C. Martel, "A Fast Algorithm for Optimally Increasing the Edge Connectivity", *SIAM J. Computing*, vol. 26, no. 4, August 1997, pp. 1139-1165.

[Opne70] OPNET Modeler v.7.0.B User Manual, OPNET Technologies Inc.

[Penr03] M. Penrose, "Random Geometric Graphs," Oxford University Press, 2003.

[Perl85] R. Perlman, "An algorithm for distributed computation of a spanning tree in an extended LAN", *Proceedings of the 9th Data Communications Symposium*, September 1985) ACM/IEEE, New York, 1985,44-53.

[PeSt93] S. Peng, A.B. Stephens and Y. Yesha, "Algorithms for a Core and k-Tree Core of a Tree", *Journal of Algorithms,* vol. 15, 1993, pp. 143-159.

[Rama96] S. Ramanathan, "Multicast Tree Generation in Networks with Asumemetric Links", *IEEE/ACM Transaction on Networking*, vol. 4, no. 4, August 1996, pp. 558-568.

[RaRa96] A.D. Raghavendra, S. Rai and S.S. Iyengar, "Multicast Routing in Internetworks Using Dynamic Core Based Trees", *Proc Int. Conf. Computers and Communications*, 1996.

[RaMu99a] S. Ramamurthy and B. Mukherjee, "Survivable WDM mesh networks: Part I. Protection", *Proceedings of IEEE INFOCOM'99*, Ney York, NY, USA , 21-25 March 1999, vol.2, pp 744-751.

[RaMu99b] S. Ramamurthy and B. Mukherjee, "Survivable WDM mesh networks: Part II. Restoration", *Proceedings of IEEE International Conference on Communications (ICC'99)*, Vancouver, BC, Canada, 6-10 June 1999, vol.3, pp. 2023-2030.

[Rose02] E. Rosenberg, "Capacity Requirements for Node and Arc Survivable Networks", *Telecommunication Systems*, Vol. 20, No. 1/2, 2002, pp. 107-131.

[SaNi90] H. Sakauchi, Y. Nashimura and S. Hasegawa, "A Self-Healing Network with an Economical spare Channel Assignment", *Proceedings of GLOBECOM 90*, pp. 438-443.

[ScCh00] L. Schwiebert and R. Chintalapati, "Improved Fault Recovery for Core Based Trees", *Computer Communications*, vol.23, no.9, Apr 2000.

[ScGr02] D.A. Schupke, C.G. Gruber, A. Autenrieth, "Optimal Configuration of *p*-Cycles in WDM Networks", *Proc. ICC 2002*, pp. 2761-2765.

[ScGr04] D.A. Schupke, W.D. Grover, M. Clouqueur, "Strategies for Enhanced Dual Failure Restorability with Static or Reconfigurable p-Cycle Networks," *IEEE International Conference on Communications (ICC)*, Paris, France, June 20-24, 2004.

[Schu03] D. A. Schupke, "Multiple Failure Survivability in WDM networks with p-Cycles", *Proceedings of the 2003 International Symposium on Circuits and Systems*, 25-28 May 2003, vol.3, pp. 866-869.

[Schu03a] D.A. Schupke, "The Tradeoff Between the Number of Deployed p-Cycles and the Survivability to Dual Fiber Duct Failures," *IEEE International Conference on Communications (ICC)*, Anchorage, AK, USA, May 11-15, 2003.

[Schu04] D.A. Schupke, "An ILP for Optimal p-Cycle Selection without Cycle Enumeration," *Eighth Working Conference on Optical Network Design and Modelling (ONDM)*, Ghent, Belgium, February 2-4, 2004.

[Schu04a] D.A. Schupke, "Comparison of p-Cycle Configuration Methods for Dynamic Networks", *IFIP Optical Networks & Technologies Conference (OpNeTec)*, Pisa, Italy, October 18-22, 2004.

[Schu05] D.A. Schupke, "Cycle-Based Protection for Optical Transport Networks," Dissertation, ISBN 3831604630, Herbert Utz Verlag, 2005.

[ScPr04] D.A. Schupke and S. Prinze, "Capacity Efficiency and Restorability of Path Protection and Rerouting in WDM Networks Subject to Dual Failures", *Photonic Network Communications*, vol 8, no 2, 2004, pp 191-207.

[SeBa91] A. Segall, T. Barzilai, Y. Ofek, "Reliable Multiuser Tree Setup with Local Identifiers", *IEEE Journal on Selected Areas in Communications*, vol.9, no.9, Dec 1991, pp. 1427-1439.

[ShGr04] G. Shen and W.D. Grover, "Performance of Protected Working Capacity Envelopes based on p-Cycles: Fast, Simple, and Scalable Dynamic Service Provisioning of Survivable Services", *Proc of SPIE APOC 2004,* Beijing, China, Nov. 7–11, 2004, vol. 5626, pp. 519-533.

[Shon96] C-S. Wu et al, "A New Preplanned Self-Healing Scheme for Multicast ATM Network", *Proceedings of International Conference on Communication Technology (ICCT'96)*, Beijing, China, 5-7 May, 1996, pp. 888-891.

[ShTa97] A. Shioura, A. Tamura and T. Uno, "An Optimal Algorithm for Scanning all Spanning Trees of Undirected Graphs," *SIAM Journal on Computing*, vol. 26, no. 3, 1997, pp. 678-692.

[ShYa01] S. Shah-Heydari and O. Yang, "A Tree-based algorithm for Protection &Restoration in Optical Mesh Networks", *Proc. of Canadian Conference*

on Electrical and Computer Engineering 2001, Toronto, Canada, pp. 1169-1173.

[ShYa04a] S. Shah-Heydari and O. Yang, "Hierarchical Protection Tree Scheme for Failure Recovery in Mesh Networks", *Photonic Networks Communication Journal*, vol. 7, no. 2, 2004, pp. 145-159.

[ShYa04b] S. Shah-Heydari and O. Yang, "Maximizing Restorability of Hierarchical Protection Tree in Survivable Mesh Networks", *Proc. of International Conference on Computing, Communications and Control Technologies (CCCT 2004)*, 14-17 August 2004, Austin, Texas, USA.

[ShYa04c] S. Shah-Heydari and O. Yang, "Multiple Failure Analysis with Restoration Paths Matrix", *Proc. of IEEE Globecom 2004*, 29 November – 3 December 2004, Dallas, Texas, USA.

[SoPi02] S. Soni and H. Pirkul, "Design of Survivable Networks with Connectivity Requirements", *Telecommunication Systems*, Vol. 20, No.1/2, 2002, pp. 133-149.

[Stam97] D. Stamatelakis, "Theory and algorithms for preconfiguration of spare capacity in mesh survivable networks", M.Sc. thesis, University of Alberta, 1997.

[StGr00] D. Stamatelakis and W.D. Grover,"Theoretical Underpinnings for the Efficiency of Restorable Networks Using Preconfigured Cycles (p-cycles)," *IEEE Transactions on Communications*, vol. 48, no. 8, August 2000, pp. 1262-1265.

[StGr00a] D. Stamatelakis and W.D. Grover, "Network Restorability Design using Pre-Configured Trees, Cycles and Mixture of Pattern Types", *TRLabs Technical Report TR-1999-05*, October 30, 2000.

[StGr00b] D. Stamatelakis and W.D. Grover, "IP Layer Restoration and Network Planning Based on Virtual Protection Cycles", *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 10, October 2000, pp. 1938-1949.

[T1A193] "A Technical Report on Network Survivability Performance", prepared by T1A1.2 Working group on Network Survivability Performance, Report No. 24, November 1993.

[Tarj74] R. Endre Tarjan, "A Note on Finding the Bridges of a Graph", *Information Processing Letters 2 (1974)*, pp. 160-161.

[TaRu05] F. Tang and L. Ruan, "A Protection Tree Scheme for First-Failure Protection and Second-Failure Restoration in Optical Networks", *Proc. of ICCNMC 2005*, LNCS 3619, pp. 620 – 631, 2005.

[UnJa01] L. Jereb, F. Unghvry and T. Jakab, "A Methodology for Reliability Analysis of Multi-Layer Communication Networks", *Optical Networks Magazine*, September/October 2001, pp. 42-50.

[VeGr93] B. Venables, W.D. Grover, M.H. MacGregor, "Two strategies for Spare Capacity Placement (SCP) in Mesh Restorable Networks", *Proceedings of*

IEEE International Conference on Communications (ICC'93), Geneva, Switzerland, May 1993, vol.1, pp. 267-271.

[WaCh98] S-F. Wang and R.-F. Chang, "Self-Healing on ATM Multicast Tree", *IEICE Transactions on Communications*, vol. E81-B, no. 8, August 1998, pp. 1590-1598.

[Waxm88] B.M. Waxman, "Routing of Multipoint Connections", *IEEE Journal on Selected Areas in Communications*, vol. 6, no. 9, December 1988, pp.1617-1622.

[WeEs94] L. Wei and D. Estrin, "A Comparison of Multicast Trees and Algorithms", *Proc. IEEE INFOCOM'94*, Toronto, Canada, 1994.

[Wint87] P. Winter, "Steiner Problem in Networks: A Survey", *Networks*, vol. 17, 1987, pp. 129-167.

[Wu92] T-H. Wu, "Fiber Network Service Survivability", Artech House, Boston, MA, USA, 1992.

[Wu95] T-H. Wu, "Emerging technologies for Fiber Network Survivability", *IEEE Communication magazine*, vol. 33, no. 2 (Feb. 1995), pp 58-74.

[WuCh04] B. Y. Wu and K-M. Chao, "Spanning Trees and Optimization Problems", Chapman & Hall/CRC, 2004.

[XiMa99] Y. Xiong and L. Mason, "Restoration Strategies and Spare Capacity Requirements in Self-Healing ATM Networks", *IEEE/ACM Transactions on Networking,* vol. 7, no. 1, February 1999, pp. 98-110

[XuTh02] G. Xue, L. Chen, K. Thulasiraman, "Delay Reduction in Redundant Trees for Pre-Planned Protection against Single Link/Node Failure in 2-Connected Graphs". *Proc. of IEEE GLOBECOM2002* (Optical Networking Symposium), 2002, pp. 2691-2695.

[XuTh03] G. Xue, L. Chen, K. Thulasiraman, "Quality-of-Service and Quality-of-Protection Issues in Preplanned Recovery Schemes Using redundant Trees", *IEEE Journal of Selected Areas in Communications*, vol. 21, no.8, October 2003, pp. 1332-1345.

[YaFi04] Z. Yao, J. Fitchett and K. Felske, "Fast-Unreserved Failure Restoration for Meshed Intelligent Photonic Networks", *Photonic Network Communications*, vol.8, no.1, 2004, pp. 105-117.

[YaMa86] O.W.W. Yang, J.W.Mark, "Design Issues in Metropolitan Area Networks", IEEEICC'86 Conf. Proceedings, pp.899-903, Toronto, Ontario, Jun 1986.

[Yang92] O.W.W. Yang, "Terminal-Pair Reliability of Tree-Type Computer Communication Networks", *IEEE Transaction on Reliability*, vol. 41, no. 1, March 1992, pp. 49-56.

[Yang94] O.W.W. Yang, "Two-Center Tree Topologies for Metropolitan Area Network", *IEE Proceedings-I: Communications, Speech and Vision*, pp.280-288, Vol.141, No.4, Aug. 1994.

[YeOf94] B. Yener, Y. Ofek and M. Yung, "Design and Performance of Convergence Routing on Multiple Spanning Trees", *Proceedings of IEEE Globecom'94*, Nov. 27 - Dec. 1, 1994, San Francisco, CA, USA, pp. 169-175.

[ZaFa02] D. Zappala, A. Fabbri and V. Lo, "An Evaluation of Shared Multicast Trees with Multiple Cores", *Telecommunication Systems*, vol 19:3,4, 2002, pp. 461-479.

[ZeIt89] A. Zehavi and A. Itai, "Three Tree-Paths", *J. Graph Theory (13)*, 1989, pp.175-188.

[ZeCa97] E. Zegura, K. Calvert and M. Donahoo, "A Quantitative Comparison of Graph-Based Models for Internet Topology", *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, December 1997, pp. 770-783.

[Zhan02a] Y. Zhang, "Distributed tree schemes for DWDM network protection and restoration", M.Sc. thesis, University of Ottawa, 2002.

[Zhan02b] H. Zhang, "DWDM Network Protection/Restoration with Ring/Cycle Topologies", M. Eng thesis, University of Ottawa, Ottawa, Canada, 2002.

[Zhan04] H. Zhang et al, "A Cycle-Based Rerouting Scheme for Wavelength-Routed WDM Networks", *Proceedings of the International Conference on Parallel Processing Workshops (ICPPW'04)*, 2004.

[ZhSh02] Y. Zhang, S. Shah-Heydari, O. Yang, "Complexity Analysis of Hierarchical Tree Algorithm", *Proceedings of 21st Biennial Symposium on Communication, 2002*, Kingston, Canada, pp. 46-50.

[ZhYa02a] Y. Zhang, O. Yang, "A distributed tree algorithm for WDM network protection/restoration", *Proc. of IEEE International Conference on High Speed Networks and Multimedia Communications HSNMC'02*, Jeju Island, Korea, July 2002, pp. 289 –294.

[ZhYa02b] Y. Zhang and O. Yang, "Different implementations of token tree algorithm for DWDM network protection/restoration", *Proceedings of IEEE Conference on Computer Communications and Networks (ICCN 2002)*, Miami, FL, USA, Oct. 2002, pp. 290 –295.

[ZhYa02c] H. Zhang and O. Yang, "Finding protection cycles in DWDM networks", *Proc. of ICC'02*, vol.5, pp. 2756- 2760.

[ZhZh04] J. Zhang, K. Zhu and B. Mukherjee, "A Comprehensive Study on Backup Reprovisioning to Remedy the Effects of Multiple-Link Failures in WDM Mesh Networks", *Proc. of IEEE International Conference on Communications (ICC-04)*, Paris, June 2004, pp. 1654-1658.

Appendix A: Network Topologies

A.1. Real Network Topologies

The network topologies in this section have been constructed based on real commercial networks from the MAPNET database [Mapnet]. The links to hanging nodes (nodes with nodal degree of one) were removed as such nodes would violate the two-edge connectivity requirements of link restoration. The total link capacities were normalized to DS1 or DS3 channel capacity depending on the granularity of the link capacities.

ATT2

Node 1	Node 2	Total Link Capacity
1	2	242
1	4	50
1	5	2
1	10	2
1	16	1
1	19	50
1	20	48
1	21	48
1	22	48
1	30	4
1	31	1
2	4	48
2	22	48
2	23	1
2	26	3
2	30	3
3	4	50
3	5	48
3	26	3
4	5	50
4	8	2
4	9	50
4	12	48
4	19	2
4	21	1
4	23	48
4	24	49
4	28	4
5	6	48
5	8	50
5	9	48
5	10	2

5	11	2
5	16	2
6	7	48
7	8	49
8	10	96
8	27	1
9	10	2
9	12	48
9	17	2
9	19	48
10	11	2
10	12	96
10	14	48
10	15	48
10	16	48
10	17	48
10	18	48
11	12	48
11	14	4
12	13	48
12	25	49
13	14	48
14	27	3
15	18	48
16	17	49
16	18	2
16	19	1
17	19	50
17	28	3
19	20	48
19	29	3
20	29	3
23	31	1
24	25	48

BBN

Node 1	Node 2	Total Link Capacity
1	2	3
1	3	12
1	4	24
1	8	12
2	3	6
3	7	12
3	8	24
3	9	12
4	6	12
4	10	12
5	6	12

5	10	12
5	11	12
6	7	3
8	9	12
8	10	12
8	12	12
9	12	12
10	11	24

Brazil RNP

Node 1	Node 2	Total Link Capacity
1	2	30
1	3	20
2	3	40
2	4	8
2	5	40
2	6	20
2	7	12
2	8	6
2	9	50
2	10	40
2	11	28
3	4	6
3	5	36
3	6	14
3	7	8
3	8	6
3	9	20
3	10	24
3	11	16

Data Exchange

Node 1	Node 2	Total Link Capacity
1	2	2
1	3	6
1	4	6
1	5	6
1	6	6
1	7	6
2	3	6
2	4	6
2	5	6
2	6	6
2	7	6
3	4	6
3	5	6

3	6	6
3	7	6
4	5	6
4	6	6
4	7	6
5	6	6
5	7	6
5	8	6
6	7	6
6	8	6
7	8	6

GoodNet

Node 1	Node 2	Total Link Capacity
1	2	56
1	3	56
1	4	56
2	6	56
2	7	56
2	18	56
3	4	56
3	7	56
3	18	56
4	5	28
4	6	152
4	7	56
4	8	84
4	9	56
4	12	56
4	13	56
4	15	56
4	16	56
4	20	56
4	21	193
4	22	193
4	23	96
5	6	56
5	7	28
6	7	56
6	8	56
6	9	56
6	10	56
6	11	28
6	12	56
6	13	56
6	14	56
6	15	56

6	16	56
6	17	56
6	21	193
6	22	193
6	23	96
7	8	28
7	9	56
7	10	56
7	11	28
7	13	56
7	14	56
7	15	56
7	16	56
7	17	56
7	20	56
7	21	56
9	11	28
9	19	28
17	19	28
21	23	193
22	23	193

GridNet

Node 1	Node 2	Total Link Capacity
1	2	2
1	3	1
1	4	2
1	5	2
1	6	2
1	8	1
2	3	2
2	7	2
2	10	2
3	6	1
3	7	2
3	9	1
3	10	2
4	5	2
4	6	1
4	7	2
4	8	2
4	9	1
5	6	1
5	8	2
5	9	1
6	8	1

7	10	2
8	9	1
8	10	2

Qwest

Node 1	Node 2	Total Link Capacity
1	2	48
1	3	96
1	4	24
2	4	96
2	10	96
2	14	96
3	5	96
3	7	96
3	8	48
3	9	96
3	10	48
4	5	96
4	6	96
4	10	96
4	14	96
5	6	96
6	10	96
6	11	96
7	8	384
7	9	96
7	10	96
7	11	96
7	12	96
7	13	144
8	9	48
8	10	48
8	11	48
8	12	48
8	13	96
10	12	96
10	13	144
12	13	144

Sprint

Node 1	Node 2	Total Link Capacity
1	2	192

1	3	3
1	5	144
2	15	192
2	16	96
2	18	96
3	15	3
4	5	96
4	6	96
4	8	96
4	18	96
5	8	192
5	9	192
5	12	96
6	8	96
6	12	288
6	13	96
6	14	12
6	18	192
7	8	96
7	16	96
8	9	48
8	10	48
8	11	48
8	17	48
9	12	96
9	17	48
10	12	48
11	12	48
12	15	144
12	18	96
13	14	192
13	18	48
16	17	144

UUNet

Node 1	Node 2	Total Link Capacity
1	2	48
1	3	50
1	4	49
1	5	61
1	6	13
1	7	16
1	8	13
1	9	12
1	10	13
1	12	1
1	21	1

1	22	1
2	3	205
2	10	12
2	11	208
2	15	48
2	16	60
2	28	1
2	30	3
2	33	3
2	35	13
2	36	1
2	41	3
2	42	3
2	43	3
2	44	3
2	45	3
3	10	12
3	11	193
3	12	48
3	13	12
3	17	48
3	20	12
3	23	1
3	26	1
3	27	13
3	28	14
3	29	1
3	30	18
3	32	3
3	35	15
4	5	12
4	13	60
4	14	61
4	15	13
4	17	12
4	20	12
4	21	1
4	23	14
4	24	13
4	26	12
5	10	61
5	13	48
5	17	48
5	20	12
5	23	13
6	9	48
6	10	12
6	13	13
7	31	3
7	32	3

8	10	60
9	10	48
9	12	48
9	13	48
10	11	60
10	12	48
10	13	13
11	13	12
11	15	61
11	17	48
11	20	12
11	28	1
11	30	3
11	36	1
11	37	12
11	42	3
11	43	3
11	44	3
13	18	25
13	19	13
13	20	12
14	15	60
14	24	12
15	24	12
15	25	1
15	38	12
15	39	12
15	40	12
16	34	1
17	23	13
18	19	1
20	23	1
22	35	12
25	38	12
26	28	1
27	28	13
28	34	1
28	36	1
28	37	1
29	30	1
30	31	1
30	32	4
30	33	12
30	45	3
32	33	1
38	39	12
38	40	12
41	42	3
41	43	1
42	44	3

42	48	4
43	44	3
43	47	3
43	48	1
44	46	3
44	47	6
44	48	1
46	47	3

XOnet

Node 1	Node 2	Total Link Capacity
1	2	24
1	3	434
1	4	386
1	5	386
2	3	36
3	6	386
4	7	386
4	8	386
4	9	193
5	6	386
5	7	386
6	7	386
6	8	386
7	9	386

A.2. *TangNet* Topologies from [TaRu05]

In Section 3.5.5 we used three sample networks from [TaRu05] for performance comparison between HP-tree design and the tree optimal design formulation. Topology vectors for these tree networks are provided in the following. The third column indicates here the working capacity on each link.

TangNet1

Node 1	Node 2	Working Capacity
1	2	10
1	3	6
1	4	6
2	4	10
2	5	4
2	6	10
3	4	8

3	7	2
3	9	6
4	5	6
4	7	8
4	8	12
5	6	8
5	7	4
5	8	4
6	8	4
6	10	8
7	8	4
7	9	4
8	9	8
8	10	6
9	10	4

TangNet1

Node 1	Node 2	Working Capacity
1	2	16
1	15	18
1	3	2
2	14	28
2	3	14
2	4	6
11	13	22
11	15	42
12	13	22
14	15	12
3	15	20
4	14	12
4	15	10
5	12	6
5	8	22
6	14	16
6	7	4
6	8	16
7	14	32
7	8	20
7	9	14
7	10	10
8	11	28
8	12	16
8	9	14
10	11	4
10	13	12
10	15	18

TangNet3

Node 1	Node 2	Working Capacity
1	2	4
1	3	4
1	4	6
1	5	10
2	3	4
2	5	12
3	11	12
3	4	6
3	5	10
4	5	10
4	6	6
5	6	6
5	7	28
5	10	16
6	7	8
7	11	6
7	8	14
7	9	12
8	9	2
8	10	4
9	11	6
9	10	4

Appendix B: Theorem Proofs

In the following, mathematical proofs for the theorems in Section 2-6 are presented from [TaRu05].

Theorem 2-2: Let $T(V_T, E_T)$ be a sub-graph of a connected graph $G(V, E)$. T is a spanning tree of G if the following two conditions hold: (1) $|E_T| = |V| - 1$, (2) For all (u, v) in $E - E_T$, there exists a path P between u and v such that for all link l in P, l is in E_T.

Proof: Condition (1) is required for establishing the acyclic characteristic of the tree. Now all is required is to prove that condition (2) would require the subgraph T to be connected and to include all nodes of the graph ($|V_T| = |V|$). First to prove that T is connected by contradiction. Assume T is not connected, then T has at least two components, T_1 and T_2. Since G is connected, there must exist an edge (u, v) in $E\text{-}E_T$ with u in T_1 and v in T_2. By condition (2), there is a path in T that connects u and v. This means that T_1 and T_2 are connected in T, which is a contradiction.

Next to prove $|V_T| = |V|$ by contradiction. Assume $|V_T| \neq |V|$, then there exists a node u in $V - V_T$. Since G is connected, u must be connected with some node v in V_T and (u, v) in $E - E_T$. By condition (2), there exists a path in T that connects u and v. This means that u is in V_T, which is a contradiction. We have proved that $T(V_T, E_T)$ is connected and $|V_T| = |V|$. Combined with condition (1), T must be a spanning tree of G. *QED*■

Theorem 2-3: Given a two-edge connected graph G and a spanning tree T of G, for any tree links there exists a path between the two end nodes of the link that contains exactly one non-tree link.

Proof: Let $e = (u, v)$ be a tree link. Removing e from T will break T into two components C_1 and C_2. Since G is two-edge connected, by Menger's theorem (see Section 2-1) there must exist a path between u and v in G that does not contain e. Thus, there must exist a non-tree link, say (m, n), with one end node in C_1 and the other end node in C_2. Without loss of generality, assume m is in C_1 and n is in C_2. (It's possible that $m = u$ or $n = v$.) Since both C_1 and C_2 are part of T and each of them is a connected component, there exists a tree path P_1 between u and m in C_1 and there exists a tree path P_2 between v and

n in C_2. Hence, the path $P_1 \cup (m,n) \cup P_2$ is a path between u and v that contains exactly one non-tree link (m, n). *QED*■

Appendix C: Simulated Random Networks

In this section we present detailed description of the sets of random networks used in the simulations in Chapter 3. The networks were all generated using the degree- and distance-controlled algorithm in Section 2.2.2. The input to the algorithm for each graph included the following:

- N : Total number of nodes
- M : Total number of links
- *Dmin*: Minimum nodal degree of a node in the network

We used a maximum nodal degree of 7. The range of N was from 10 to 120 nodes, and the average nodal degree was changed from 2.6 to 5 for networks of 80 nodes and lower. For networks with 90 to 120 nodes, we changed the average nodal degree from 3 to 4. The minimum nodal degree of the network was set to three values: 2, 3 and 4 as long as it was less than the average nodal degree. Therefore for instance for average nodal degree of 3.4, the *Dmin* = 2 and *Dmin* = 3 were considered. For smaller values of N (70 and less), 100 networks were generated for each set of parameters. For larger values of N, this number was reduced to 60 because of complexity of random graph generation and difficulty of computing all end-to-end paths for the full demand matrix. A total of slightly more than 12000 random graphs were generated in this way.

Table C.1 presents the sets of parameters and the number of graphs generated using each set of parameters. Figures C-1 and C-2 show the distribution of radius and number of 3-node chains.

Table C-1: Number of random graphs for each parameter set

		# of Graphs		
N	***M***	***Dmin=2***	***Dmin=3***	***Dmin=4***
10	13	100	0	0
10	15	100	0	0
10	17	100	100	0
10	19	100	100	0
10	21	100	100	100
10	23	100	100	100
10	25	100	100	100

20	26	100	0	0
20	30	100	0	0
20	34	100	100	0
20	38	100	100	0
20	42	100	100	100
20	46	100	100	100
20	50	100	100	100
30	39	100	0	0
30	45	100	0	0
30	51	100	100	0
30	57	100	100	0
30	63	100	100	100
30	69	100	100	100
30	75	100	100	100
40	52	100	0	0
40	60	100	0	0
40	68	100	100	0
40	76	100	100	0
40	84	100	100	100
40	92	100	100	100
40	100	100	100	100
50	65	100	0	0
50	75	100	0	0
50	85	100	100	0
50	95	100	100	0
50	105	100	100	100
50	115	100	100	100
50	125	100	100	100
60	78	100	0	0
60	90	100	0	0
60	102	100	100	0
60	114	100	100	0
60	126	100	100	100
60	138	100	100	100
60	150	100	100	100
70	91	100	0	0
70	105	100	0	0
70	119	100	100	0
70	133	100	100	0
70	147	100	100	100
70	161	100	100	100
70	175	100	100	100
80	104	100	0	0

80	120	100	0	0
80	136	100	69	0
80	160	60	60	0
90	135	60	0	0
90	158	60	60	0
90	180	60	60	0
100	150	60	0	0
100	175	60	60	0
100	200	60	60	0
110	165	60	0	0
110	193	60	60	0
110	220	60	60	0
120	180	60	0	0
120	210	60	60	0
120	240	54	0	0

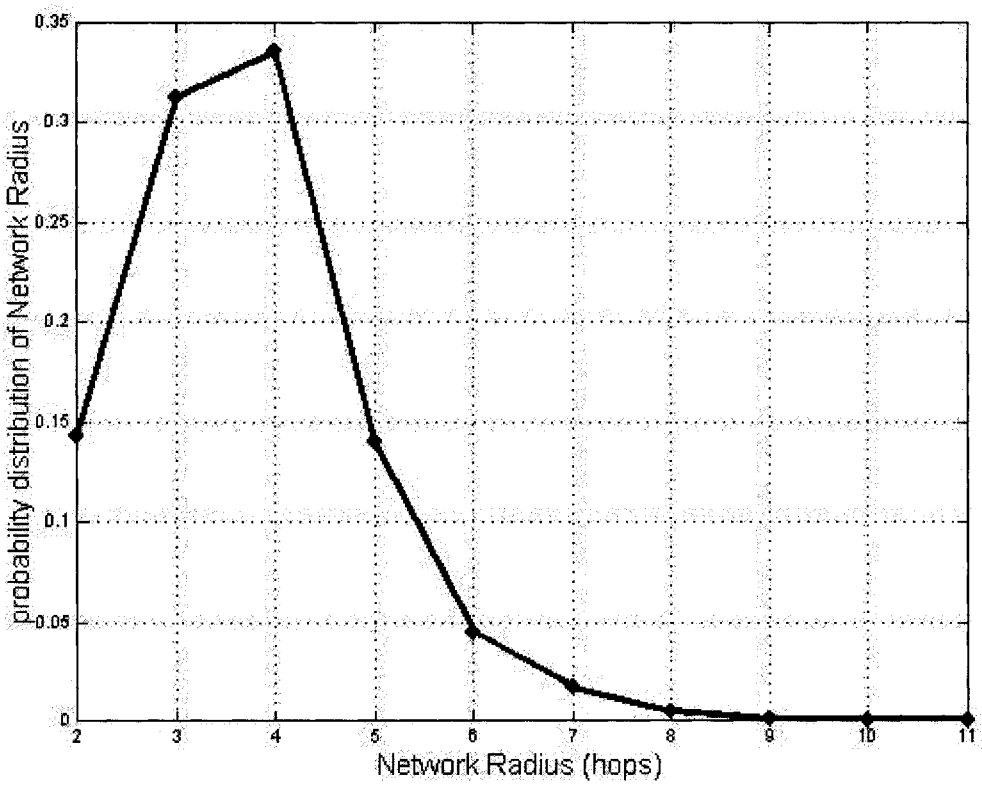


Figure C-1: Distribution of Network Radius in the Random Graph Database

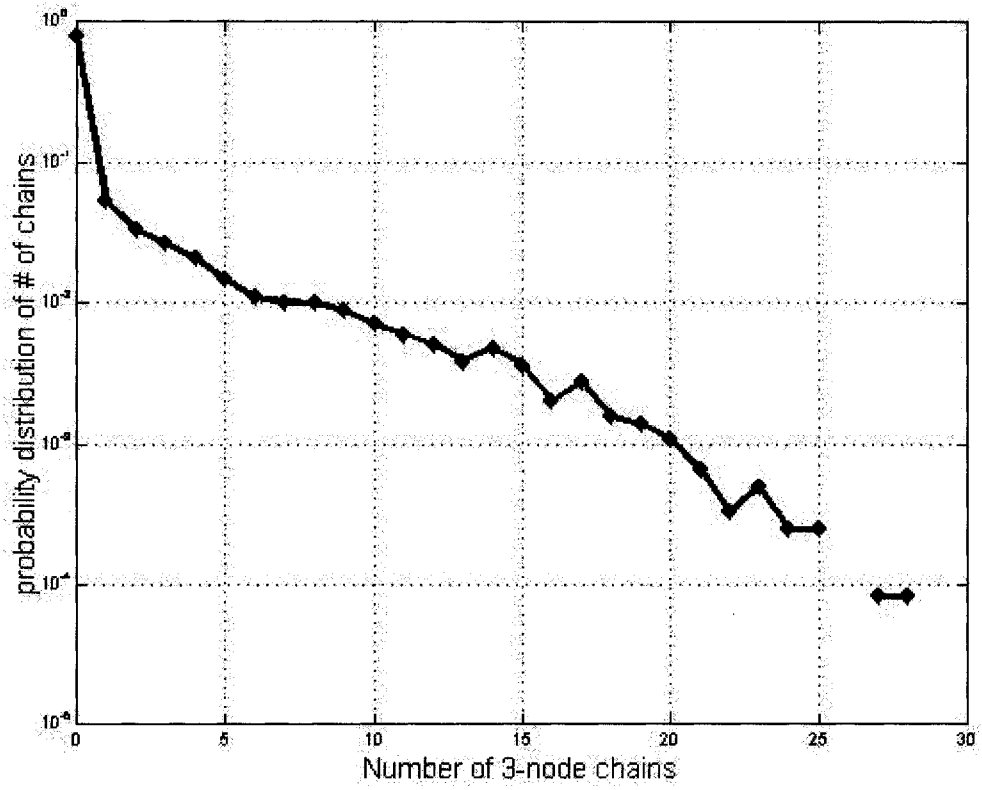


Figure C-2: Distribution of Number of Chains in the Random Graph Database

Note - In Figure C-2, there is a discontinuity of the curve at N=26 because no network in our random graph database had 26 chains, therefore the number was zero at that point and did not show on logarithmic scale.